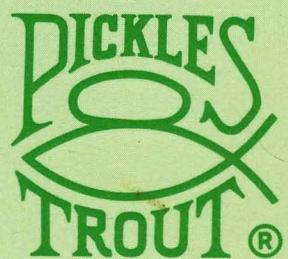


# P&T CP/M® 2 USER'S MANUAL



**P&T CP/M® 2**  
for the  
**TRS-80 Models II, 12, and 16**  
**User's Manual**

Published by  
Pickles & Trout  
P.O. Box 1206  
Goleta, California, 93116  
U.S.A.

Copyright © 1983 Pickles & Trout

All rights reserved. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language, in any form or by any means, electronic, magnetic, optical, chemical, manual, or otherwise, without the prior written permission of the publisher. Printed in the United States of America.

10 9 8 7 6 5

Pickles & Trout is a registered trademark of Pickles & Trout  
CP/M is a registered trademark of Digital Research, Inc.  
MAC is a trademark of Digital Research, Inc.  
TRS-80 and TRSDOS are trademarks of Tandy Corp.  
Z-80 is a trademark of Zilog, Inc.

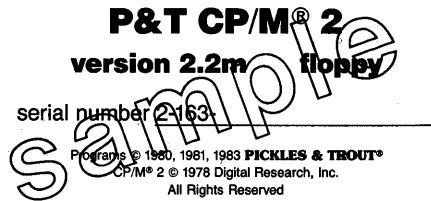
#### IMPORTANT NOTE

You should have received P&T CP/M 2 on a diskette with a label like the one shown below. If you did not receive such a diskette, you may not have received a valid copy of the system. Please contact Pickles & Trout at once. All P&T CP/M 2 labels are printed in green ink on white paper.

If you purchased a registered user's copy of P&T CP/M 2 you are required to transfer the registration to your name. There is a fee for this transfer. If you fail to transfer the registration you will not receive the Pickles & Trout newsletter, will not be eligible for updates to the system, and will not be able to receive assistance from Pickles & Trout.



P.O. Box 1206  
Goleta, CA 93116



All software on this diskette, whether in source or object form, is copyrighted and may be used and copied only under the terms of the Pickles & Trout Software License Agreement. This diskette is serial-  
zed and may be used only by the registered user. Neither it nor the software on it may be distributed, resold, or transferred without the written consent of Pickles & Trout.



#### DISCLAIMER

The publisher has made a reasonable effort to insure that the computer programs described herein are correct and operate properly and that the information presented in this publication is accurate; however they are sold and licensed without warranties either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The publisher is not liable for consequential damages resulting from the use of this product either individually or in concert with other computer programs. Further, the publisher reserves the right to revise this publication and the programs described herein and to make changes from time to time in the contents thereof without obligation of the publisher to notify any person or organization of such revision or changes.

## TABLE OF CONTENTS

### 1. INTRODUCTION

General Comments .....	1.1
What is an Operating System? .....	1.1
P&T CP/M 2 Capabilities .....	1.2
Files on the Disk .....	1.4
Other Documentation .....	1.5

### 2. NOTATION

Conventions of Notation .....	2.1
-------------------------------	-----

### 3. GETTING ON THE AIR

Introduction .....	3.1
Making a Working System Diskette .....	3.1
Operational Notes .....	3.3

### 4. THE SYSTEM MENU

Introduction .....	4.1
General Comments .....	4.3
Selection Menus .....	4.5
AE (Auto Execution) .....	4.8
AK (Save Autokey Strings) .....	4.9
AS (Set I/O Device Assignment) .....	4.10
CA (Change CCB Port Number) .....	4.11
CD (Copy Diskette) .....	4.12
CE (Non-destructive Check of Diskette) .....	4.13
CH (Clean Read/Write Heads) .....	4.14
CP (Change Console Parameters) .....	4.15
CS (Set CCB Date and Time) .....	4.16
DM (Disable Menu Options) .....	4.17
DP (Set Drive Step Rate) .....	4.18
DT (Display Date and Time) .....	4.20
EX (Exit Menu) .....	4.21
FR (Freeze I/O Parameters) .....	4.22
GD (Generate Data Diskette and Copy Files) .....	4.23
GS (Generate System Diskette) .....	4.25
HZ (Change Power Frequencies) .....	4.27
KT (Save Key Translations) .....	4.28
LA (Set Last Address Used by CP/M) .....	4.29
MT (Test Computer's Memory) .....	4.30
ND (Format a New Diskette) .....	4.31
PP (Set Parallel Port Parameters) .....	4.32
SD (Set System Date) .....	4.33
SM (Select System Modules) .....	4.34
SP (Set Serial Port Parameters) .....	4.35
ST (Set System Time) .....	4.36
SY (Synchronize System Date and Time with CCB) .....	4.37
TE (Test Diskette) .....	4.38

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

A

G

I

<b>5. OVERVIEW OF CP/M</b>	
Memory Usage .....	5.1
Disk Storage .....	5.2
Files and Disk Storage .....	5.4
Disk Allocation Blocks .....	5.7
Random Access Files .....	5.8
File Attributes .....	5.8
Non-Disk I/O .....	5.9
Command Structure .....	5.11
DIR Command .....	5.13
ERA Command .....	5.14
REN Command .....	5.15
SAVE Command .....	5.16
TYPE Command .....	5.17
USER Command .....	5.18
Transient Commands .....	5.19
Boots: Warm and Cold .....	5.19
Console Input .....	5.20
<b>6. MODULARITY</b>	
Introduction .....	6.1
Types of Modules .....	6.2
Selecting Modules (MODSEL) .....	6.3
<b>7. UTILITY MODULES</b>	
Introduction .....	7.1
AUTOKEY Module .....	7.1
KEYXLATE Module .....	7.4
SCRNDUMP Module .....	7.6
ADM3A Module .....	7.6
<b>8. UTILITY PROGRAMS</b>	
Introduction .....	8.1
Command Line Mode .....	8.1
List of Utility Programs .....	8.2
ASM .....	8.4
ASSIGN .....	8.9
CLEAN .....	8.13
CLONE .....	8.15
DATIME .....	8.27
DDT .....	8.29
DENSITY .....	8.36
DISKCHK .....	8.39
DISKTEST .....	8.46
DUMP .....	8.55
ED .....	8.56
ERROR .....	8.57
FASTCOPY .....	8.58
FORMAT .....	8.69
KXEDIT .....	8.78
LOAD .....	8.82
MODSEL .....	8.84
PATCH .....	8.85
PIP .....	8.90
SETCCB .....	8.104
SETDATE .....	8.108
SETMISC .....	8.111

SETTIME .....	8.120
SETUP .....	8.122
STAT .....	8.130
SUBMIT .....	8.137
SYNCRO .....	8.140
TRS2CPM .....	8.141
VERIFY .....	8.149
XSUB .....	8.151
Examples of Using the Command Mode .....	8.154

## 9. SYSTEM ERROR MESSAGES

Introduction .....	9.1
BDOS Error Messages .....	9.2
BIOS Error Messages .....	9.3
Warm Boot Error Messages .....	9.3
System Load Error Messages .....	9.4
Module Error Messages .....	9.5

## 10. CONSOLE DISPLAY

General Comments .....	10.1
Display Control Codes .....	10.1
Cursor Addressing .....	10.4
Graphics Mode .....	10.6
Languages with Auto New Line .....	10.6
Languages with Buffered Output .....	10.7
Converting TRS Basic Programs .....	10.7
Using Console Functions .....	10.8
Special System Functions .....	10.9
Line Wrap .....	10.11
Direct Video Access .....	10.12
SCRNDUMP Module .....	10.13

## 11. THE KEYBOARD

General Comments .....	11.1
Direct Keyboard Input .....	11.2
Type-ahead .....	11.3
The <break> Trap .....	11.4
AUTOKEY Module .....	11.5
KEYXLATE Module .....	11.5

## 12. SERIAL I/O

General Comments .....	12.1
SIO Protocols .....	12.1
Serial Port Parameters .....	12.2
Accessing the Ports .....	12.3
Connecting to the Ports .....	12.4

## 13. PARALLEL PRINTER PORT

General Comments .....	13.1
Parallel Port Options .....	13.2
Special Notes (Radio Shack Printers) .....	13.3
Error Messages .....	13.3

## 14. I/O ASSIGNMENT

IOBYTE .....	14.1
--------------	------

## **15. NOTES ABOUT BIOS**

General Comments .....	15.1
Register Preservation .....	15.1
BIOS Jump Table .....	15.2
System Parameter Area .....	15.4
Interrupts .....	15.4

## **16. SPECIAL SYSTEM FUNCTIONS**

Introduction .....	16.1
Using Special System Functions from Assembly Language .....	16.1
Using Special System Functions from High Level Languages .....	16.2
#0 Set up serial ports .....	16.7
#1 Read from serial port A .....	16.8
#2 Read from serial port B .....	16.9
#3 Output to serial port A .....	16.9
#4 Output to serial port B .....	16.10
#5 Read serial port A status .....	16.10
#6 Read serial port B status .....	16.11
#7 Read parallel port status .....	16.11
#8 Output to parallel port .....	16.12
#9 Set parallel port options .....	16.12
#10 Set parallel printer page length .....	16.13
#11 Set parallel printer lines/page .....	16.13
#12 Set parallel printer top of page .....	16.14
#13 Set all drives to unknown density .....	16.14
#14 Read real time clock .....	16.15
#15 Read time of day clock .....	16.15
#16 Set time of day clock .....	16.16
#17 Read XY location of cursor .....	16.16
#18 Read character at cursor location .....	16.17
#19 Set cursor size and blink .....	16.17
#20 Set cursor blink and on/off .....	16.18
#21 Enable access to screen .....	16.18
#22 Disable access to screen .....	16.19
#23 Set split screen mode .....	16.19
#24 Read system date .....	16.20
#25 Set system date .....	16.20
#26 Set ctl-C trap .....	16.21
#27 Set/clear drive flag .....	16.21
#28 Set/clear drive read/write flag .....	16.22
#29 Reserved .....	16.22
#30 Send break to serial port .....	16.23
#31 Terminal emulation on/off .....	16.23
#32 Initialize flashing message .....	16.24
#33 Flash message .....	16.25
#34 Restore display after flashing message .....	16.26
#35 Read disk status .....	16.26
#36 Reserved .....	16.27

## **APPENDICES**

### **GLOSSARY**

### **INDEX**

## 1.1 General Comments

P&T CP/M 2 is a customization of the popular CP/M operating system for the TRS-80 Models II, 12, and 16. This manual is intended to provide the information necessary to make use of P&T CP/M 2. It provides a brief introduction to the CP/M operating system in general and specifically describes the extensions that P&T CP/M 2 adds to standard CP/M. Also covered are topics that are unique to the TRS-80 Models II/12/16, such as using the serial I/O ports and the video display. Finally, all of the utility routines that have been added by Pickles & Trout are fully documented.

This manual is not a tutorial introduction to the CP/M operating system. In many places it will assume some familiarity on the part of the user with CP/M. If you are just beginning to use CP/M, you should consider acquiring one of the several books currently on the market dealing with using the system. As with any new subject of study, it will take some time to become familiar with CP/M and learn the most efficient ways of using it.

Neither does this manual attempt to instruct you in the use of application programs. Many application programs require some customization to make use of the system console, printer, and other facilities. This manual presents general information about P&T CP/M 2 which is needed to make these customizations, but it is not possible to give exact instructions for every application program available.

If you desire assistance from Pickles & Trout, it is absolutely necessary that your copy of P&T CP/M 2 be registered. To register your copy, you must fill out and return the registration cards you received with the original diskette. If you did not receive the cards, please call Pickles & Trout at once for instructions on registration.

Please realize that we must limit assistance with P&T CP/M 2 to matters concerning the operating system and its supporting utility routines. We would like to help everyone with their programming problems, but that would be impossible. We cannot, for example, provide step-by-step instructions on customizing a specific software package. In such a case, we can only supply general directions and point out the appropriate parts of this manual.

## 1.2 What is an Operating System

Any computer system is made up of a variety of component parts. These components can be loosely defined as falling into two groups: those that are closely associated with the processing unit of the computer and those which are not very closely tied to the processor. The elements of the system that are not closely tied to the computer are commonly referred to as peripherals, since they usually reside around the fringes of the system.

Typically, there is much variation among computer systems in the type and arrangement of peripheral devices. For this reason, it is very undesirable to force each program to deal with the peripherals directly. To do so would impose a heavy burden on the programmer, and it would cause great difficulty in running the same program on different machines. If an accounting program, for instance, is to keep files of information on a diskette, it is not productive to have the applications

## What is an Operating System?

programmer write the diskette interface routines himself. Besides extra time and effort, the routines may require programming in a language the programmer doesn't even know!

The function of an operating system is to provide a unified, coherent way for programs to interact with the resources of a computer system. By providing this standardized interaction with system resources, the operating system makes it possible to write programs for a large variety of computer configurations without knowledge of the details of each one. Such standardization also makes it possible for a number of programs, perhaps written by a number of programmers, to work on the same pool of data and to be coordinated in their actions.

The operating system, then, spares the programmer a formidable task because it shields the program from the mundane but complex details of hardware interaction. The programmer no longer worries whether the system printer is a dot matrix or complete character type, or whether it is interfaced to a serial or parallel port. When a program sends a character to the printer, the operating system sees that it gets to the printer, regardless of how the printer is connected to the system.

CP/M 2 is a powerful single-user operating system for all microcomputers based on 8080 and Z-80. Since its introduction, it has gained rapid and continuing popularity in the small computer field and, in fact, has become one of the few de facto standards in the widely diversified field of microcomputing. Thousands of applications programs written in a dozen or more languages have been developed for computers running CP/M. This diversity allows the programmer to choose from a range of languages not found in most mainframe and mini computer systems. Because of the standardization CP/M has brought to microcomputer systems, it is possible to buy "off the shelf" software that will actually run with no modification on a machine different than the one on which it was developed.

### 1.3 P&T CP/M 2 Capabilities

P&T CP/M 2 provides access to all the peripheral devices and ports which are part of the standard equipment of the TRS-80 Models II/12/16. Each peripheral device is tied into the system with the CP/M standard facilities, allowing programs to use the full power of the TRS-80 Models II/12/16 without resorting to tricks and special I/O handlers.

P&T CP/M 2 is designed on a modular concept which gives you considerable flexibility over the system configuration. For example, this modularity allows you to select only the I/O drivers you need for your system. If you do not have a parallel printer, you need not include the parallel port driver software in the system. You can also choose to include a variety of utility modules that are designed to provide various useful functions (like programmable function keys). Selecting the modules to be included in the system is a very easy task and may be done at any time. To change the modules included in the system, you need merely run the MODSEL utility program and reboot the system.

Many of the utility programs included with P&T CP/M 2 are designed to make them useful in an integrated system of programs. These utility programs have both a prompted mode and a command line mode. In the prompted mode, the program will ask you for all the information (e.g. drive letters, if you want verification, etc.) it needs in order to perform its job. In the command line mode all of the necessary information is included on the command line that executes the program. Figure 1.1 gives an example of using the FORMAT utility program in the command line mode to

format the diskette on drive B at double density and verify that there are no bad spots on the diskette.

FORMAT DR=B,DD,V

Figure 1.1 Sample Command Line for FORMAT Program

If you use a particular program frequently, you may want to use the command line mode to avoid having to answer the questions the program asks in the prompted mode. The command line mode also makes it possible to use standard system utility programs in submit files, with menu systems, and (in some cases) from other programs. Other examples of using the command line mode are given in Chapter 8, UTILITY PROGRAMS.

Full access to the serial ports is available through the standard I/O mechanisms of CP/M and by special system functions. These special functions provide capabilities not normally available with CP/M, such as port status checking. Within the standard serial port routine, several common printer protocols are available (such as RS-232, XON/XOFF, and ETX/ACK) which allow the use of almost any serial printer without special programming.

The parallel printer port is also accessible via the standard I/O functions of CP/M and by special system functions added by P&T CP/M 2. Options available for the parallel printer port allow the use of printers that perform automatic line feeds with carriage returns.

P&T CP/M 2 implements a real time clock which can be accessed by programs for timing purposes. As part of the real time clock, the system maintains a time of day clock and a system date. When the Pickles & Trout CCB Clock/Calendar/Bell board is installed in the computer, the system time of day and date are automatically set whenever the system is reset.

P&T CP/M 2 supports single sided double density, double sided double density, and standard CP/M single density floppy diskette storage. Standard single density diskettes are fully interchangeable with other CP/M systems supporting standard 8 inch single density diskettes. The double density formats are, in general, not interchangeable with other CP/M systems. Figure 1.2 shows the usable capacity and number of directory entries for each of the formats.

format	usable storage	Directory Entries
single density	243 Kbytes	64
double density	596 Kbytes	128
double sided	1210 Kbytes	192

Figure 1.2 Capacities and Number of Directory Entries for Disk Formats

In addition to the greater storage capacity, double density diskettes improve the speed of disk access by approximately a factor of 3 over single density. The operating system detects the density of the diskettes and automatically treats them properly, making diskette density completely transparent to application programs. For most programs, either a single or double density diskette may be used (unless the greater storage of a double density diskette is needed). However, all system diskettes must be double density.

A full function console interface is provided in P&T CP/M 2, including a comprehensive set of display functions and type-ahead capability on the standard console keyboard.

## 1.4 Files on the Disk

The distribution diskette you received should contain the following files:

ASM.COM	The standard CP/M assembler. ( <u>CP/M Operating System Manual</u> , p.47)
ASSIGN.COM	P&T CP/M 2 utility to change the I/O device assignment quickly (e.g. change the system printer from the parallel port to a serial port).
BIOSMODS.PNT	Library of I/O driver modules that can be included in the operating system when it is loaded.
BIOSPARM.PNT	Parameter file that contains information about how the system is to be configured when it is booted.
CLEAN.COM	P&T CP/M 2 utility for use when cleaning diskette drive heads.
CLONE.COM	P&T CP/M 2 utility to make an image copy of a diskette.
CRT.DEF	CRT parameter file for using Structured Systems Group software with P&T CP/M 2.
DATIME.COM	P&T CP/M 2 utility to display the current system date and time.
DDT.COM	The standard CP/M debugger. ( <u>CP/M Operating System Manual</u> , p.69)
DENSITY.COM	P&T CP/M 2 utility to report and, optionally, change the density flag on a diskette.
DISKCHK.COM	P&T CP/M 2 utility to check a diskette for flaws. It does not affect the data on a diskette and may be used at any time.
DISKTEST.COM	P&T CP/M 2 utility to test a diskette for flaws. This program destroys any data on a diskette and should be used only on a new diskette or before reusing one.
DUMP.COM	Standard CP/M 2 program to display a disk file in hexadecimal format.
ED.COM	Standard CP/M 2 line oriented editor. ( <u>CP/M Operating System Manual</u> , p.33)
ERROR.COM	P&T CP/M 2 utility to explain error messages given by the system.
FASTCOPY.COM	P&T CP/M 2 utility to make a file by file copy of all files under all user numbers from one drive to another. FASTCOPY reads as many files from the source drive into memory as will fit before beginning to write them out to the destination drive.
FORMAT.COM	P&T CP/M 2 utility to format diskettes.
KXEDIT.COM	P&T CP/M 2 utility to define and edit the keyboard character translations performed by the KEYXLATE utility module.
LOAD.COM	Standard CP/M 2 utility that creates an executable "COM" file from the "HEX" file generated by an assembler. ( <u>CP/M Operating System Manual</u> , p.16)
MENU.COM	P&T CP/M 2 system menu program to assist you in common system operations like generating new diskettes
MENUOLY1.COM	Additional portions of the system menu that are loaded as overlays.
MODSEL.COM	P&T CP/M 2 utility to select the modules that are to be included in the system when it is loaded.
PAGE.COM	P&T CP/M 2 utility which sends a form feed (page eject) character to the system printer.

PATCH.COM	P&T CP/M 2 utility for installing P&T supplied patches to programs.
PIP.COM	Standard CP/M 2 program for file transfer. ( <u>CP/M Operating System Manual</u> , p.17)
SAMPLE.TXT	A sample text file for use in trying out the VEDIT text editor.
SETCCB.COM	P&T CP/M 2 utility for setting the P&T CCB clock, calendar, and bell board.
SETDATE.COM	P&T CP/M 2 utility for setting the system date.
SETMISC.COM	P&T CP/M 2 utility for setting miscellaneous I/O parameters such as the cursor size and blink rate, floppy drive stepping rate, parallel printer port options, etc.
SETTIME.COM	P&T CP/M 2 utility for setting the system time.
SETUP.COM	P&T CP/M 2 utility for setting the serial port parameters and the I/O device assignments.
STAT.COM	Standard CP/M 2 utility for displaying statistics on disk drives and files. ( <u>CP/M Operating System Manual</u> , p.10)
SYNCRO.COM	P&T CP/M 2 utility for synchronizing the system date and time to the P&T CCB clock, calendar, and bell board.
SUBMIT.COM	Standard CP/M 2 utility for submitting multiple command lines for batch processing. ( <u>CP/M Operating System Manual</u> , p.25)
TRS2CPM.COM	P&T CP/M 2 utility to transfer files from a TRSDOS diskette to a CP/M diskette.
VEDIT.COM	Preconfigured copy of the special P&T CP/M 2 version of the VEDIT full screen editor. This program is ready to run as is.
VEDIT.SET	Unconfigured copy of the special P&T CP/M 2 version of the VEDIT editor. This file must be configured before it can be used.
VEDSET.COM	The program for configuring the special P&T CP/M 2 version of the VEDIT full screen editor.
VERIFY.COM	P&T CP/M 2 utility to verify that a copy of a utility program is correct. This program may be used periodically to insure that nothing has damaged the copies of programs you are using.
XSUB.COM	Standard CP/M 2 utility that allows a submit file to feed console input into running programs.

In addition to these files there are several files with names having the form "FIGxxxx.yyy". These files contain the source code used in the examples given in various figures throughout this manual. The "xxxxx" will be replaced by the number of the figure with which the file is associated. These files have been included as a convenience to you in case you want to use some of the code appearing in the examples.

## 1.5 Other Documentation

The standard Digital Research manual CP/M Operating System Manual is included with the documentation for P&T CP/M 2. You will be referred to this manual, in some instances, for information about standard CP/M 2 features and programs.

In addition to this manual, a number of books have appeared which provide information on the CP/M operating system. Most of these books are written in a

general manner, which allows them to serve most implementations of CP/M. Because of this generality, most of them contain information irrelevant to P&T CP/M 2. In general, their discussion of the standard CP/M utilities should be directly applicable, but most of the information on system generation procedures is not.

## 2.1 Conventions of Notation

For ease of reference, all page numbers in this manual consist of two numbers. The first refers to the chapter number and the second to the page number within the chapter.

Figures within this manual are numbered in a similar way, but the second number denotes the figure, not the page, within the chapter. For example, Figure 5.8 refers to the eighth figure in the fifth chapter. If it is necessary to indicate a specific line within a figure, a hyphen separates the figure and line numbers (e.g. Line 5.8-12 means line 12 of Figure 5.8). Keep in mind that "Line ..." refers to a line of a figure, not a line of the text.

When numbers are used within the manual, they should be considered to be decimal (base 10) unless otherwise noted. A hexadecimal number (base 16) is indicated by appending the letter "h" to the number (e.g. 1Ah). A binary number (base 2) is indicated by appending the letter "b" to the number (e.g. 101b). In figures that represent console displays, this convention will not be used if the program that generated the display does not follow it. Every effort has been made to make the figures representing console displays as accurate as possible. The text relating to a figure will specify the base of the numbers displayed if it is not obvious from the context.

When it is necessary to refer to one of the named keys on the keyboard, the name of the key is enclosed in angle brackets. For example, <enter> refers to the key on the keyboard labeled "ENTER". If you are instructed to type or press <enter> at some point, it is expected that you will press the key labeled "ENTER" rather than typing the 7 characters "<", "e", "n", "t", "e", "r", ">". Note that the <enter> key generates an ASCII carriage return character. The key that generates this character is sometimes labeled "RETURN" on other computers and terminals. In some cases the symbol <cr> is used to denote the ASCII carriage return character.

Control keys and control codes are denoted by the characters "ctl-" followed by a letter and enclosed in angle brackets (e.g. <ctl-A>). In other manuals, control codes are often indicated by a caret or up-arrow immediately preceding the letter (e.g. A). The distinction between control keys and control codes is a fine one. This manual will use the term "control key" to refer to the key that is actually typed to generate a control code. For example, to generate a <ctl-A>, you would type the "A" key on the keyboard while holding down the <ctrl> key; the <ctrl> key functions as a special type of shift key.

The control code is the numeric code that is generated by the keyboard and sent to the computer when a control key is typed. Control codes are sometimes called control characters. Keep in mind that the term "control code" may be used without a reference to the keyboard. For example, some programs may use control codes to perform certain functions, such as manipulating the console display. In this case, the codes are generated by the program and the keyboard is not involved.

Many figures show a dialog between the computer and the user. This technique is used heavily when explaining how to use various utility programs. In these dialogs, characters displayed on the console by the computer are shown in plain text; characters typed in by the user are shown underlined.

In this manual, both the terms "diskette" and "disk" are used. "Diskette" refers only to a floppy diskette. An example would be the diskette you originally received. The term "disk" has a general meaning. It can refer to a floppy diskette or to a hard disk. For example, "disk" will be used in the discussion of the PIP routine for file transfer, since PIP can access any type of disk on the system.

The programming examples in this manual are written in several languages, including assembler, BASIC, and PASCAL. The assembly language examples are compatible with the standard CP/M assembler, ASM. Examples in BASIC are written either in Microsoft Basic-80, Microsoft Basic Compiler, or CBASIC 2. If the version of BASIC is not specified for an example, it is written in Microsoft Basic-80. PASCAL examples are given in Pascal/MT+. All complete programs included as examples have actually been run on a P&T CP/M 2 system to insure that they work properly.

In this manual the term "CP/M" is used in referring to general features and capabilities of the CP/M operating system. The term "P&T CP/M 2" refers specifically to the Pickles & Trout adaptation of the CP/M 2 operating system for the TRS-80 Models II/12/16.

### 3.1 Introduction

In order to protect your P&T CP/M 2 master diskette you should immediately make a working system diskette from it and then store the master diskette in a safe place. You should ALWAYS use a working system diskette when running the system. You should NEVER use the P&T CP/M 2 master diskette except to make a working system diskette. You should NEVER cover the write protect notch on the P&T CP/M 2 master diskette.

P&T CP/M 2 has been designed to make it very easy for you to generate working system diskettes. The master diskette has been configured to execute a special system generation program automatically when you load the system. This program will lead you step by step through the procedure of making a working system diskette.

You should not attempt to defeat the system generation program nor alter anything on the master system diskette. Every working system diskette is capable of performing all of the functions of the master system diskette, including creating new working system diskettes. If you need to make changes, you should make them on a working system diskette, not the master diskette.

### 3.2 Making a Working System Diskette

To make your first working system diskette you should mount the P&T CP/M 2 master diskette on floppy drive 0 (the left hand drive if two drives are installed in the computer) and RESET the computer. If you have a Radio Shack hard disk drive installed, you may need to hold down the <break> and <repeat> keys while RESETing in order to make the system boot from the floppy diskette.

After the system is loaded it should immediately begin running a special program to help you create a working system diskette. You should generate a working system diskette from the master diskette and then store the master diskette away in a safe place. The working system diskette you generate will have all the capabilities of the master diskette. You should NEVER cover the write protect notch on the master diskette. There is no need to write information to the master diskette and covering it's write protect notch is an invitation to trouble.

To generate a working system diskette you will need a blank diskette or a diskette that can be reused. The working system diskette can be either single or double sided and will always be double density. We suggest that you use a new double density certified diskette for your first working system diskette since you will be using it a lot.

The console display created by the system generation program is shown in Figure 3.1. It first asks you whether you have more than one floppy drive (Line 3.1-3). If you have only one floppy drive, you will have to swap the master diskette and the new diskette several times during the generation process. If you have two or more floppy drives, you should give an affirmative response as is shown on Line 3.1-3. The program will then use the second floppy drive (drive B) for generating the working system diskette, eliminating the need for swapping diskettes.

```

1:                               Generate a System Disk
2:
3: Do you have more than one floppy disk drive? [Y/N] y
4:
5:           We will now format (initialize) a disk so that we can
6:           put the Pickles & Trout CP/M operating system on it.
7:
8:           NOTE: Any information on the disk will be DESTROYED!!!
9:
10:          Mount the new disk on drive B and
11:          Press <ENTER> when ready
12:
13: Formatting Single-sided disk in drive B: at Double-density.
14:     Format complete.
15:     Checking disk on drive B:
16:     Checking complete.
17:
18: Copying from drive A: to drive B: at Double Density.
19: System tracks copied.
20: Copying from drive A: to drive B:
21:     Reading BIOSMODS.PNT R/W, DIR, user= 0
22:     .      .      .      .
23:     :      :      :      :
24:     .      .      .      :
25:
26: The new disk is now a working CP/M system disk.
27: Mount it on drive A and RESET the computer

```

Figure 3.1 Making Your First Working System Diskette

The program will then tell you that the new diskette will be formatted before the system is installed on it (Lines 3.1-5 and 3.1-6). It also warns you that any information on the diskette will be destroyed by the format operation. This is a reminder that you should use only a new diskette or one that has no needed information recorded on it. On Lines 3.1-10 and 3.1-11 the program asks you to mount the diskette on drive B and press the <enter> key.

After you press <enter> the format operation begins and is announced by Line 3.1-13. If any errors occur during the format operation, error messages will be displayed. This program makes use of the FORMAT utility program to do the disk formatting; refer to Section 8.17 for an explanation of any error messages that are displayed.

Should an error occur while any utility program is running, the error message from that program will be displayed on the console. After the utility program is finished the system generation program will inform you that the diskette was bad and will ask you to press <enter>. After you press <enter>, the program will begin over again.

After the format is complete, the diskette is checked for bad spots as shown on Lines 3.1-15 and 3.1-16. The check is also performed by the FORMAT utility program; refer to Section 8.17 for an explanation of any error messages that are displayed.

After the diskette is formatted and checked, the system tracks are copied to it as shown on Lines 3.1-18 and 3.1-19. This operation is performed by the CLONE utility program; refer to Section 8.7 for an explanation of any error messages that are displayed.

After the system tracks have been copied to the new diskette, all files are copied from the master to the new diskette (Lines 3.1-20 to 3.1-24). The files are copied using the FASTCOPY utility program; refer to Section 8.16 for an explanation of any error messages that are displayed. After all files are copied to the new diskette, the message shown on Lines 3.1-26 and 3.1-27 is displayed. At this point you should

remove the master diskette and store it in a safe place. You may now mount the new working system diskette on the system drive and RESET the computer to begin using it.

### 3.3 Operational Notes

Everyone will have his or her own way of using a computer system. There is no "best" way to work with a system, but a few general ideas have gained wide acceptance. The following comments result from several years of work with microcomputers and larger computers. They do not represent any absolute truths, but we feel that they are worth considering.

#### 1. Plan Ahead

This notion may seem trite, but planning can make a crucial difference in what you get out of your system. Take a few minutes (or hours, if necessary) and carefully think over what you want to do with your computer and how you intend to do it. It is advisable to adopt a system philosophy and to follow it consistently. It may be modified as necessary, but only with careful consideration.

#### 2. Make Frequent Backups (copies)

If there is one golden rule in working with microcomputers, it is: "**Backup important files often!!!**". The importance of this statement cannot be overemphasized. The agony a system crash, diskette failure, or a program bug can cause is inversely proportional to the frequency of backup. A diskette may cost 3 or 4 dollars, but the information stored on it can cost hundreds or thousands of dollars. **Frequent backups are cheap insurance.**

#### 3. Divide Diskette Usage

It is an undeniable fact of microcomputers that some diskettes will occasionally become unreadable for one reason or another. The one which is lost is usually the working system diskette, since it spends more time in the machine than any other. (Note: If the system is functioning properly, this should be a relatively rare occurrence. In a system that is running 8 hours a day, 5 or more days a week, we lose about one diskette every 3 or 4 months, and it is nearly always the system diskette.)

For this reason, it is usually a good idea to keep system-related files and programs on the system diskette while keeping data files and applications programs on another. By keeping the program generated data files separate from the working system diskette, you will minimize the possibility of losing important information when a diskette dies. If something does happen to the system diskette, it can then be easily regenerated.

You should make up 2 or 3 working system diskettes and transfer all of your standard programs (such as text editors, compilers, and the like) onto them. If you use many different programs, you may wish to make up system diskettes with different combinations of programs on them. In any case, keep one of the diskettes as a backup and put the others into use. Store data files generated by these programs on data diskettes rather than the working system diskette.

It also pays to keep different data diskettes for different tasks. This practice can reduce the number of files which could be lost with a diskette, and it helps maintain order in data storage. Of course, you should still back up important files often.

In some cases (as with a single drive system) it is not practical to make this division of usage. Backup then becomes even more important.

#### 4. Maintain Your Equipment

Computers share with most other machines the need for periodic maintenance. Perhaps the most important maintenance item is the cleaning of diskette drive heads, as dirty heads are a common cause of unreliable system performance. Special head cleaning diskettes offer a very simple solution, and they are available from many sources. The CLEAN utility program supplied with P&T CP/M 2 is specifically designed to be used with these cleaning diskettes. For other aspects of routine maintenance, contact your hardware supplier.

### Apply Power to Your Computer With Care

In general, it is not recommended that power be applied to the computer or any external drives when diskettes are mounted and the drive door closed. If the drive door is open, the read/write head of the drive is not engaged and the diskettes are 99.9% safe from any anomalies that occur while the power is being applied. For complete safety, the diskettes should be entirely removed from the drives when the power is turned on or off.

With most TRS-80 Model II computers manufactured before 1982, it is extremely important to apply power to the various devices in the system in the proper order. Improper application of power will not result in physical harm to the computer, but it is possible to destroy the information stored on a diskette. This information is lost REGARDLESS OF WHETHER THE DISKETTE IS "WRITE PROTECTED" OR NOT.

All of the problems associated with the "power on" sequence result from a diskette being mounted in the built-in drive and the door closed before power is applied to the external drives. If you should attempt to boot the operating system before the external drives are running, an error message will indicate that the system diskette cannot be read. In this event, OPEN THE BUILT-IN DRIVE DOOR BEFORE APPLYING POWER TO THE EXTERNAL DRIVES.

You can avoid the risk to your system diskette by insuring that power is supplied both to the computer and to the external drives before closing the door of the built-in drive. An easy way to do this is to plug the computer and external drives into a switched outlet (or extension cord). By letting this switch turn the system on and off, power is always applied simultaneously to the computer and external drives, and the problem is solved.

### Changing Disks

When changing disks, the operating system must be informed of the change so it can make allowances for it. If you change disks while at the system command level, you need merely press the <break> key after mounting the new diskette. When a program is running, you should change disks only at the times the program allows you to do so (you should see the program documentation to find out when you can change disks). Programs that allow you to change disks call a system function to inform the system of the change so you do not need to press the <break> key.

When each disk is first accessed, P&T CP/M 2 begins keeping a map of its free space in order to determine which areas of the disk may be assigned to files as they are written. The map is initialized (at the very first access) by a reading of the disk's directory. Thereafter, it is updated as storage on the disk is increased

or decreased. All free space maps are kept until a warm boot or disk system reset occurs. Each is then cleared and must be rebuilt, which cannot take place until the next access of the disk.

It is vitally important that the free space maps agree with the actual status of the disks. Otherwise, information could be written to a part of the disk which has been assigned to another file, with disastrous results. As a safeguard, P&T CP/M 2 keeps information regarding the current contents of each disk directory, and this information is compared to the disk's actual directory each time it is accessed. If a discrepancy is found, it is assumed that the disk has been changed and should not be written on, so the system sets the disk to "read only" status. It is still possible to read information from the disk, but an attempt to write on it will result in a "Read Only" error message. (See Chapter 9 for a description of error messages.)

Once a map is made up for a given disk, the corresponding drive is said to be active. When a warm boot is performed, all active drives except the current drive and logical drive A (often one and the same) are returned to an inactive state. (Actually, as noted above, all free space maps are cleared, but the current drive and logical drive A are immediately "reactivated".)

Since the free space map being kept for a given disk is unique to it, it is standard practice to do a warm boot (press <break> at the command level of the system) or reset the disk system whenever a disk is changed. Many programming languages have a standard CP/M function for resetting the disk system when diskettes are changed. (In Microsoft Basic-80 it is called RESET.) In this way, the map is rebuilt and is thus valid for the new disk. NOTE: Disks may be changed with neither of these actions taken, but ONLY if no writing operations are going to be performed on the disk. Any attempt to write on the disk (such as erasing or renaming a file) will result in an error, and the program which is running will be aborted.

A warm boot sets all disks to "read/write" status. Since a warm boot automatically follows a "Read Only" error message (see Chapter 9), information cannot be effectively protected by setting a disk to "read only" status. Such protection is available through one of the standard CP/M system functions (see BDOS Function 28, p.105 of CP/M Operating System Manual), but the only sure method is to uncover the disk's write protect notch. When this notch is detected by an optical sensor in the drive, the drive's write circuitry is disabled.

P&T CP/M 2 also keeps information on the density of a diskette. As a diskette is first accessed, the automatic density selection subsystem determines its density and sets the I/O routines accordingly. A warm boot or a disk system reset will clear the density information for the diskette drives affected. You can therefore change diskette densities when you change diskettes provided you perform a warm boot or disk system reset before next using the diskette.

Note that if you mount a different density of diskette when changing, you will not even be able to read it until a warm boot or disk system reset is performed since the system will still be accessing it at the previous density. A program can also make use of P&T Special System Function 13 (see Section 16.17) to set all drives to unknown density before accessing a drive on which a new diskette of a different density may have been mounted.

Because a system diskette must always be double density, a program must perform a disk system reset (or optionally use Special System Function 13) to access a single density diskette on the system boot drive. NOTE: Any program that permits a non-system diskette (single or double density) to be mounted on logical drive A should prompt the user to mount a system diskette before attempting to return to

the system. If a system diskette is not mounted, the system may attempt to warm boot from a non-system diskette, which is guaranteed to spell disaster.

Some systems (e.g. Model 12 computers and computers with the Radio Shack hard disk installed) do not require a system diskette for a warm boot operation. Therefore the previous paragraph does not apply to these systems.

### Single Drive Systems

P&T CP/M 2 has a built-in mechanism which enables TRS-80 Models II/12/16 with only a single floppy drive to run software designed for multiple drives. As the program is running, different diskettes must be substituted in the one physical drive. While this practice allows you to run software requiring multiple drives on a single drive system, it is not a recommended alternative to the purchase of additional drives. A multiple-drive system will do the same job in much less time. (If your machine operates in a commercial environment, a second diskette drive will often pay for itself in a month or two.) It also simplifies file copying, which tends to encourage frequent backups, and it avoids the risk of human error in the interchange of diskettes.

On a system which does have only one floppy drive, P&T CP/M 2 allows software to act as though up to 4 logical drives are available to it. The operating system automatically assigns a different diskette to each logical drive, and it keeps track of the diskette which is currently mounted in the actual (built-in) physical drive. When it is necessary to access a diskette other than the one in the drive, the appropriate diskette is specified in a mount request on the console display. This mechanism can be used only on a one-drive system. It is not possible, for example, to simulate 4 logical drives when there are 2 or 3 physical drives on the system.

To illustrate the disk swapping process, let us suppose that a program expects to find four logical drives on the system. Running the program would then require 4 different diskettes. Logical drive A would be assigned to one diskette we shall call "disk A," logical drive B to one called "disk B", and so on. When the program needs to access a particular drive and the appropriate diskette is not currently mounted, the special message shown in Figure 3.2 is flashed in the center of the console display.

Please mount disk x on main drive and hit enter

Figure 3.2 Diskette Swapping Prompt

The "x" in the request would be replaced by the letter of the needed logical drive. If, for example, disk B is currently mounted and the program needs the third logical drive, the message would call for disk C. Once you have mounted the requested diskette, the system will continue when you press the <enter> key. After you press <enter>, the flashing message will disappear and the console display will be returned to its state before the message was given. The swapping process continues until the program is finished.

In order to hold diskette swapping to an absolute minimum, P&T CP/M 2 does not issue a mount request (Figure 3.2) until an actual operation needs to be performed on the disk. For this reason, there can be situations when the CCP command prompt on the console (indicating the current default drive) does not agree with the diskette currently mounted on the drive.

As an example, suppose that drive A is the current default drive and that disk A is mounted on the drive. You wish to list the directory on drive B, so you type "DIR B<enter>". A message on the console then instructs you to mount disk B. After you mount the disk and press <enter>, the directory is listed. Now the CCP prompt on the console shows "A", the current default drive, even though disk B is on the drive. There will be no request to remount disk A until the system actually attempts to access it. Except for mild confusion, the situation poses no potential problems. The appropriate message will be given as it is needed.

Swapping diskettes to simulate a multi-drive system can lead to errors. In particular, there is always the danger of mounting the wrong diskette. It is a very good idea to invest in a package of small removable labels. If you need to do a lot of diskette swapping, you can use these labels to temporarily mark each diskette with its logical drive letter (ie. "A", "B", etc.). However, it is an even better idea to invest in a second disk drive.

### Drive Not Ready

When the system attempts to access a floppy drive that is not ready, the message shown in Figure 3.3 is flashed in the center of the console display. A drive may be "not ready" if there is no diskette mounted, the door is not closed, the diskette is inserted backwards, or a double sided diskette is mounted on a single sided drive. In rare cases this may also be caused by a hardware problem.

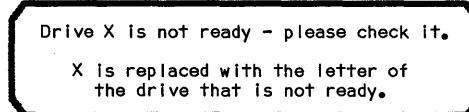


Figure 3.3 Message Flashed if a Drive is Not Ready

While the message is flashing on the screen, there are three actions you may take:

1. You may make the drive ready (by mounting a diskette and closing the door).
2. You may press the <break> key to perform a warm boot and return immediately to the operating system. Note that in this case any program that was running when the error occurred will be terminated.
3. You may press the <F1> key which will perform a warm boot to return you to the system and will make drive A the current drive regardless of what the current drive was previously. As with case 2, pressing <F1> will terminate any program that was running when the error occurred.

After you take one of these actions, the console display will be restored to its previous condition before the system continues. In this way, the console display is not disturbed when a drive is found to be "not ready" while a program is running. After you make the drive ready, the program will continue just as if nothing had happened.

## NOTES

## 4.1 Introduction

The P&T CP/M 2 system MENU has been developed to help you with many of the everyday operations you will need to perform in the course of running the system. It provides a list of the various operations that can be performed and allows you to choose the ones you want. It prompts you for any needed information such as disk drive letters and then performs the requested function. In some cases this function is performed by the MENU itself while in other cases, MENU actually uses P&T CP/M 2 utility programs.

Should any errors occur while a particular function is being performed, the MENU informs you of them and allows you to take remedial action, if possible.

MENU is actually organized as a set of nested menus. A selection on a given menu may result in another menu being displayed with further choices. The following list of menu options (each identified by a unique two character code) is organized just as they appear in MENU. If a menu option leads to another menu, the options on the second menu are shown indented. Some options are useful in several different contexts, hence they appear in several different places in the menus. For example, the EX option (exit to CP/M) appears in several places. It performs exactly the same function no matter which menu is displayed at the time you select it.

**DF Disk and File Operations.** Leads to another menu with options for preparing a diskette for use and copying data to it.

**ND Prepare a new disk for use (FORMAT).**

**GS Generate a system disk (SYSGEN).** Allows you to format a diskette, place a copy of the system on it, and optionally transfer files to it.

**GD Generate a Data (non-system) disk.** Allows you to format a diskette and copy files to it.

**CD Copy a disk.** Allows you to copy a system or data diskette.

**CE Check a disk for errors (does not affect information on the disk).** Allows you to perform a non-destructive check for bad spots on a diskette.

**TE Test a disk for bad areas (destroys any information on the disk).** Allows you to perform a comprehensive test of a diskette for media flaws. Any data on the diskette will be lost.

**DM Disable Menu functions.** Allows you to remove options from the menu.

**EX Exit to CP/M.** Allows you to return immediately to the command level of CP/M.

**SC Change system configuration (Cursor, Disk step rate, Baud rate, etc.).** Leads to another menu that allows you to set various system parameters.

**SM Select system modules.** Allows you to select the modules that are to be included in the system when it is loaded.

**AS Assign logical devices CON:, LST:, PUN: and RDR:.** Allows you to change the logical I/O device assignments. (e.g. change the system printer to a serial port.)

**SP Set serial port parameters (baud rate, parity, etc).**

**CP Set CRT parameters (cursor size, blink rate, etc).**

- DP **Set floppy disk parameters (Disk step rate).** Allows you to change the head step rate to maximize system performance.
- PP **Set parallel port parameters (suppress extra line feed on Tandy printers).**
- CA **Set CCB port address.** Allows you to set the CCB port address so that the system can access it for reading the date and time or ringing the bell.
- HZ **Set up system for 50 Hz or 60 Hz line frequency.** Allows you to configure the system so that the video display will function properly with either 50 or 60 Hz power.
- LA **Set last memory address used by CP/M (MOVCPM).** Allows you to reserve space at the top of memory which will not be used by the system.
- AE **Set up AUTO EXECUTE command.** Allows you to set up a command line to be automatically executed after the system is loaded or on every warm boot.
- AK **Save AUTOKEY table.** Allows you to make the current strings programmed into the "autokeys" permanent so that they will be in effect every time the system is loaded.
- KT **Save KEY TRANSLATION table.** Allows you to make the current translations made by the KEYXLATE module permanent so that they will be in effect every time the system is loaded.
- FR **Save current system parameters permanently (FREEZE).** Allows you to make a variety of system parameters (serial port configuration, parallel port configuration, console configuration, etc.) permanent so that they will be in effect every time the system is loaded.
- DM **Disable Menu functions.** Allows you to remove options from the menu.
- EX **Exit to CP/M.** Allows you to return immediately to the command level of CP/M.
- CL **Clock Functions.** Allows you to make changes to the system date and time and to set up the system to access a CCB board if one is installed.
- DT **Display system date and time.**
- ST **Set system time.** Allows you to set the current system time. The system time must be set after the system is loaded if your programs make use of it and you do not have a CCB board installed in the computer.
- SD **Set system date.** Allows you to set the current system date. The system date must be set after the system is loaded if your programs make use of it and you do not have a CCB board installed in the computer.
- SY **Synchronize system date and time with CCB.**
- CS **Set CCB date and time.** Since the CCB retains the date and time while the computer is off, this is only necessary when time changes take effect or for periodic correction of the time.
- CA **Set CCB port address.** Allows you to set the CCB port address so that the system can access it for reading the date and time or ringing the bell.
- DM **Disable Menu functions.** Allows you to remove options from the menu.
- EX **Exit to CP/M.** Allows you to return immediately to the command level of CP/M.

- AK **Save AUTOKEY table.** Allows you to make the current strings programmed into the "autokeys" permanent so that they will be in effect every time the system is loaded.
- KT **Save KEY TRANSLATION table.** Allows you to make the current translations made by the KEYXLATE module permanent so that they will be in effect every time the system is loaded.
- GS **Generate a system disk (SYSGEN).** Allows you to format a diskette, place a copy of the system on it, and optionally transfer files to it.
- GD **Generate a Data (non-system) disk.** Allows you to format a diskette and copy files to it.
- CD **Copy a disk.** Allows you to copy a system or data diskette.
- MT **Test Memory.** Tests all banks of memory (RAM) in the computer and reports the results.
- DM **Disable Menu functions.** Allows you to remove options from the menu.
- EX **Exit to CP/M.** Allows you to return immediately to the command level of CP/M.

4

## 4.2 General Comments

The MENU performs some operations on its own and uses standard P&T CP/M 2 utility programs for others. While it is running, MENU will need access to its overlay file (MENUOLY1.COM) as well as any utility programs that are needed. MENU assumes that all these files are available on a single disk.

When MENU looks for a program or overlay file, it may not be able to find it. This can occur if you removed the disk on which the files are stored or if you forgot to put a needed program file on the disk. In this case, MENU will display the message shown in Figure 4.2. You may then mount the disk with the required file on the specified drive and press <enter>. MENU will then load the file from the disk you have mounted. If you wish to abort the operation that needs the file, you may press <esc> to do so. In this case, MENU will cease attempting to load the file and return the selection menu.

MENU files are not on the current disk.  
Enter the letter of the drive on which to find them (<ESC> to quit) :

Figure 4.1 Message When MENU Cannot Find Its Files

When MENU looks for a program or overlay file, it may not be able to find it. This can occur if you removed the disk on which the files are stored or if you forgot to put a needed program file on the disk. In this case, MENU will display the message shown in Figure 4.2. You may then mount the disk with the required file on the specified drive and press <enter>. MENU will then load the file from the disk you have mounted. If you wish to abort the operation that needs the file, you may press <esc> to do so. In this case, MENU will cease attempting to load the file and return to its initial display.

- 1: Cannot find xxxxxxxx.yyy on Drive X user N
- 2:
- 3: Mount correct disk and press <ENTER> or press <ESC> to quit

Figure 4.2 Message Given When a File Cannot be Found

Since MENU loads and executes utility programs in some cases, it also needs to reload itself after the utility programs are finished. When reloading itself, MENU always looks for a file named "MENU.COM". For this reason, you must not rename MENU. If you want to execute MENU with another name, you should make a copy of MENU.COM with another name but leave the original MENU.COM file on the disk. In a similiar manner, the MENUOLY1.COM file must not be renamed.

If MENU requests information and your response is invalid, MENU will ignore your response and ask the question again. The appearance on the console display is that the invalid response is erased and the cursor is moved to the position where it is waiting for a response. As an example, when MENU asks you to specify a disk drive, you may enter a logical drive letter in the range A-P or a physical drive number in the range 0-3. If the letter or number you enter is not in one of these ranges or it corresponds to a drive that does not exist on the system, MENU will ignore your response and ask the question again.

Several of the options available with MENU involve copying information from one disk to another. These options will ask you to specify source and destination drives. The **source** drive is the one from which the data is to be copied and the **destination** drive is the one to which the information is to be copied. Note that when generating a system diskette, a copy operation is involved to install the operating system on the new diskette. In this case, the source disk should be a working system diskette (ie. one from which the system can be loaded).

In the discussion of the various MENU options there are two cases where figures are abbreviated to save space. One is when a number of files are being copied. The names of the files are displayed on the console as they are copied. Only one or two file names are shown in the figure. The second abbreviation involves leaving blank lines out of the figures. In some cases, the display presented by MENU will have several lines at the top, one line at the bottom, and several blank lines in between. Some figures leave out most of these blank lines since they are not necessary to understanding how the option works.

Some options involve changes to the system that you may or may not wish to make permanent. For example, if you change the baud rate on a serial port, you might want to make it permanent if you are connecting a printer to the port. You may not want to make the new baud rate permanent if you changed it temporarily to communicate to another computer over the telephone lines. Most changes will stay in effect only until the next time the system is loaded (RESET). When the system is loaded, the default (permanent) system configuration will be in effect.

The default system configuration is stored in a disk file named BIOSPARM.PNT. In order to change the default (or permanent) configuration, MENU must make changes in this file. After you are finished with an option that made changes to the default configuration, MENU will ask you the question shown in Figure 4.3. If you respond affirmatively, MENU will install the changes in the BIOSPARM.PNT file. If you respond negatively, the BIOSPARM.PNT file will not be modified and any changes you have made will remain in effect only until the next time you RESET the computer.

Do you want these changes to remain in effect after RESET? [Y/N]

Figure 4.3 MENU Prompt for Making Changes Permanent

In order to change the BIOSPARM.PNT file, MENU must first locate it. It first looks for the file on the drive from which the system was initially loaded. Assuming it finds the file, it displays the message shown in Figure 44 on the bottom 5 lines of the screen. This message shows you the drive on which the file was found and gives you the option of specifying a different drive. If you just want to use the drive specified, merely press <enter>. If you want to have MENU change a BIOSPARM.PNT file on another drive, enter the drive letter or number. You might want to specify another drive if you have moved the disk from which the system was loaded or if you want to modify some other disk. After you have responded to the question, MENU will restore the bottom 5 lines of the screen to their previous state.

- 1: System parameter file (BIOSPARM.PNT) is on drive X
- 2: Press <ENTER> if this is the one you want to use. Otherwise,
- 3: specify drive on which to find the system parameter file [A-P, 0-3] :

Figure 4.4 MENU Prompt for Locating BIOSPARM.PNT

If MENU cannot find the BIOSPARM.PNT file on the drive from which the system was loaded, it displays the message shown in Figure 45 on the bottom 5 lines of the screen. This allows you to direct MENU to another drive if you have moved the disk. If the disk is not mounted, you should mount it (on any available drive) at this time and then specify the drive in response to the message in Figure 45. After you have responded to the question, MENU will restore the bottom 5 lines of the screen to their previous state.

- 1: Cannot find system parameter file (BIOSPARM.PNT) on drive X
- 2:
- 3: Specify drive on which to find the system parameter file [A-P, 0-3] :

Figure 4.5 MENU Prompt if it Cannot Find BIOSPARM.PNT

MENU will display the prompts shown in Figures 4.4 and 4.5 only once each time you run MENU. After it has located the BIOSPARM.PNT file, it will assume that it can always find the file on the same drive. If it cannot find the file on that drive at a later time, it will display a message as shown in Figure 4.2 and allow you to mount the disk with the file.

### 4.3 Selection Menus

When MENU is first executed, it displays the selection menu shown in Figure 4.6. It also returns to this menu after an option is completed. To select an option you need merely enter the two character code corresponding to that option on Line 4.6-20.

```
1:                                     MENU
2:             Copyright 1983 by Pickles & Trout
3:
4:
5: Type
6: This           To Perform This Function
7:
8: DF  Disk and File Operations
9: SC  Change system configuration (Cursor, Disk step rate, Baud rate, etc.)
10: CL  Clock Functions
11: AK  Save AUTOKEY table
12: KT  Save KEY TRANSLATION table
13: GS  Generate a System disk (SYSGEN)
14: GD  Generate a Data (non-system) disk
15: CD  Copy a disk
16: MT  Test Memory
17: DM  Disable Menu functions
18: EX  Exit to CP/M
19:
20: Selection?
```

Figure 4.6 Initial Selection Menu

If you select option DF on the initial selection menu, you will be presented with another selection menu as shown in Figure 4.7. This selection menu contains options that deal with preparing diskettes for use and copying files to them.

```
1:                                     Disk and File Operations
2:                                     *****
3:
4:
5: Type
6: This           To Perform This Function
7:
8: ND  Prepare a new disk for use (FORMAT)
9: GS  Generate a System disk (SYSGEN)
10: GD  Generate a Data (non-system) disk
11: CD  Copy a disk
12: CE  Check a disk for errors (does not affect information on the disk)
13: TE  Test a disk for bad areas (destroys any information on the disk)
14: CH  Clean disk drive heads
15: DM  Disable Menu functions
16: EX  Exit to CP/M
17:
18: Selection?
19:
20:
21:
22:
23:                                     Press <ESC> to quit
```

Figure 4.7 MENU Display for Disk and File Operations (option DF)

If you select option SC on the initial selection menu, another selection menu, as shown in Figure 4.8 will be displayed. This selection menu contains options that are used to change the system configuration. In general, most of the options presented in this menu will be used only once. Once you have configured the system, you may wish to disable (see option DM) the options you no longer need.

```
1: Change System Configuration
2: *****
3:
4: Type
5: This      To Perform This Function
6:
7: SM Select system modules
8: AS Assign logical devices CON:, LST:, PUN: and RDR:
9: SP Set serial port parameters (baud rate, parity, etc)
10: CP Set CRT parameters (cursor size, blink rate, etc)
11: DP Set floppy disk parameters (Disk step rate)
12: PP Set parallel port parameters (suppress extra line feed on Tandy printers)
13: CA Set CCB port address
14: HZ Set up system for 50 Hz or 60 Hz line frequency
15: LA Set last memory address used by CP/M (MOVCPM)
16: AE Set up AUTO EXECUTE command
17: AK Save AUTOKEY table
18: KT Save KEY TRANSLATION table
19: FR Save current system parameters permanently (FREEZE)
20: DM Disable Menu functions
21: EX Exit to CP/M
22:
23: Selection?
24: Press <ESC> to quit
```

Figure 4.8 MENU Display for Changing the System Configuration (option SC)

If you select option CL on the initial selection menu, you will be presented with another selection menu as shown in Figure 4.9. This menu includes options that deal with system timekeeping functions.

```
1: Clock Functions
2: *****
3:
4:
5: Type
6: This      To Perform This Function
7:
8: DT Display system date and time
9: ST Set system time
10: SD Set system date
11: SY Synchronize system date and time with CCB
12: CS Set CCB date and time
13: CA Set CCB port address
14: DM Disable Menu functions
15: EX Exit to CP/M
16:
17: Selection?
18:
19:
20:
21:
22:
23:
24: Press <ESC> to quit
```

Figure 4.9 MENU Display for Clock Functions (option CL)

**4.4 Option code: AE**

Purpose: **To set up a command line for automatic execution after the system is loaded or a warm boot.**

Programs used: **none**

The AE option allows you to specify a command line that will be automatically executed either when the system is initially loaded (cold boot) or both after the initial load and on every warm boot. The console display for this option is shown in Figure 4.10. MENU first reads the BIOSPARM.PNT file (where the automatic execution information is kept) and reports to you the current state of the auto command line (Lines 4.10-5 to 4.10-7).

```
1:                                         Set up AUTO command
2:                                         *****
3:
4:
5: 5: The command line
6: 6: MENU
7: 7: is executed on each warm and cold boot
8:
9:
10: 10: Do you want to change the command line? [Y/N] Y<enter>
11: 11: Enter new command
12: 12: SUBMIT STARTUP<enter>
13:
14: 14: Do you want to change when the command line executes? [Y/N] Y<enter>
15:
16: 16: 1 Never
17: 17: 2 Cold boot only
18: 18: 3 On each warm and cold boot
19:
20: 20: Selection? 2<enter>
21: 21: Is this correct? [Y/N] Y<enter>
22:
23:
24:                                         Press <ESC> to quit
```

Figure 4.10 MENU Display for Setting the Auto Command (option AE)

On Line 4.10-10 you are given the option of changing the command line. If you indicate that you want to make a change (as shown), you will be prompted for the new command line (Lines 4.10-11 and 4.10-12). You will then be asked if you want to change when the command is executed (Line 4.10-14). If you want to make a change respond affirmatively (as shown) and MENU will present you with the options shown on Lines 4.10-16 to 4.10-17. Enter your choice on Line 4.10-20.

On Line 4.10-21, MENU asks you to verify that what you have entered is correct. If you respond affirmatively, it will make any necessary changes to the BIOSPARM.PNT file and return to the selection menu. If you respond negatively, MENU allows you to re-enter your choice. Note that if you simply want to stop a command line from executing, you should choose option 1.

Note that any changes in the auto executed command line or when it is executed will not take effect until the next time the system is loaded (RESET).

**4.5 Option code: AK**

**Purpose:** To save the strings programmed into the autokeys.  
**Programs used:** none

If you elect to include the AUTOKEY module with the system when it is loaded, you will have available 5 programmable function keys. These keys may be programmed at any time either directly from the keyboard or by a program (see Section 7.2). Normally, the programmed strings are lost if the system is reloaded (RESET). This option allows you to save the current AUTOKEY strings in the BIOSPARM.PNT file so that they still will be in effect the next time the system is reloaded (RESET).

Figure 4.11 shows the console display as the programmed strings are being saved. After you press <enter> MENU will return to the selection menu.

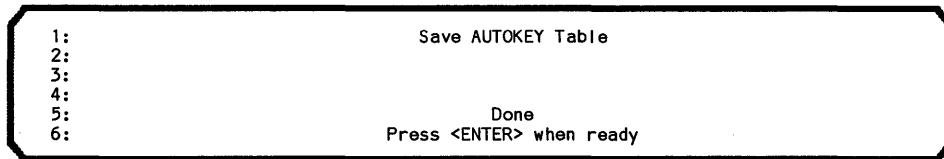


Figure 4.11 MENU Display for Saving the AUTOKEY Strings (option AK)

**4.6 Option code: AS**

**Purpose:** To change the logical-to-physical I/O device assignments.

**Programs used:** ASSIGN

This option executes the P&T CP/M 2 utility program ASSIGN for changing the logical-to-physical I/O device assignments. Refer to Section 5.7 for a discussion of I/O devices and to Section 8.5 for instructions on using ASSIGN.

**4.7 Option code: CA**

**Purpose:** To change the port number used by the system to access a CCB board.

**Programs used:** none

If a Pickles & Trout CCB board is installed in the computer, the system must know how to access it. This option allows you to specify the port address you have assigned to the CCB. Note that a port address of FF is interpreted as meaning that no CCB is installed in the computer. The standard port address for CCB boards is BEh. See the CCB manual for instructions on changing the port address.

Figure 4.12 shows the console display for changing the port address. On Line 4.12-5, the current setting is reported. This example shows the current setting as port FF indicating that no CCB is present. You are then asked if the current setting is correct (Line 4.12-7). If you respond negatively, you are asked to enter the new port address as shown on Line 4.12-8. After entering the new port address, the display will start over with the new port address shown on Line 4.12-5. Again you will be asked if it is correct. You may continue to change the port address as many times as you wish until you have it the way you want it. When you indicate that the port address is correct MENU will return to the selection menu.

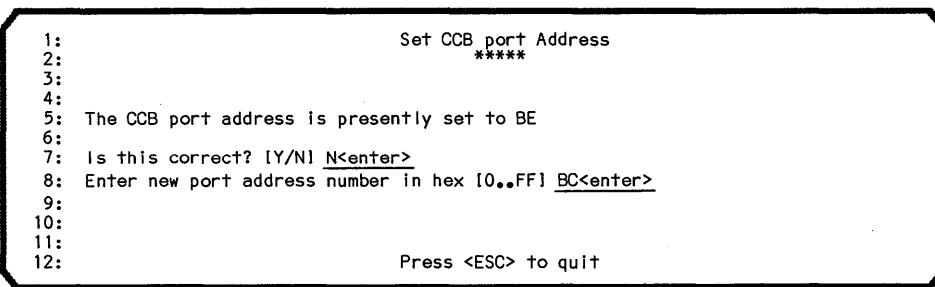


Figure 4.12 MENU Display for Setting the CCB Port Address (option CA)

Note that the CA option of menu asks you for the CCB port number in hexadecimal rather than decimal as did previous versions of P&T CP/M 2.

**4.8 Option code: CD****Purpose:** To make a copy (backup) of a diskette.**Programs used:** **FORMAT, CLONE, FASTCOPY, PIP**

This option allows you to make copies of diskettes for backup purposes. You have a choice of either an image copy or a file-by-file copy. The image copy is faster but it does not remove any file fragmentation which might limit disk performance.

Figure 4.13 shows the console display for making an image copy of a diskette. First you are asked to specify the source and destination drives for the copy operation (Lines 4.13-5 and 4.13-6). Next you are asked if you want to make an image copy (Line 4.13-7). In this example, an image copy is specified. An image copy always formats and checks the destination disk (Lines 4.13-10 to 4.13-15) before copying. The FORMAT program (see Section 8.17) is used for the format and check.

```
1:                               Copy a Disk
2:
3:                               Press <ESC> to quit
4:
5: 5: Copy from disk (SOURCE) [A-P, 0-31] A<enter>
6: 6: Copy to disk (DESTINATION) [A-P, 0-31] B<enter>
7: 7: Do you want an exact image of the source disk? [Y/N] Y<enter>
8:
9:
10: 10: Mount disk to format on drive B: and press <ENTER> when ready to start :<enter>
11:
12: 12: Formatting Single-sided disk in drive B: at Double-density.
13:     Format complete.
14:     Checking disk on drive B:
15:     Checking complete.
16:
17:
18: 18: Mount source disk on drive A:
19: 19: Mount destination disk on drive B:
20: 20: Press <ENTER> when ready : <enter>
21: 21: Copying from drive A: to drive B: at Double Density.
22:     Data and System tracks copied.
23:
24:
25: 25: >>> DISK OK <<<      Do you want to do another disk? [Y/N] N<enter>
```

Figure 4.13 MENU Display for Copying a Diskette (option CD)

The CLONE utility is then used to make an image copy of the source diskette to the destination diskette (Lines 4.13-18 to 4.13-22). See Section 8.7 for a description of CLONE and its error messages. On Line 4.13-25, MENU indicates that the copy was successful and gives you the chance to make another. If you respond affirmatively, the CD option will start over again while a negative response will return you to the initial selection menu.

If you do not select a image copy, you will be given the option of formatting the destination disk and will be given a selection of the types of files to be copied. In this case the format is performed only if you request it and the copy is performed by either FASTCOPY (see Section 8.16) or PIP (see Section 8.22) depending on whether you elect to transfer all or some of the files on the source disk.

**4.9 Option code: CE**

**Purpose:** To perform a non-destructive check on a diskette for bad areas.

**Programs used:** DISKCHK

This option allows you to perform a non-destructive check for bad spots on a diskette. Performing such a check periodically will help you to detect any degradation of a diskette while there is still time to make a backup before data is lost. Figure 4.14 shows the console display for performing such a check.

```
1: Check a Disk for Errors
2: NOTE: Information on the disk will NOT be affected
3:
4: Press <ESC> to quit
5:
6: Drive to be Checked [A-P, 0-3] B<enter>
7:
8:
9:
10: Mount disk to check on drive B: and press <ENTER> when ready to start :<enter>
11:
12: Checking Double-density disk in drive B:
13:     Checking complete.
14:
15:
16: >>> DISK OK <<<      Do you want to do another disk? [Y/N] N<enter>
```

Figure 4.14 MENU Display for Checking (Non-Destructive) a Disk (option CE)

On Line 4.14-6, MENU asks you to specify the drive on which the check is to take place. After receiving this information from you, MENU loads and executes the DISKCHK (see Section 8.11 for information about DISKCHK and its error messages) utility program (Lines 4.14-10 to 4.14-13) to perform the check. Finally, MENU reports the results of the check and gives you the option of checking another one (Line 4.14-16). An affirmative response will cause the CE option to start over while a negative response will return you to the selection menu.

**4.10 Option code: CH**

**Purpose:** To clean the read/write heads of a floppy drive with a cleaning diskette.

**Programs used:** CLEAN

This option loads and executes the CLEAN utility program. This program allows you to load the heads on a specified disk drive and move them around for about 20 seconds. CLEAN is designed to be used with a head cleaning diskette for periodic cleaning of floppy drive read/write heads. See Section 8.6 for further information on using CLEAN. After CLEAN is finished, MENU will return to the selection menu.

**4.11 Option code: CP**

**Purpose: To change system parameters related to the console display.**

**Programs used: MENUOLY1**

After selecting this option MENU will show you a display like that shown in Figure 4.15. You may change any of the listed parameters by entering the number of the parameter. After you enter a number, you will be presented with a variety of choices for that parameter. You may change as many parameters as you wish. Press <esc> when you are finished making changes and wish to return to the selection menu.

```
1: CRT Parameters
2: Press <ESC> to quit
3:
4:
5:
6: 1. Line wrap = enabled
7: 2. First line of cursor = 9
8: 3. Cursor blink = slow
9:
10:
11:
12: Enter number of item to change:
```

Figure 4.15 MENU Display for Changing CRT Parameters (option CP)

Note that these parameters can also be changed by the SETMISC utility program. See Section 8.25 for further information.

**4.12 Option code: CS**

**Purpose:** To set the CCB board's date and time if one is installed in the computer.

**Programs used:** SETCCB

This option allows you to set the date and time of a Pickles & Trout CCB board if one is installed in the system. It loads and executes the SETCCB utility program. See Section 8.23 for information on using SETCCB and the error messages it might display. Note that, since the CCB includes a backup battery, you need set it only occasionally to correct any long term errors that might accumulate (just like a watch).

**4.13 Option code: DM**

Purpose: To disable options from a selection menu.  
Programs used: none

One of the convenient features of MENU is the ability to disable options on a selection menu. Since some of the options are very rarely used you may wish to configure the system and then disable them. This will result in less cluttered selection menus and reduce the possibility of selecting a wrong option.

Disabling selection menu options is also useful when setting up a disk for a non-technical user. After configuring the system, you may disable all options that could allow the neophyte user to make the system un-useable. For example, you might disable the SP option so that the serial port configuration cannot be changed.

When you select the DM option from any selection menu, the menu will be displayed as shown in Figure 4.16. As you enter the two character option codes, the options will be removed from the menu and the menu will be redisplayed.

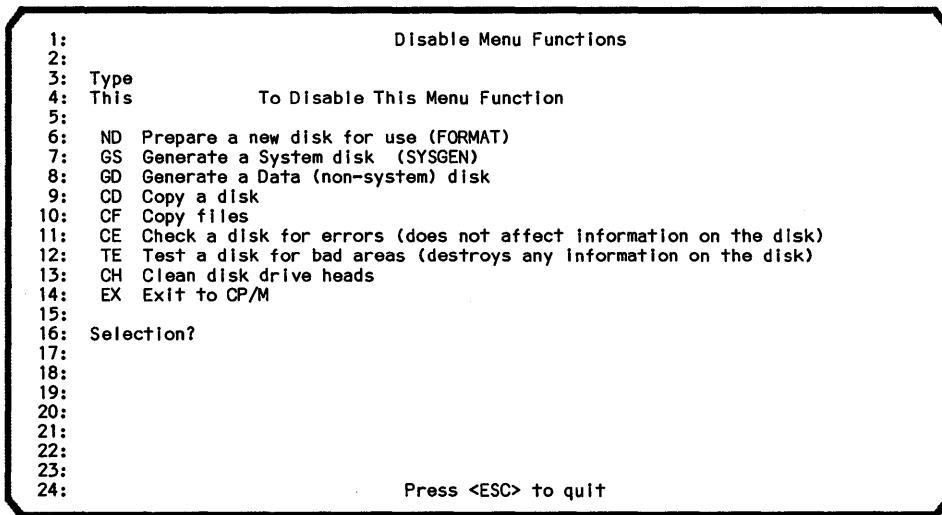


Figure 4.16 MENU Display for Disabling Menu Functions (option DM)

Important notes about the DM option are:

1. Once options are disabled there is no way to bring them back. You must get a fresh copy of MENU.COM to reinstate disabled options.
2. Selecting the DM option disables the DM option itself from a selection menu. This means that you must disable all the options you want disabled from a selection menu at one time.
3. You should NEVER attempt to use the DM option on your master disk.
4. If an undesired option appears in multiple selection menus, you must disable it everywhere that it appears to eliminate access to it.

**4.14 Option code: DP**

Purpose: To set the floppy drive step rates and the number of floppy drives on a floppy only system.

Programs used: none

In order to optimize system performance, P&T CP/M 2 allows you to specify the head step rate individually for each floppy drive on the system. This option allows you to perform this operation. For floppy only systems, it also allows you to specify the number of floppy drives on the system.

The thinline disk drives used in the Models 12 and 16 are capable of a 3 msec step rate while the full width drives used in the Model II cannot move the head that fast. The full width drive should all work at a 10 msec step rate but many of them will also run reliably at 6 msec. You will have to try different values to find out what works with your hardware.

To allow such experimentation, this option allows you to make changes either to the running system (in memory) or permanently in the BIOSPARM.PNT file (which will take effect after the next RESET). If you are trying different values, you should change only the current system until you are satisfied that you have a set of parameters that work properly. Making a permanent change with untested stepping rates can result in a non-functional system disk.

Figure 4.17 shows the console display for setting the floppy parameters. You may use the arrow keys to move the cursor to the parameter you want to change. Note that the cursor can be positioned only in the two "step rate" columns and on the number of drives for the new system (after RESET). You cannot change the number of drives in a running system.

Floppy Disk Parameters	
Press <ESC> to quit	
1:	
2:	
3:	
4:	
5:	Current System
6:	
7:	Number of Floppy Drives: 2
8:	
9:	Drive Name Step Rate
10:	A 3
11:	B 3
12:	- -
13:	- -
14:	
15:	
16:	
17:	
18:	
19:	
20:	
21:	
22:	
23:	
24:	Arrows move cursor smaller: <F1> comma - larger: <F2> period =

Figure 4.17 MENU Display for Changing Floppy Drive Parameters (option DP)

When the cursor is positioned at a parameter you want to change, you may press <F1>, comma, or - to make the parameter smaller. Alternately you may press <F2>,

period, or = to make the parameter larger. The step rates can assume the values 3, 6, 10, and 15 while up to 4 floppy drives can be specified for the new system. When you have finished the changes, press <esc> to return to the selection menu. Any changes you have made in the "new system" column will be recorded in the BIOSPARMPNT file so that they will be in effect the next time the system is loaded (RESET).

Note: If you increase the number of drives, you must reload the system (RESET) before you can change the step rate of the additional drives.

**4.15 Option code: DT**

**Purpose:** To display the system date and time.  
**Programs used:** DATIME

This option allows you to display the current system date and time on the console. The form of the display is described in Section 8.8 which discusses the DATIME utility program.

**4.16 Option code: EX**

**Purpose:** To exit from the MENU program to the command level of P&T CP/M 2.

**Programs used:** none

Selecting this option will return you directly to the command level of P&T CP/M 2. The EX option is the only way to exit MENU so that you can perform other system operations. You should not disable the EX option in all the selection menus; doing so will make it impossible to exit MENU once it is executed.

4.17 Option code: **FR**

Purpose: To make the current I/O parameters permanent so that they will be in effect when the system is reloaded (RESET).

Programs used: none

When you are first configuring the system or adding a new piece of equipment, you may make several changes to the system I/O configuration without making them permanent. After you have arrived at the configuration you want, you may make it permanent by selecting this option.

The console display for this option is shown in Figure 4.18. On Lines 4.18-5 to 4.18-11 it tells you which parameters have been made permanent. It then gives you the option (Line 4.18-13) of saving any strings programmed into the programmable function keys (if AUTOKEY has been included in the system). It also gives you the option (Line 4.18-16) of saving the key translation table if the KEYXLATE module has been included in the system.

After pressing <enter> on Line 4.18-19, MENU will return to the selection menu.

```
1:                                     FREEZE
2:             Save Current System Parameters
3:
4:
5:   The following system parameters have been permanently saved:
6:   * Serial Port parameters (baud rates, parity, etc)
7:   * Parallel printer port parameters (line feed suppression, page length, etc)
8:   * Disk step rates
9:   * CRT parameters (line wrap, cursor size and blink rate)
10:  * Logical device assignments (CON:, LST:, etc)
11:  * CCB port address
12:
13:  Do you want to also permanently save the Auto Key strings? [Y/N] Y<enter>
14:                                         Done
15:
16:  Do you want to also permanently save the Key Translation table? [Y/N] Y<enter>
17:                                         Done
18:
19:             Press <ENTER> when ready
```

Figure 4.18 MENU Display for Freezing the I/O Configuration (option FR)

**4.18 Option code: GD****Purpose:** To create a new data diskette and copy files to it.**Programs used:** FORMAT, FASTCOPY, PIP

This option allows you to create a new data diskette (ie. no system on the diskette) and copy files to it. Figure 4.19 shows the console display for using this option. It first asks you whether you intend to copy files to the new diskette (Line 4.19-5). If you indicate that you do not wish to copy files to the diskette, it will inform you that you should be using the ND option and will give you a chance to go directly to ND.

Assuming that you want to copy files to the diskette, MENU will then ask you to specify the source and destination drives (Lines 4.19-8 and 4.19-9). The destination drive is the drive on which the new data diskette will be created. MENU then asks if you want to format the diskette before transferring files to it (Line 4.19-10). If you answer affirmatively, as shown, MENU will then ask for the density at which the new diskette is to be formatted (Line 4.19-11).

```
1:                                     Generate a Data Disk
2:
3:                                     Press <ESC> to quit
4:
5: Do you want to copy files to the new disk? [Y/N] Y<enter>
6:
7:
8: Copy from disk (SOURCE) [A-P, 0-3] A<enter>
9: Copy to disk (DESTINATION) [A-P, 0-3] B<enter>
10: Initialize (format) the DESTINATION disk [Y/N] Y<enter>
11: Density [1 for single, 2 for double] 2<enter>
12:
13: Do you want to copy
14: A. All .COM files
15: B. All files
16:
17: Selection? [A/B] A<enter>
18:
19:
20: Mount disk to format on drive B: and press <ENTER> when ready to start :<enter>
21: Formatting Single-sided disk in drive B: at Double-density.
22: Format complete.
23: Checking disk on drive B:
24: Checking complete.
25:
26: COPYING -
27: VERIFY.COM
28:   :
29:   :
30:   :
31:
32:
33: >>> DISK OK <<<      Do you want to do another disk? [Y/N] N<enter>
```

Figure 4.19 MENU Display for Generating a Data Diskette (option GD)

MENU then gives you two choices for which files to copy. After entering your selection (Line 4.19-17) MENU begins the generation process. First it loads the FORMAT utility (if requested) to format and check the new diskette (Lines 4.19-20 to 4.19-24). See Section 8.17 for information on the FORMAT utility and the error messages it might report.

MENU then uses either PIP or FASTCOPY to copy the specified files from the source disk to the destination diskette. If you selected option A (Line 419-14) MENU will use PIP to copy all ".COM" files from the current user. If you selected option B (Line 419-15) FASTCOPY will be used to copy all files from all user numbers. See Section 8.22 for further information on PIP and Section 8.16 for further information on FASTCOPY. In either case the names of the files transferred will be displayed on the console as they are copied.

Finally MENU reports whether or not all went well and gives you the option of generating another data diskette (Line 419-33). An affirmative response will cause the GD option to be restarted while a negative response will return you to the selection menu.

**4.19 Option code: GS****Purpose:** To generate a new working system diskette.**Programs used:** FORMAT, FASTCOPY, CLONE, PIP

This option allows you to create a new working system diskette and copy files to it. Figure 4.20 shows the console display for using this option. It first asks you to specify the source and destination drives (Lines 4.20-5 and 4.20-6). The source drive is the drive on which a working system diskette is (or will be) mounted and the destination drive is the drive on which the new system diskette will be created. MENU then asks if you want to format the diskette before transferring the system and files to it (Line 4.20-7).

**Note:** The new diskette must be double density. If it is not, you must format it or switch to one which is already formatted double density.

```
1:                                     Generate a System Disk
2:
3:                                     Press <ESC> to quit
4:
5: Copy from disk (SOURCE) [A-P, 0-3] A<enter>
6: Copy to disk (DESTINATION) [A-P, 0-3] B<enter>
7: Initialize (format) the DESTINATION disk [Y/N] Y<enter>
8:
9: Do you want to copy
10: A. Only System (.PNT) files
11: B. System (.PNT) and all .COM files
12: C. All files
13:
14: Selection? [A/B/C] C<enter>
15:
16:
17: Mount disk to format on drive B: and press <ENTER> when ready to start :<enter>
18: Formatting Single-sided disk in drive B: at Double-density.
19:     Format complete.
20:     Checking disk on drive B:
21:     Checking complete.
22:
23:
24: Mount source disk on drive A:
25: Mount destination disk on drive B:
26:     Press <ENTER> when ready : <enter>
27: Copying from drive A: to drive B: at Double Density.
28:     System tracks copied.
29:
30: Mount source disk on drive A:
31: Mount destination disk on drive B:
32:     Press <ENTER> when ready : <enter>
33: Copying from drive A: to drive B:
34:     Reading BIOSPARAM.PNT R/W, DIR, user= 0
35:     .
36:     .
37:     .
38:     .
39:
40: >>> DISK OK <<<      Do you want to do another disk? [Y/N] N<enter>
```

Figure 4.20 MENU Display for Generating a System Diskette (option GS)

MENU then gives you three choices for which files to copy. In this instance, "system files" refer to the disk files that must be present on the diskette in order

for the system to be loaded. All of these files have the file type "PNT". After entering your selection (Line 4.20-14) MENU begins the generation process. First it loads the FORMAT utility (if requested) to format and check the new diskette (Lines 4.20-17 to 4.20-21). See Section 8.17 for information on the FORMAT utility and the error messages it might report.

Menu next uses CLONE to copy the system tracks from the source drive to the new diskette (Lines 4.20-24 to 4.20-28). See Section 8.7 for information on the CLONE utility program and the error messages it might produce.

MENU then uses either PIP or FASTCOPY to copy the specified files from the source disk to the destination diskette. If you selected options A or B (Lines 4.20-10 and 4.20-11) MENU will use PIP to copy files from the current user (usually user 0). If you selected option C (Line 4.20-12) FASTCOPY will be used to copy all files from all user numbers. In either case the names of the files transferred will be displayed on the console as they are copied. See Section 8.22 for further information on PIP and Section 8.16 for further information on FASTCOPY.

Finally MENU reports whether or not all went well and gives you the option of generating another system diskette (Line 4.20-40). An affirmative response will cause the GS option to be restarted while a negative response will return you to the selection menu.

**4.20 Option code: HZ**

**Purpose:** To set up the console display for different power line frequencies.

**Programs used:** none

The console display is normally set up to be used with a 60 hertz power line frequency. If you have 50 hertz power, use this option to configure the display for it. Figure 4.21 shows the console display when this option is selected. On Line 4.21-11 you should enter either 50 or 60 depending on your power line frequency. The change will take effect immediately on the console display and the BIOSPARM.PNT file will be updated so that the display is initialized properly when the system is loaded (RESET).

**Note:** You do not need to use this option if you have 60 hertz power since the system defaults to it.

```
1:           Set up system for 50 or 60 Hz line frequency
2:
3:
4:
5: Type
6: This          To Perform This Function
7:
8: 50 Set up system for 50 Hz line frequency
9: 60 Set up system for 60 Hz line frequency
10:
11: Selection? 50<enter>
12:
13:
14:
15:           Press <ESC> to quit
```

Figure 4.21 MENU Display for Configuring Power Line Frequency (option HZ)

**4.21 Option code: KT**

**Purpose:** To save the key translations currently defined for the KEYXLATE module.

**Programs used:** none

If you elect to include the KEYXLATE module with the system when it is loaded, you will be able to translate 16 key codes generated by the keyboard into 16 other codes. These translations may be set by the utility program KXEDIT (see Section 8.18) or by a program (see Section 7.3). Normally, the translations are lost if the system is reloaded (RESET). This option allows you to save the current translations in the BIOSPARM.PNT file so that they still will be in effect the next time the system is reloaded (RESET).

Figure 4.22 shows the console display as the translations are being saved. After you press <enter> MENU will return to the selection menu.

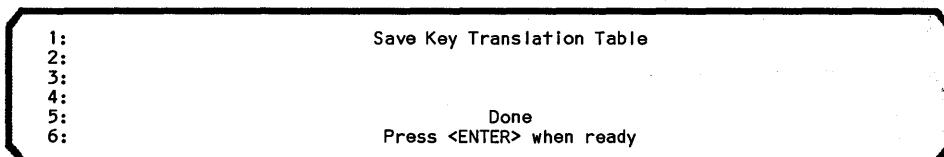


Figure 4.22 MENU Display for Saving Key Translation Table (option KT)

**4.22 Option code: LA**

**Purpose:** To set the last address in memory that may be used by P&T CP/M 2.

**Programs used:** none

In some cases it is desirable to reserve some memory above the operating system for special usage. This memory may be used for loading special programs or as a communication area between programs. The system functions as if this memory did not exist, hence it is completely free for other uses. This option allows you to specify the last address in memory that may be used by the system. After using this option and reloading (RESETting) the system, you are guaranteed that the system will use no memory addresses above the one you specified.

Figure 4.23 shows the console display while using this option. It first gives a little explanation about what it does (Lines 4.23-5 to 4.23-7). It then reports the current last address available to P&T CP/M 2 (Line 4.23-9). Next it asks you if the value just displayed is correct (Line 4.23-11). If you respond negatively, it then asks you for the last address that you want CP/M to use (Line 4.23-12).

```
1:           Set last memory address that CP/M may use
2:           *****
3:
4:
5:   You may reserve some space in memory above CP/M. This is done by setting
6:   the last memory address that CP/M may use. If you do not want to reserve
7:   any space above CP/M, enter the value FFFF.
8:
9:   The last memory address that CP/M may now use is FFFF
10:
11:  Is this correct? [Y/N] N<enter>
12:  What is the last address that CP/M may use? (in hex) [BFFF.. FFFF] FEFF<enter>
13:
14:
15:
16:           Press <ESC> to quit
```

Figure 4.23 MENU Display for Setting Last Address Used by CP/M (option LA)

Note that you should enter the last address that you want P&T CP/M 2 to use, not the first address that you want free. In this example, memory from FF00 to FFFF is reserved above the system. Note that the address entered is FEFF, the last address available to the system.

After you enter the last address to be used, MENU redisplays lines 4.23-5 to 4.23-11, reporting the new last address. You may change the last address as many times as you wish. When you respond with an affirmative answer to the question on Line 4.23-11, the new last address value will be written to the BIOSPARM.PNT file and will be effective the next time the system is loaded (RESET). If you press <esc> in response to the question, any changes that you have made to the last address will be ignored and MENU will return to the selection menu.

**4.23 Option code: MT**

**Purpose:** To test the computer's memory and report any bad memory cells.

**Programs used:** none

This option tests all banks of memory (including the console display memory) that are installed in the computer. Half banks such as are found in the Model 12 and with the Radio Shack hard disk are detected and properly reported. The console display for the test is shown in Figure 4.24. In this case the test was run on a computer that has a Radio Shack hard disk installed which provides a half bank (16 Kbytes) of memory accessible as banks E and F. If memory errors are detected, the half bank in which the error was found is reported as well as the bits which are in error. This information may help in describing the problem to your Radio Shack service center.

Memory Test		
Press <ESC> to quit		
4:	Bank	Comments
5:		
6:	CRT	OK
7:	MAIN	OK
8:	2	Not Installed
9:	3	Not Installed
10:	4	Not Installed
11:	5	Not Installed
12:	6	Not Installed
13:	7	Not Installed
14:	8	Not Installed
15:	9	Not Installed
16:	A	Not Installed
17:	B	Not Installed
18:	C	Not Installed
19:	D	Not Installed
20:	E	C000-FFFF Not Installed
21:	F	C000-FFFF Not Installed
22:		

Figure 4.24 MENU Display for Memory Test (option MT)

**4.24 Option code: ND**

**Purpose:** To initialize (format) a new diskette for use.  
**Programs used:** FORMAT

This option formats a diskette so that it can be used on the system. It allows you to format the diskette at either single or double density. Figure 4.25 shows the console display for this option.

On Line 4.25-7 it asks you for the drive on which the format operation is to take place. On Line 4.25-8 it asks you for the density at which the diskette is to be formatted. After gathering this information, MENU loads and executes the FORMAT utility program to format and check the diskette (Lines 4.25-10 to 4.25-15). See Section 8.17 for further information on FORMAT and the error messages it might display.

After the format operation is complete, MENU tells you that the diskette is initialized and gives you a chance to format another one (Line 4.25-18). An affirmative response will cause the ND option to be started over while a negative response will return you to the selection menu.

```
1:                                     FORMAT
2:                                     Prepare a disk for use
3:                                     NOTE: Any information on the disk will be DESTROYED!!!
4:
5:                                     Press <ESC> to quit
6:
7: 7: Drive to be formatted (initialized) [A-P, 0-3] B<enter>
8: 8: Density [1 for single, 2 for double] 2<enter>
9:
10: 10: Mount disk to format on drive B: and press <ENTER> when ready to start :<enter>
11:
12: 12: Formatting Single-sided disk in drive B: at Double-density.
13:     Format complete.
14:     Checking disk on drive B:
15:     Checking complete.
16:
17:
18: 18: Disk Initialized      Do you want to do another disk? [Y/N] N<enter>
```

Figure 4.25 MENU Display for Formatting a New Diskette (option ND)

**4.25 Option code: PP**

Purpose: To set the parallel printer port parameters.  
Programs used: MENUOLY1

After selecting this option MENU will show you a display like that shown in Figure 4.26. You may change any of the listed parameters by entering the number of the parameter. After you enter a number, you will be presented with a variety of choices for that parameter. You may change as many parameters as you wish. Press <esc> when you are finished making changes and wish to return to the selection menu.

```
1:                               Parallel Port Parameters
2:                               Press <ESC> to quit
3:
4:
5:
6:                               Centronics Options
7:   1. Suppress extra line feeds: disabled
8:   2. Emulate form feed:        disabled
9:   3. Do auto form feed:      disabled
10:  4. Paper length (in lines): 66
11:  5. Number of lines before auto form feed: 60
12:
13:
14:
15:
16: Enter number of item to change:
```

Figure 4.26 MENU Display for Changing Parallel Port Parameters (option PP)

The functions available for the parallel printer port are discussed in Chapter 13. They may also be changed by the SETMISC utility program (see Section 8.25).

**4.26 Option code: SD**

**Purpose:** To set the system date.

**Programs used:** SETDATE

This option allows you to set the system date. If you do not have a P&T CCB board installed in your computer, you must set the system date if you have programs that use it. This option loads and executes the SETDATE utility program. See Section 8.24 for details on using SETDATE.

**4.27 Option code: SM**

**Purpose:** To select the modules that are to be loaded with the system.

**Programs used:** MODSEL

This option allows you to select the modules that are to be included in the system when it is loaded. You may use this option to include or remove utility modules such as AUTOKEY and KEYXLATE or to change I/O drive modules. It loads and executes the MODSEL utility program. See Section 6.3 for details on using MODSEL.

**Note:** Any changes you make with MODSEL will not take effect until the next time the system is loaded (RESET).

**4.28 Option code: SP**

**Purpose:** To change the serial port configuration.  
**Programs used:** SETUP

This option allows you to configure the serial ports for your particular use. The features of the serial port driver are discussed in Chapter 12. This option loads and executes the SETUP utility program. See Section 8.27 for details on using SETUP.

**4.29 Option code: ST**

**Purpose:** To set the system time.

**Programs used:** SETTIME

This option allows you to set the system time. If you do not have a P&T CCB board installed in your computer, you must set the system time if you have programs that use it. This option loads and executes the SETTIME utility program. See Section 8.26 for details on using SETTIME.

4.30 Option code: SY

Purpose: To synchronize the system date and time to a P&T CCB board.

Programs used: none

Since the time of day clock in the system loses about 5 seconds an hour, it may be desirable to resynchronize it to a P&T CCB board (if one is installed) periodically. This menu option performs this function. It also sets the system date according to the CCB board. This can be useful when running programs that change the system time or date.

Figure 4.27 shows the console display as the synchronization takes place. After you press <enter> at the end of the process, MENU will return to the selection menu.

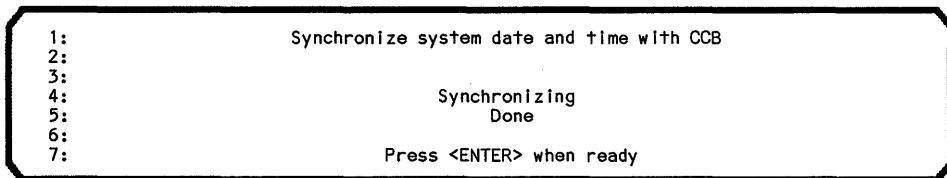


Figure 4.27 MENU Display for Synchronizing Date & Time to a CCB (option SY)

4.31 Option code: **TE**

Purpose: **To test a diskette for bad areas (destroys data on the diskette).**

Programs used: **DISKTEST**

This option tests a diskette by writing several data patterns on it and checking that they can be read back correctly. This test, by its nature, destroys any data already recorded on the diskette, so it should be used only on diskettes that contain no useful information. Figure 4.28 shows the console display for this option.

On Line 4.28-6, MENU asks for the drive on which the test is to be conducted. It then loads and executes the DISKTEST utility program to perform the test (Lines 4.28-11 to 4.28-16). After the test is complete, MENU gives you a chance to test another diskette (Line 4.28-19). An affirmative response at this point will cause the TE option to be restarted while a negative response will return you to the selection menu.

```
1:           Test a Disk for Bad Areas
2:           NOTE: Any Information on the disk will be DESTROYED!!!
3:
4:           Press <ESC> to quit
5:
6:           Drive to be Tested [A-P, 0-3] B<enter>
7:
8:
9:
10:
11:          Mount disk to test on drive B: and press <ENTER> when ready to start :<enter>
12:          Testing Double-density disk in drive B:
13:             Beginning pass with pattern = 00
14:             Beginning pass with pattern = FF
15:             Beginning pass with pattern = DBB6
16:             Beginning pass with pattern = DBB6
17:             Testing complete.
18:
19:
20:          >>> DISK OK <<<      Do you want to do another disk? [Y/N] N<enter>
```

Figure 4.28 MENU Display for Testing (destructive) a Disk (option TE)

## 5.1 Memory Usage

P&T CP/M 2 divides memory into four parts, each having a specific purpose. These four parts are illustrated in Figure 5.1 and are described as follows:

The System Parameter Area (SPA) occupies the region of memory from location 0000h through 0FFh. In this area are various system entry points, buffers, and other information. In general, programs are not allowed to make use of the memory in the System Parameter Area other than to access the buffers and pointers that are kept there. This region of memory is sometimes called the Base Page.

The Transient Program Area (TPA) begins at location 100h in memory and extends upward. The upper limit of the TPA varies with the total amount of memory allotted to CP/M. This is the region of memory in which most programs run.

The Console Command Processor (CCP) begins at the end of the TPA and extends upward in memory for 800h bytes (2K bytes). This section of memory is occupied by the program which interacts with the user when the system is at the command level. It is responsible for carrying out certain operations, such as loading and executing a program, in response to input from the user.

The final region of memory is referred to as FDOS. It consists of the Basic Disk Operating System (BDOS) and the Basic Input Output Subsystem (BIOS). The BDOS is the part of the system which maintains the diskette data structure and supervises input/output operations. BIOS is responsible for the customization of CP/M to a specific set of hardware. P&T CP/M 2 integrates a custom BIOS and special utility programs with the standard CP/M 2 operating system.

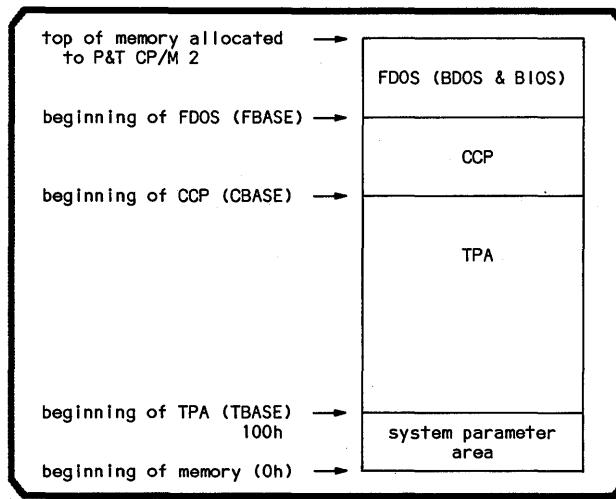


Figure 5.1 Memory Map of P&T CP/M 2

For additional information on memory usage by CP/M 2 see pp 89, 90, 144 of CP/M Operating System Manual.

## 5.2 Disk Storage

Mass storage of information on most CP/M systems is accomplished by use of magnetic disks. These storage units may be flexible diskette drives (called floppy diskette drives), or they may be hard disk units. The basic concepts of disk storage are the same for all types of disks.

A magnetic disk consists of a thin disc of some sort of base material coated with a compound that can be easily magnetized. The base material can be plastic, as with floppy diskettes, or metal, as with hard disks. The coating resembles the type found on audio magnetic tape. Information is stored in the form of minute magnetized areas in the coating of the disk.

In order to provide rapid access to the stored information, the coated disk is spun and a magnetic recording head is moved back and forth over its surface. The head can be stopped at various locations for reading and writing information. While the head is in a given location, it reads and/or writes along a circular path on the disk surface, called a track. The tracks on a disk can be pictured as a series of concentric circular recordings on the disk's surface. Since the recording head can be moved from track to track quite rapidly, all of the information recorded on the disk can be accessed in a short period of time.

Since a large amount of information can be stored on one track, it may be inconvenient to deal with it all at one time. For this reason, a track is divided into smaller portions, called sectors, in one of several possible ways. Some disk drives have internal mechanisms for creating the divisions, while others require instructions from the computer.

The floppy diskette drives used with the TRS-80 Model II/12/16 do not force a sector structure on the diskette. Through an operation called "formatting the diskette", the computer determines the sector size to be used and writes an entire set of sectors to each track on the disk. For all subsequent operations on the diskette, the sector size determined by the formatting process will be used. The diskette can be erased and re-formatted with a different sector size if desired.

The track recorded on a disk is often called a **physical track**, since it is determined by the physical attributes of the disk drive and what is recorded on the disk. Similarly, the sectors on a track are referred to as **physical sectors**. In other words, physical tracks and sectors are the actual patterns of recording on the disk and the term **physical drive** refers to the actual disk drive.

In order to treat all disk storage in a uniform way, CP/M operates in terms of logical drives. In contrast to physical drives, logical drives are what CP/M considers as one storage unit. A logical drive may be identical to a physical drive (this is usually the case for floppy diskette drives), or it might encompass only a portion of a disk drive (this often occurs on hard disk drives). In CP/M, the logical drives are named with letters of the alphabet starting with "A" and continuing through "P" for a total of 16 logical drives. Up to 8 Mbytes (8,388,608 bytes) can be stored on each logical drive.

The correlation between logical and physical drives is set when the CP/M system is adapted to a particular type of hardware. For the TRS-80 Model II/12/16 with floppy disks only, the built-in floppy drive is assigned to logical drive A, while the expansion drives (if any) are assigned to logical drives B, C, and D. (With hard disk versions of P&T CP/M 2 these assignments are under your control.) Under TRSDOS, the floppy drives are referred to as drives 0, 1, 2, and 3. The correspondence

between the TRSDOS drive numbers and the CP/M logical drive assignments is shown in Figure 5.2.

TRSDOS Drive Number	P&T CP/M 2 Drive Letter
0	A
1	B
2	C
3	D

Figure 5.2 TRSDOS to CP/M Logical Drive Correspondence

Once again for the sake of uniformity, CP/M reads and writes to a disk in standard chunks of information called **logical sectors** (sometimes referred to as **logical records**). The basic element of information used by CP/M is the byte (8 bits). 128 bytes are grouped together to form a logical sector, which is the smallest amount of data that can be input from or output to a disk. Since access to a disk must be in multiples of these sectors, a diskette is sometimes referred to as a record or block oriented device.

Whenever CP/M reads or writes to disk storage, it ALWAYS treats 128 bytes of information as one logical sector. If the size of the physical sector on the actual disk drive is not 128 bytes, the BIOS disk I/O routines of CP/M make the necessary correlation between logical and physical sectors. For example, a P&T CP/M 2 double density disk stores 512 bytes in each physical sector, so BIOS assigns four logical sectors to each physical sector. In general, a physical sector on a disk may accommodate one or several logical sectors.

The sum total of the logical sectors allowed on a given track is, quite naturally, called a logical track. The size of a logical track is not rigidly fixed by CP/M. When CP/M is customized to a certain set of hardware, the logical track size for each disk drive on the system may be fixed at a convenient value. Usually the logical track is chosen to correspond to the physical track on the disk drive.

The logical drive, logical track, and logical sector numbers can be thought of as forming a unique address for each logical sector of disk storage available in the computer system.

P&T CP/M 2 divides logical drives into three regions, each of which has a distinct usage. A number of tracks at the beginning of a disk may be reserved for system usage and are typically called the system tracks or the reserved tracks. The first two tracks (0,1) of a diskette are reserved for the operating system. These tracks are often referred to as the system area of the diskette. They contain the program which is the operating system and various other parameters. Under normal circumstances you should never need or attempt direct access to these tracks.

The second region of a disk is where the directory is maintained. The directory is used to store information about the files on the disk. This area takes up part of track 2 of the diskette, and its size depends on the number of directory entries allowed in the system.

The third region starts immediately after the end of the directory and continues to the end of the disk. This area is where file data is actually stored. Often, the directory and data regions are lumped together and referred to as the data area.

Figure 5.3 gives a schematic representation of the regions into which a floppy diskette is divided.

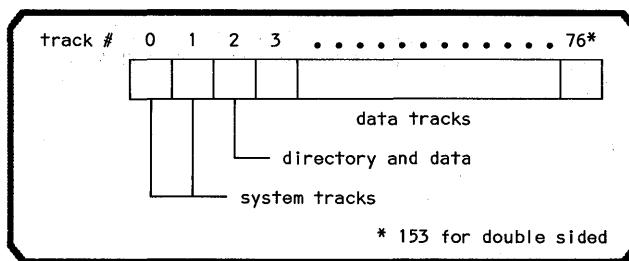


Figure 5.3 Floppy Disk Track Assignments

### 5.3 Files and Disk Storage

Information is stored on the diskettes in logical entities called files. A file is a collection of data that has been grouped together because of some common tie between the data elements. For example, the characters which make up this manual are grouped together in a file which was generated by a text editor and subsequently manipulated by a text processing program. Another example of a file could be the collection of characters which makes up the text of a computer program. Yet another example of a file is the collection of bytes of data which make up the binary image of a program that can be loaded into the computer's memory and executed.

A text file consists of a group of coded representations of numeric, alphabetic, and special characters. The most common character code in use today, and the one used by P&T CP/M 2, is the American Standard Code for Information Interchange (ASCII). The numerical codes for the characters defined by ASCII are given in Appendix A. In ASCII files, only 7 bits out of each byte carry the encoding of the character. The eighth bit is sometimes used for error detection but is frequently merely ignored. Since the characters in a text file are represented by ASCII, text files are often referred to as ASCII files.

A binary file contains information which utilizes all eight bits of each byte and does not usually represent character data. The information may be actual machine instructions of a program or the binary representation of numerical data. With binary files all of the bits of each byte are significant.

Each file on a P&T CP/M 2 diskette is given a unique name for purposes of referring to the file. CP/M takes care of all the details of finding room on the diskette and allocating that room to the file. This allows programs to refer to files by name alone. Typically, a program will have no information about the actual location of the file's information on the diskette, yet it still is able to access any part of the file via CP/M.

A file name consists of up to 11 characters divided into a 1 to 8 character primary name and a 0 to 3 character secondary name (sometimes called an extension or type). A file name is written with the primary and secondary names separated by a period (.) as shown in Figure 5.4. In Figure 5.4, "p" stands for a character of the primary file name and "s" stands for a character of the secondary file name.



Figure 5.4 Example of a P&amp;T CP/M 2 File Name

The primary and secondary names may be made up of any combination of characters, with the exception of the special characters shown in Figure 5.5.

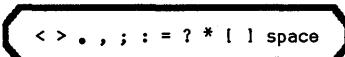


Figure 5.5 Characters not Allowed in a File Name

The secondary name frequently denotes the type of information contained in the file. Many programs which are designed to operate on a specific type of information will require a specific secondary name as an indication that the information in the file is of the proper type. Consequently the secondary name is also referred to as the "file type". Some common file types are given in Figure 5.6.

5

extension	common usage
ASM	assembly language source code file
BAS	Basic language source code file
COM	executable object code file
HEX	object code file in hex format - usually output from an assembler
LIB	library of macros or re-locatable modules
OVL	object code that is used as an overlay
PAS	Pascal language source code file
PRN	listing file from assemblers and compilers
REL	relocatable object code file - usually output from an assembler or compiler
SRC	source code file - used by several languages
SUB	file containing commands to use with SUBMIT
TXT	text file

Figure 5.6 Common File Types

If all the characters of the primary and secondary names of a file are given in referring to it, the reference is said to be unique, since only one file on a disk can be indicated by the name. Several examples of unique file names are given in Figure 5.7.



Figure 5.7 Examples of Unique File Names

In the remainder of this manual, the abbreviation "unqfn" will represent a unique file name. When "unqfn" appears in an example or reference, it means that some unique file name is to be inserted in its place. (Note that the CP/M Operating System Manual and some other documents refer to a unique file name as an unambiguous file name and abbreviate it as "ufn".)

It is often desirable to refer to a number of files at one time and P&T CP/M 2 provides this capability by allowing wildcard file names under certain circumstances. Two special characters are used to construct wildcard file names, "?" and "\*".

"?" will match any one character (including the absence of a character) in a file name. As an example, suppose a program has created a series of work files named WORK1.DAT, WORK2.DAT, WORK3.DAT, WORK4.DAT, and WORK5.DAT. Having run the

program, the user now wishes to erase all the work files. Instead of naming individual files in the erase command, the user could specify the wildcard filename WORK?.DAT, which would match all the workfiles mentioned above. (Note that a file with the name WORK.DAT will also be matched, which may cause a problem if you don't want it erased.) WORK?.DAT will not match WORK10.DAT because only one "?" was used in the wildcard file name, leaving one character of the file name (the 0) unmatched in the wildcard file name.

"\*\*" will match any number of characters in a file name. If "\*" is used as the primary part of a wildcard file name, it will match all possible primary names from 0 to 8 characters. If "\*" is used as the secondary part of a wildcard file name, it will match all possible secondary names from 0 to 3 characters. For example, \*.ASM will match all files with the secondary name ASM. Likewise, TESTPROG.\* will match all files with the primary name TESTPROG and \*\* will match all files on a disk.

"\*\*" may be used in conjunction with other characters to specify wildcard file names. For example, if a program created work files with the names WORK1.DAT, WORK5.DAT, WORK10.DAT, WORK15.DAT, and WORK100.DAT they could all be referred to by the wildcard file name WORK\*.DAT. The "\*" will match any 4 characters following WORK in the primary file name (in this case 1, 5, 10, 15, 100). If a "\*" is used with other characters as just illustrated, it must appear as the last character in the primary or secondary name. Note: an embedded "\*" in the primary or secondary part of a file name is not allowed.

**SPECIAL WARNING:** If you use an embedded "\*\*", all of the remaining characters of the primary or secondary name will be ignored. As an example, if you give the command ERA \*WORK.DAT, all files with a secondary name of "DAT" will be erased. The characters "WORK" will be ignored.

Figure 5.8 gives several examples of wildcard file names.

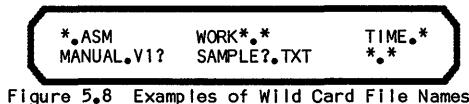


Figure 5.8 Examples of Wild Card File Names

In the remainder of this manual, a wildcard file name will be abbreviated as "wcfn". When this abbreviation appears in an example, it should be replaced by an actual wildcard file name. Note that a unique file name is a subset of wildcard file names; hence a unqfn can be used wherever a wcfn is indicated. (Note that the CP/M Operating System Manual and other documents refer to a wildcard file name as an ambiguous file name and abbreviate it as "afn".)

P&T CP/M 2 allows access to as many as 16 disk drives. In order to specify which drive is to be searched for the desired file, the file name may be prefixed by the drive name character (A through P) followed by a colon (:) as shown in Figure 5.9. Note that when a drive designation is included in a file name, the entire name may have up to 14 characters (2 for the drive designation, 8 for the primary name, 1 for the period, and 3 for the secondary file name).

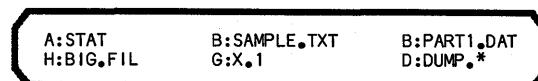


Figure 5.9 Examples of File Names with Drive Designation

**Note to programmers:** While it is possible to have file names with lower case letters in them, it is not usually a good practice. The CCP always translates file names to upper case, making it impossible to refer to a file whose name contains lower case letters while in the CCP. The most common way in which file names containing lower case letters are created on a disk is by some sort of high level language. Some high level languages do not convert lower case letters in a file name to upper case before creating the file. (While it is not required for a program to make this conversion, it is customary to do so to avoid the previously mentioned problem.)

If a file name is specified with lower case letters in these languages, the file will be created on the disk with lower case letters in its name. Since it is not possible to erase a file with lower case letters in its name directly from the CCP (remember that it translates file names to upper case), deletion of the file must use a more indirect method. The easiest way of deleting such a file is to use the facilities of the language that created it in the first place. Most languages provide the ability to delete files, hence a short program should enable you to erase the file.

## 5.4 Disk Allocation Blocks

P&T CP/M 2 uses a method of dynamic disk storage allocation which eliminates the need for periodic repacking of a disk to reclaim unused space. The disk is broken up into a number of sections called allocation blocks. When a file needs room on the disk, an unused allocation block is assigned to it. As one allocation block is filled, another is assigned to the file, and so on. These blocks need not be in the same area of the disk and may, in fact, be scattered all over the disk. Also, any created file which has been written to will consume at least the amount of storage required by one allocation block: 2K bytes on a double density diskette and 1K byte on a single density diskette.

When a file is deleted, the allocation blocks it occupied are returned to the pool of free blocks and can be reused as necessary by other files. After using a diskette for some period of time, the storage allocated to a disk file may become scattered over different parts of the disk. This condition is called fragmentation and can cause substantially slower disk performance.

When files become severely fragmented, the read/write head of the disk drive must be moved frequently while accessing the file. As an extreme example, consider a file whose first block is on track 3, second block is on track 76, third block is on track 4, fourth block is on track 75, etc. In order to access the file sequentially, the head would have to move nearly the full width of the diskette (taking as much as .7 seconds) between blocks. Fortunately, fragmentation rarely becomes this severe.

If a diskette becomes severely fragmented, the solution is to copy the files of interest to an empty diskette using a routine that handles each file through CP/M, such as PIP or FASTCOPY. When CP/M is putting a file on an empty diskette, it allocates sequential allocation blocks, so a newly copied diskette will have no fragmentation. Do not use an image copy routine such as CLONE. This type of copy routine bypasses CP/M to create a direct image of the original diskette, complete with any fragmentation that existed on the original.

CP/M maintains a directory of the files on a diskette which associates the file name with the file's actual storage locations on the diskette. Figure 5.10 shows the number of directory entries, allocation block size, and number of logical sectors per track for each of the supported diskette formats.

Format	Directory Entries	Allocation Block Size	Logical Sectors per Track
single density	64	1 Kb	26
double density	128	2 Kb	64
double sided	192	2 Kb	64

Figure 5.10 Number of Directory Entries for Disk Formats

Each directory entry has a limited amount of space. If the numbers of the allocation blocks occupied by a large file cannot be stored in one directory entry, another entry with the same name is created to provide more room. For this reason, one file may take up several directory entries, even though the file appears only once when the directory is listed. Obviously, it follows that the number of different files which may be stored on a disk can vary according to their sizes.

## 5.5 Random Access Files

In order to maintain compatibility with previous versions of CP/M, disk files are considered to be broken up into 16K byte segments called logical extents. (16K bytes is the maximum amount of storage for 1 directory entry on a single density disk.) Compatibility with earlier versions of CP/M exists for all file sizes when the files are accessed sequentially, but is limited to files of 512K bytes if the random access technique of the earlier CP/M versions is used. Application programs requiring random access to files larger than 512K bytes must be converted to use the random access capability of P&T CP/M 2.

In particular, if you are using CBASIC programs which make random access to disk files, you must use the appropriate CRUN file to run the programs. If the digit following the decimal point in the version number of the CRUN file is "3" (e.g. 2.38), that file uses the version 2 random access technique. Using this CRUN file will assure you of being able to randomly access files of any size. If you are not sure of the version number of your CRUN file, merely type "CRUN<enter>" as a command to the system. The CRUN package will first display its version number and then give you an error message indicating that you did not specify a program to run. If the digit following the decimal point in the version number is not "3" you will not be able to use random access files larger than 512K bytes. CBASIC is typically distributed with several CRUN files. The one you need is probably on your CBASIC master diskette. If it is not, you should contact the dealer from whom you purchased CBASIC.

## 5.6 File Attributes

P&T CP/M 2 allows files to be tagged with two types of attributes. This special information is used by CP/M to modify the way the files are handled in some instances.

The first attribute determines if the file is "read only" (R/O) or "read write" (R/W). A "read only" file can be read but not modified. This attribute is useful for protecting files from accidental modification. When a file is first created using the normal system calls, it is a R/W file. It may later be changed to R/O using the utility routine STAT or by using a standard system call. See BDOS function 30 on p.106 of CP/M Operating System Manual for further information.

The second attribute determines whether or not the file's name will appear in the directory listing for a disk. If a file is tagged as a SYS file, it will not appear in

the directory listing. If a file is tagged as a DIR file, it will appear in the directory listing. This attribute is useful for suppressing the listing of common utility routines on the disk, making the directory listing less cluttered. A file created in the normal manner will be tagged as a DIR file and will appear in the directory listing. It may later be changed to a SYS file using the STAT utility or a standard system call (BDOS function 30).

## 5.7 Non-Disk I/O

Micro-computer systems are likely to have a variety of non-disk input/output (I/O) devices connected to them. For example, most systems include some sort of console and a printer. Such actual devices are often called "physical I/O devices." However, this term can also apply to the interface within the computer that is used to connect peripheral equipment. Both definitions are used by P&T CP/M 2.

P&T CP/M 2 implements four non-disk physical I/O devices. The video display and keyboard are grouped together under the physical device name CRT. The RS-232 ports are given the physical device names SIOA and SIOB, and the name CENTRON refers to the parallel port printer. You may have noticed that these physical device names do not agree with those of generalized CP/M 2. Standard CP/M's physical device names do not correspond very well to the actual devices available on the TRS-80 Model II/12/16, so appropriate changes were made in P&T CP/M 2. Although you may never need to use the general physical device names, Figure 5.11 shows the correspondence between the general names and the P&T CP/M 2 names. The general names are only defined for certain logical devices hence they are grouped by logical device in Figure 5.11.

general name	P&T CP/M 2 name	function
for the CON: logical device:		
TTY:	SIOA	input from / output to serial port A
CRT:	CRT	input from keyboard / output to video
BAT:	CENTRON	input from keyboard / output to Centronics
UC1:	SIOB	input from / output to serial port B
for the RDR: logical device:		
PTR:	URDR1	input only from user supplied reader
UR1:	URDR2	input only from user supplied reader
UR2:	SIOB	input only from serial port B
for the PUN: logical device:		
PTP:	UPUN1	output only to user supplied punch routine
UP1:	UPUN2	output only to user supplied punch routine
UP2:	SIOB	output only to serial port B
for the LST: logical device:		
LPT:	CENTRON	output only to Centronics port
UL1:	SIOB	output only to serial port B

Figure 5.11 CP/M 2 General Physical I/O Device Names

CP/M also allows programs to deal with I/O devices in a standard way. Any program input or output uses one of four logical I/O devices. To the program, a logical device simply represents a place to which or from which data is transferred. The program needs no knowledge of the actual hardware connected to the logical I/O device. In fact, the hardware can be changed without affecting the program operation. Each of the four logical I/O devices has a specific function as shown in Figure 5.12.

CON: The system console. All user interaction is done via the system console.  
LST: The system printer. This is the standard hardcopy device of the system.  
RDR: An input only device which is sometimes associated with a paper tape reader.  
PUN: An output only device which is sometimes associated with a paper tape punch.

Figure 5.12 Logical I/O Devices

P&T CP/M 2 normally associates the CON: logical device with the computer's video display and keyboard. CON: is the only logical I/O device in the system which has both input and output capabilities. LST: and PUN: are both "output only" devices, while RDR: is an "input only" device.

Each logical device has standard system calls which perform I/O to and/or from the device. Most high level languages already make use of the CON: and LST: devices for console I/O and hardcopy output. It is less common for languages to utilize RDR: and PUN: since many systems do not have these devices available. Most users will use only CON: and LST:.

In order for a program to communicate with a physical I/O device, some association must be made between the logical and physical I/O devices. At any instant, each logical device is assigned one and only one physical device. A physical device can be assigned to more than one logical device, however. The assignments can be changed directly by programs or through the ASSIGN, SETUP and STAT utilities supplied with the system. These changes are possible because the assignments are controlled by what is called the IOBYTE. Discussion of the IOBYTE and changing I/O device assignments is covered in Chapter 14 of this manual. Figure 5.13 shows the assignments that can be made with ASSIGN and SETUP.

CON:	SIOA	CRT	CENTRON	SIOB
LST:	SIOA	CRT	CENTRON	SIOB
PUN:	SIOA	UPUN1	UPUN2	SIOB
RDR:	SIOA	URDR1	URDR2	SIOB

Figure 5.13 Assignments That Can be Made With ASSIGN and SETUP

The assignment of physical devices to logical devices acts like a switching mechanism in the flow of data. As an example, suppose that two printers are connected to a system. One, a high speed printer, is connected to the parallel printer port. The other, a high quality printer, is connected to serial port B. A program always sends characters to be printed to the LST: logical device, but any of the physical devices can, in turn, be assigned to the LST:. For data to reach the high speed printer, LST: would have to be assigned to the CENTRON physical device assigned to it. At another time, the program could use the high quality printer by assigning LST: to the SIOB physical device. Figures 5.14a through 5.14d show schematically how two logical devices (in this case, CON: and LST:) can communicate with a variety of physical devices.

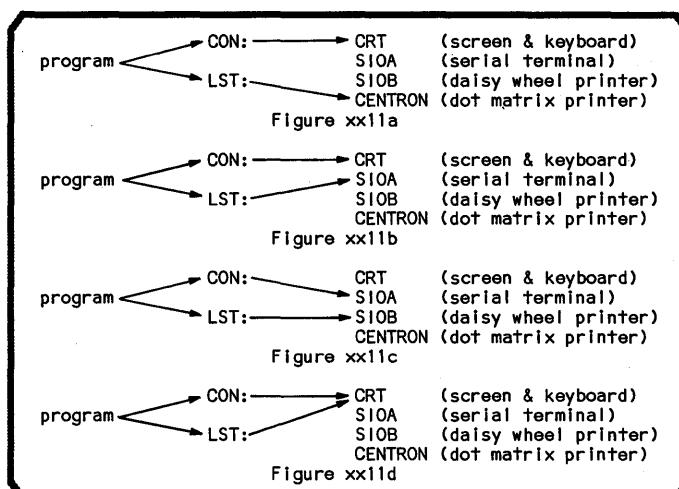


Figure 5.14 Illustration of Logical to Physical Device Assignment

## 5.8 Command Structure

When P&T CP/M 2 begins operation, you will be at the command level (sometimes called the command mode) of the system. (An exception occurs when a program is automatically executed - see MENU option AE, Section 44 for details.) At the command level, you interact with the portion of the system called the Console Command Processor (CCP). It is the CCP which allows you to direct the system manually from the console keyboard. P&T CP/M 2 indicates it is at the command level by issuing a special prompt, consisting of the current default drive letter followed by a greater than sign (>), to the console. This prompt is illustrated in Figure 5.15.



Figure 5.15 Command Level Prompt

The **current default drive** has a variety of other names. Most commonly, it may also be called the "current drive", "default drive", or "logged on drive". In any case, it is the logical disk drive that is used for any disk operation when no particular logical drive is specified. This is true when the system is in the command mode and when programs are running. As shown in Figure 5.16, the current default drive can be changed from the command mode merely by typing the letter of the desired drive, a colon, and <enter>. The current default drive may also be changed by programs through standard system calls.



Figure 5.16 Changing the Current Default Drive from A to B

There are two types of commands that can be given to the CCP: those that use the internal functions of the CCP, and those that execute programs. The internal functions are contained entirely within the CCP, so there is no need to access a disk unless the function requires it (for example, listing a directory or erasing a

file). Functions that are too complex to be an integral part of the CCP are supplied as utility programs which must be loaded into the TPA (Transient Program Area of memory) and executed. Since they remain in memory only while they are executed, they belong to a group known as "transient programs." Transient programs retain control of the computer until they have finished their task and then return control to the CCP.

The six functions that are built into the CCP are summarized in Figure 5.17. Additional information about the CCP built-in commands may be found on pp.6-8 of CP/M Operating System Manual.

DIR	Lists files in a disk's directory
ERA	Erases (deletes) files from a disk
REN	Renames a file
SAVE	Saves a portion of the TPA as a disk file
TYPE	Displays a disk file on the console
USER	Changes the current user number

Figure 5.17 Summary of the Built-In Commands of the CCP

## 5.9 The DIR Command

This command displays all the files in the directory which are tagged with the DIR attribute (see Section 5.2) and which are stored under the current user number on the console. The file names are shown four to a line and in the order of their appearance in the disk directory. Several variations of the DIR command are allowed, as shown in Figure 5.18.

DIR<enter>	lists the names of all files on the current drive
DIR unqfn<enter>	lists the name of the file specified if it exists on the current disk. This command is useful for finding out if a specific file is on a disk.
DIR wcfn<enter>	lists the names of all files matching the wildcard file name on the current disk
DIR x:<enter>	lists the names of all files on drive x

Figure 5.18 Examples of Using the DIR Command

The DIR command may specify the name of a file or set of files to list. If a unique file name is given on the command line, the disk directory will be searched for a file by that name. If one is found, the name will be displayed on the console. The command may also specify the drive on which to search. A wild card file name may be used with the DIR command. In this case, all file names that match the wild card name will be listed. Any character typed at the console (except <hold> and <ctl-S>) will cause the directory listing to be aborted and the system to return to the command level. <hold> and <ctl-S> will cause the display to stop until another key is pressed. They are especially useful if the directory is so long that it cannot be displayed on a single screen.

Figure 5.19 shows the console dialog for a listing of all files on a disk that have the extension "ASM".

```
A>DIR *.ASM<enter>
A: DUMP      ASM : SETTIME  ASM : TIME    ASM
A>
```

Figure 5.19 Console Display for DIR \*.ASM Command

## Possible Error Messages

### NO FILE

This response is given if no files are found on the specified disk and user number that match the one given in the DIR command. If no file name was given with the DIR command, this message indicates that the directory of the disk is empty. If files do exist that match the file specification but have the SYS attribute set (so they will not be displayed in a directory listing), neither the NO FILE message nor any file names will be displayed.

## 5.10 The ERA Command

The ERA command is used to ERASE one or more files from a disk. ERA has the general form shown in Figure 5.20.

ERA wcfn<enter>

Figure 5.20 General Form of the ERA Command

All files under the current user number that match the wildcard file name will be removed from the disk. As always, the term "wildcard file name" includes unique file names as a special case, so individual files can also be erased by the ERA command. The erasure will occur on the current default drive unless another logical drive is specified in the command. Sample ERA commands are given in Figure 5.21.

**CAUTION:** ERA performs the erasure immediately after you press <enter> at the end of the command line. You do not get a chance to change your mind after pressing <enter>. Check your command line carefully when using ERA.

You should also keep in mind that ERA will erase files with the SYS attribute set as well as DIR files. (Remember SYS files do not show up in a directory display.) Some people develop the habit of using DIR to see what files will be affected by an ERA command with a wildcard file name. If you do this and then use ERA with the same wildcard file name, you may end up erasing some SYS files (if they happen to match the wildcard) without even knowing it.

ERA TAX.DAT<enter>	removes the file TAX.DAT from the current disk
ERA TAX.*<enter>	removes all files with the primary name TAX from the current disk
ERA *.*<enter>	removes all files from the current disk. As a precaution, the system prompts the user with All Files (Y/N)? Any response other than Y<enter> will abort the process and return the user to the command level of the CCP.
ERA B:TAX.DAT<enter>	removes the file named TAX.DAT from the diskette on drive B.

Figure 5.21 Examples of Using the ERA Command

## Possible Error Messages

### NO FILE

This message indicates that no files could be found matching the given file specification.

### BDOS Err on x: File R/O

This message indicates that you have attempted to erase a file which has its read only attribute set. The "x" in the message will be replaced with the single letter name of the drive on which the error occurred (A - P). After displaying this message, the system will wait for input from the console. The first character input from the console will cancel the ERA command and cause the system to perform a warm boot. Essentially, a warm boot involves the reloading of the CCP and BDOS from the diskette and a return to command level (this may require a system diskette on the system drive - see Section 5.16).

## 5.11 The REN Command

The REN command is used to RENAME disk files. It has the general form shown in Figure 5.22.

```
RENAME unqfn1=unqfn2<enter>
```

Figure 5.22 General Form of the REN Command

If the file with name "unqfn2" is found on the disk under the current user number, its name will be changed to "unqfn1". The file to be renamed must be on the current default drive unless another drive is specified in the command. In this case, the letter of the drive and a colon are placed before "unqfn1". Both file names may be prefixed with logical drive indications, but only if they specify the same drive. If the drive prefix is placed only before the second name, it will be interpreted as part of the file name itself. Note that both file names must be unique; REN cannot rename a group of files by means of wildcard file names. Some specific examples of REN commands are given in Figure 5.23.

```
REN OLDTAX.DAT=TAX.DAT<enter>      renames the file TAX.DAT to OLDTAX.DAT  
                                         on the current drive.  
REN B:REDUC.BAK=REDUC.BAS<enter>      renames the file REDUC.BAS to REDUC.BAK  
                                         on drive B.
```

Figure 5.23 Examples of Using the REN Command

## Possible Error Messages

### NO FILE

This message indicates that no file can be found to match "unqfn2".

### FILE EXISTS

This message indicates that a file already exists on the specified disk with the name given by unqfn1. REN will take no further action. If you wish to perform the rename operation, you must first erase the file that has the name you want to use.

### BDOS Err on x: File R/O

This message indicates that the file referred to by "unqfn2" is set to "read only" status. The "x" in the message will be replaced with the single letter name of the drive on which the error occurred (A - P). The system will then wait for input from the console. The first character input will cancel the REN command and cause the system to perform a warm boot.

## 5.12 The SAVE Command

The SAVE command is used to save a portion of the transient program area (TPA) as a file on the disk. This command is most used by experienced programmers. You may find that you have very little use for it in your normal system usage. If you don't understand SAVE very well it will not affect your ability to use the system. The general form of the SAVE command is given in Figure 5.24.

SAVE n unqfn<enter>

Figure 5.24 General Form of the SAVE Command

The "n" in the SAVE command specifies the number of "pages" (256 byte blocks) of memory from the beginning of the TPA that is to be saved in the file specified by "unqfn". "n" in the command must be a decimal number. As an example, suppose that you wish to save the section of memory from 100h (the start of TPA) to A75h. The amount of memory that you want to save is 975h bytes (A75h - 100h = 975h).

Since you must save memory in entire page increments, you would then save 10 pages. (There are 100h (256) bytes per "page".) Note that you will actually save more than the desired section. In this example, all of the memory between 100h to AFFh would be saved. WARNING: if a file exists with the "unqfn" specified in the SAVE command, it will be replaced by the new file.

The "unqfn" may specify a logical drive on which to save the file with the standard drive specification. A specific example of the SAVE command is given in Figure XX21.

SAVE 40 B:SAMPLE.COM      saves memory from 100H to 28FFH in a file  
                              named SAMPLE.COM on drive B.

Figure 5.25 Example of Using the SAVE Command

## Possible Error Messages

### NO SPACE

The message is displayed if there is no room in the disk directory to create a new file or there is insufficient space on the disk to save the requested number of pages. In most cases, no action will be taken and you should repeat the command with another disk. If storage space on the disk is exhausted in the middle of the SAVE operation, a portion of the created file may be left on the disk (since this is an incomplete file, you should probably erase it to prevent future confusion).

### BDOS Err on x: File R/O

This message indicates that a file with the specified "unqfn" already exists and is set to "read only" status. The "x" in the message will be replaced by the single letter name of the drive on which the error occurred (A - P). The system will then wait for input from the console. The first character input will cancel the SAVE command and cause the system to perform a warm boot. The warm boot operation will not affect the contents of the TPA so you may use the SAVE command again with another file name.

### 5.13 The TYPE Command

The contents of a file may be displayed on the console using the TYPE command. Its general form is given in Figure 5.26.

```
TYPE unqfn<enter>
```

Figure 5.26 General Form of the TYPE Command

The "unqfn" must be specified, and it must be listed under the current user number. The command may also specify the logical drive from which to access the file by specifying the drive letter at the beginning of the file name in the normal manner. As a file is being displayed, any character typed at the console keyboard (except <hold> and <ctl-S>) will abort the operation and cause a return to the command level of the system. <hold> and <ctl-S> will cause the display to stop until another key is pressed. They are especially useful if the file is so long that it cannot be displayed on a single screen.

It is assumed that the file contains ASCII characters. Tab characters will be expanded with spaces, with a tab stop at every eighth column. NOTE: no checks are made to insure that the information is in ASCII characters. Typing a non-ASCII file can produce very strange results.

Figure 5.27 gives an example of using the TYPE command.

```
A>TYPE TIME.ASM<enter>
      title  'Utility routine to report current time 4/27/80'
      page   62

      maclib z80s

      ; This routine reads the current value of the system time of day clock
      ; and reports it on the console.

      spcfun  equ    40h          ;entry point for special system functions
      bufprn  equ    9           ;CPM function number for line output
      .
      .
      .
      timsg: db     cr,lf,cr,lf,'The current system time is '
      hmsg:  db     '00:'
      mmsg:  db     '00:'
      smsg:  db     '00',cr,lf,es

      end    0100h

A>
```

Figure 5.27 Example of Using the TYPE Command

### Possible Error Messages

#### filename?

If a file cannot be found that matches the specified unqfn, TYPE will display the file name followed by a question mark on the console and then return to the system command level. You can use the DIR command to display the names of the files on the disk.

## 5.14 The USER Command

P&T CP/M 2 supports a division of the directory into as many as 16 sections, with each section assigned a number. When the disk system is set to a certain user number, access to files stored in other sections of the directory is not allowed (except in special circumstances). This partitioning of the directory may be used to provide some separation between groups of programs that have different functions.

When the system is first started, it comes up as USER 0. With the system set to USER 0, the directory entries generated are compatible with earlier versions of CP/M. If you are reading or writing a diskette that must be compatible with earlier versions of CP/M, work at single density only and always remain at USER 0.

The USER command changes the current user number. Its form is shown in Figure 5.28. The "n" is a number from 0 to 15 which specifies the desired user number.

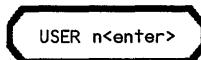


Figure 5.28 General Form of the USER Command

**NOTE OF CAUTION:** under earlier versions of CP/M, ERA \*\* was a convenient way to remove all files from a diskette, leaving it essentially blank. In P&T CP/M 2, this command will erase only the files under the current user number. In order to clear the disk completely, you must repeat the ERA \*\* command under every user number which has file storage. A faster way of obtaining a blank diskette is to run the format program on it. However, this has the undesirable side effect of erasing the system tracks as well as the data tracks.

### Possible Error Messages

nn?

If the parameter given in the USER command is not an integral number between 0 and 15, the console will display the parameter followed by a question mark, and the system will return to the command level.

## 5.15 Transient Commands

As already noted, some of the more complex functions of CP/M cannot be included in the built-in commands of the CCP, so they are supplied as utility programs (also known as "transient programs"). The utility programs supplied with P&T CP/M 2 are described in Chapter 8. In addition, the utility programs that are a standard part of CP/M 2 are also described in the CP/M Operating System Manual.

The commands used to execute utility programs (or any programs) are no different than those which execute the built-in functions of the CCP. All executable programs are stored on the disk with a secondary file name of "COM". From the command level, the user types the primary name of the file, any parameters the program may require, and <enter>. The CCP then loads the program into memory, passes the parameters to it, and starts execution. Figure 5.29 gives an example of the command line which would be entered to edit a file named TEXT.1 with the ED text editor.

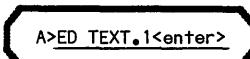


Figure 5.29 Example of Executing a Transient Program

Strictly speaking, the user is entering a transient command when any program, such as BASIC or a compiler, is executed. The command is the primary name of the file in which the program is stored. Thus, you can change the names of the transient commands at will merely by changing the file names. For example, if you want to run the text editor by typing EDITOR instead of ED, simply rename the file ED.COM as EDITOR.COM.

## 5.16 Boots: Warm and Cold

Following a RESET, it is assumed that the machine is in an unknown state, so the entire system is loaded into memory. This operation is often referred to as a **cold boot**. During a system load, the CCP, BDOS, and BIOS are loaded from the system diskette and certain initialization procedures are performed (such as setting up the serial ports, reading the date and time from the CCB board if one is installed, etc.).

The Console Command Processor (CCP) has no function while a transient program is running so transient programs are allowed to make use of the memory normally occupied by the CCP. If a transient program uses this area, the CCP must be reloaded when the program terminates. The function that reloads the CCP and BDOS is called a **warm boot**. It is the normal way of terminating any program. A warm boot also occurs when you press the <break> key at the command level of the system. The warm boot function also performs other operations like resetting the disk system so that you can change disks.

In some systems, the warm boot operation reloads the CCP and BDOS from the diskette mounted on the system drive. In others, they are stored in additional memory making a diskette unnecessary for a warm boot. When loading the CCP and BDOS from a diskette, it is necessary that they have been set up to run at the memory location to which they are loaded. (See Section 9.4 for further information about this matter).

You may switch system diskettes between warm boots as long as the CCP and BDOS on the new diskette are set up the same as on the disk from which the system was

loaded. If they are set up differently the error message shown in Figure 5.30 will be displayed. If you get this message, you have mounted a system diskette that is incompatible with the system that is running in the computer. You may either RESET the computer to reload (cold boot) the system from that disk or remount the system disk from which the system was last loaded and press <enter>.

Mismatched system disk - please try another.

Figure 5.30 Error Message for Incompatible System Diskette

## 5.17 Console Input

If a <ctl-P> is typed from the command level of CP/M, all subsequent output to the console will also be sent to the LST: logical device. This will continue until a warm boot occurs or <ctl-P> is typed again.

CP/M has a buffered input capability which is used by many programs and the CCP. When buffered input is performed, all characters entered from the console are held in a buffer until <enter> is pressed. Once you press <enter>, the entire line is acted upon. Before pressing <enter>, you have some editing capability available to you. The editing functions are accessed by means of control characters, as described in Figure 5.31.

<ctl-C>	(<break>) will cause a warm boot if it is the first character on a line.
<ctl-E>	sends a carriage return and line feed to the console but does not put them in the line. This feature is used to break up long lines for increased readability.
<ctl-H>	(<back space>) causes the cursor to back up one position and erase the character at that position.
<ctl-J>	terminates input of line.
<ctl-M>	(<enter>) terminates input of line.
<ctl-R>	move to a new line and retype the current contents of the line.
<ctl-U>	delete the contents of the current line and move to a new line.
<ctl-X>	delete the contents of the current line and erase all characters back to the beginning of the line.
rubout	(<ctl-minus>) deletes and echos the last character entered.

Figure 5.31 Line Editing Control Characters

When information is being displayed on the console, the display can be stopped by typing <ctl-S> or <hold> on the console keyboard. The display will continue when any additional character is typed. Two notes of caution are needed here: 1. Some programs bypass the part of CP/M that performs this function, so it will not work when those programs are running. 2. If, while information is being displayed on the console, you type any key other than <ctl-S>, the <ctl-S> function will not work until an input from the console has been performed (to read the character that was typed).

## 6.1 Introduction

P&T CP/M 2 revision M is designed with a modular concept. Various different functions are contained in different modules. For example, the serial port driver routines are in their own module. Similarly the parallel port driver routine is in a separate module. Although a maximum of 31 modules can be included in the system, it is very rare that more than 10 are used.

The advantage of modularity is that you can configure the system much more closely to your needs. If you do not need to use the serial ports, you need not include the serial port module in your system. The MODSEL utility program provides (see Section 6.3) an easy means of selecting the modules which are to be included in the system.

One result of the ability to select only certain modules to be included in the system is that you can have several different "versions" of the system each with a different combination of modules. In general, when you load the system from a working system diskette from one "version" of the system, the system will not be able to warm boot from diskettes containing other "versions". We suggest that you color-code the different versions so that you will not get them mixed up.

With P&T CP/M 2 the CCP and BDOS are stored on the system tracks of a system diskette in a relocatable form. The various modules that make up the remainder of the system are stored in a file called BIOSMODS.PNT. A second file, named BIOSPARM.PNT, contains the names of all the modules to be included when the system is loaded (RESET) as well as other information about the initial state of the system.

When the system is loaded, it first reads the BIOSPARM.PNT file to determine which modules are to be included. It then performs 4 operations. 1) It determines how much memory space is required for all of the modules to be included in the system and for any free memory space that is to be left above the system. 2) From this information it determines where in memory the CCP and BDOS are to reside. The copy of the CCP and BDOS on the system tracks is then set up to run at the appropriate memory address and loaded into memory. Once the CCP and BDOS on the system tracks have been set up, they can simply be reloaded when a warm boot occurs. 3) The system then loads the requested modules from BIOSMODS.PNT and locates them at the appropriate places in memory. 4) Finally any required initialization is performed before the system begins operation.

The BIOSPARM.PNT and BIOSMODS.PNT files are not needed for a warm boot operation. The only requirement for a warm boot is that an appropriate copy of the BDOS and CCP reside on the system tracks of the diskette mounted in the system boot drive (note: even this is not required in some systems such as a Radio Shack hard disk system). Thus you can change diskettes in the system boot drive as long as the CCP and BDOS appear on the system tracks and have been set up appropriately.

You can insure that the copy of CCP and BDOS on the system tracks of a diskette are properly set up in two ways. 1) You can make several working system diskettes and copy the same BIOSPARM.PNT and BIOSMODS.PNT files to all of them. Then you should load (RESET) the system at least once from each of them. By loading the system causes the CCP and BDOS to be set up and, since the same modules are

being loaded (you insure this by copying the BIOSPARM.PNT and BIOSMODS.PNT files), the CCP and BDOS will be set up identically on each diskette. 2) You can CLONE the system tracks from your working system diskette to other diskettes that are likely to be mounted on the system drive. This insures that the system tracks (and hence the copies of CCP and BDOS) are identical. Note that in this case it is not necessary to have the BIOSPARM.PNT and BIOSMODS.PNT files on each diskette. They are only required on the diskette from which the system is initially loaded.

Should you attempt to warm boot from a diskette that does not have a properly set up copy of CCP and BDOS, an error message will be given and you will have the opportunity to mount another system disk to complete the warm boot. Note that the system diskette from which the system was last loaded (RESET) will always work for a warm boot.

## 6.2 Types of Modules

There are several different types of modules available in P&T CP/M 2. When selecting modules to be included in the system there are some types that are required to be present and some that are optional. The following are the general categories of modules:

### **Core Modules (one required)**

These modules make up the basic core of the I/O system. There may be one or more core modules available to choose from.

### **Warm Boot Modules (one required)**

These modules control the action taken during a warm boot. Different warm boot modules may be available to take advantage of differing hardware configurations.

### **CRT Modules (one required)**

The CRT modules take care of output to the console display of the computer.

### **Disk Parameter Modules (one required)**

These modules contain the parameters that define how disk storage is to be allocated. For floppy based systems, there are 4 predefined disk parameter modules; one each for 1, 2, 3, and 4 floppy drive systems. For hard disk drive systems additional predefined disk parameter modules are available which cover typical hard disk configurations. Hard disk systems include the HDCONFIG utility program which allows you to edit and create disk parameter modules.

### **Hard Disk Driver Modules (optional)**

These modules contain the necessary interface software to allow the system to use a hard disk drive in addition to the floppy drives. If your system has a hard disk driver module, you may elect whether or not to use it. If you do not use it, the system will load as a standard floppy disk system.

### **Serial Port Modules (optional)**

The serial port modules provide access to the serial ports on the computer. One may be included if you require access to the serial ports.

### **Parallel Printer Port Modules (optional)**

These modules provide access to the parallel printer port. Different parallel printer port modules may be supplied for use with different parallel printers. A parallel printer port module may be included if you require access to the port.

### Utility Modules (optional)

These modules provide useful functions such as programmable function keys, key translation, terminal emulation, etc. You may select as many of them as you wish.

## 6.3 Selecting Modules

The MODSEL utility program is provided to allow you to easily select the modules you want included in the system. It is described here rather than in Chapter 8 with the rest of the utility programs.

Utility name: **MODSEL**

Purpose: **To select the modules that are to be included when P&T CP/M 2 is loaded.**

### General Description

The MODSEL program allows you to select the modules you want included when P&T CP/M 2 is loaded into memory. MODSEL shows you the module names that are currently listed in the BIOSPARM.PNT file for inclusion in the system and allows you change, add to, or delete them. MODSEL also insures that you include all required modules.

Note: MODSEL does not alter the operating system that is in memory. The selections you make will included in the system the next time it is loaded (RESET).

Each module in the library of modules includes a description of the module. As you are choosing the modules to be included in the system, MODSEL allows you to view the description as an aid to deciding which modules to include.

MODSEL requires that the BIOSPARM.PNT and BIOSMODS.PNT files be on the same logical drive (as they would be on a bootable system diskette). It does not require that they be on a particular drive. This allows you to alter a different system diskette than the one from which you loaded the system. For example, even though the system diskette from which you loaded the system may still be mounted on drive A you may select the modules to be loaded on another system diskette merely by mounting it on another drive.

### Using MODSEL

Any time that MODSEL asks a question that requires a yes/no answer, the responses "Y", "y", and "1" will be accepted as an affirmative answer and the responses "N", "n", and "0" will be accepted as a negative answer.

The command line to execute MODSEL is shown in Figure 6.1.

A>MODSEL<enter>

Figure 6.1 Command Line to Execute Module Selector

The first thing that MODSEL does is ask you on which drive it is to find the module file, BIOSMODS.PNT (Figure 6.2). Note that the file BIOSPARM.PNT must also be on the same disk, as is typically the case. On Line 6.2-13, drive D is specified as the drive on which to find these files. You should insure that the

appropriate disk is mounted and ready since MODSEL will immediately access the drive after you answer this question.

```
1:          P&T CP/M 2 Module Selection Program - Ver 1.xx
2:
3:
4:
5:
6:
7:
8:
9:
10:
11:
12:
13:          Enter drive on which to find BIOSMODS.PNT: D<enter>
14:
15:
16:
17:
18:
19:
20:
21:
22:
23:
24:
```

Figure 6.2 Entering Drive on Which to Find Module Library

After you have specified the drive, MODSEL will read the existing BIOSPARM.PNT file to determine which modules are already selected for inclusion when the system is loaded. As it does this, it displays the message "reading old system parameter file..." on the console. Once it has done this, MODSEL will begin to present you with modules to choose from.

The names of the available modules are displayed for you by type grouping. If there is only one available module for one of the required types, MODSEL will automatically select it for you without any action on your part. If there is only one available module for an optional type, MODSEL will present it to you so that you can decide whether it included or not.

After specifying the drive (Figure 6.2) the next display you see might look like Figure 6.3. This is the general form of the display for selecting one or more modules. On Line 6.3-9, the type of module under consideration is shown. On lines 6.3-11 to 6.3-21, the names of the available modules in this category are displayed. On lines 6.3-22 to 6.3-24 the actions you may take are described. The name(s) of any modules currently included from this group will be highlighted in reverse video and the cursor will be positioned at the first of the names displayed.

You may move the cursor from name to name by using the cursor arrow keys on the keyboard. If you press the <hold> key, the module at which the cursor is presently positioned will be selected. As you select the module, its name will be highlighted in reverse video. For categories from which only one module may be selected MODSEL will not allow you to select more than one module. If a module is already selected (highlighted) when you press <hold>, it will be deselected (returned to normal) before the new one is selected (highlighted).

If you press the <esc> key, the module at which the cursor is positioned will be deselected. As this is done the reverse video will be removed from the module's name. You may select and deselect modules as many times as you want. The selection does not take effect until you press the <enter> key. Upon pressing

<enter> any modules whose names are highlighted will be included in the system and MODSEL will move on to the next category of module. Note that no changes are made to the BIOSPARM.PNT file until the very end of MODSEL. You will be given a final chance to review your choices and accept them before any changes are made to the file.

```
1: P&T CP/M 2 Module Selection Program - Ver 1.xx
2:
3:
4:
5:
6:
7: Disk Table Modules
8:
9:
10:
11: 1FLOPPY 2FLOPPY 3FLOPPY 4FLOPPY
12:
13:
14:
15:
16:
17:
18:
19:
20:
21:
22: <hold>.....select module      <f1>....show description of module
23: <esc>.....remove module       <f2>....display modules already included
24: <arrow keys>..move cursor    <enter>..include selected modules
```

Figure 6.3 Selecting a Disk Parameter Module

If you press the <f2> key, MODSEL will display all the names of all the modules which have been included in the system up to this point. This list does not include any names from the category currently displayed since selections from it do not take effect until you press <enter>. Figure 6.4 shows an example of a display that might be given when <f2> is pressed while the display of Figure 6.3 is on the console. Note that STDCORE1 and STDCRT1 have already been selected; in many cases MODSEL will automatically make these selections (when there is only one available module in a required category). All module names shown in this display will be highlighted since they have all been included.

```
1:          P&T CP/M 2 Module Selection Program - Ver 1.xx
2:
3:
4:
5:
6:
7:          Modules Already Included
8:
9:
10:
11:         STDCORE1 STDCRT1
12:
13:
14:
15:
16:
17:
18:
19:
20:
21:
22:
23:
24:          (press <enter> to continue)
```

Figure 6.4 Displaying the Modules Already Included

If you press <f1> the description of the module at which the cursor is positioned will be displayed on the screen. Figure 6.5 shows the display that results from pressing <f1> when the cursor is positioned at 2FLOPPY in Figure 6.3. You can easily scan the descriptions of the various modules available in a given category by moving the cursor to each module name and pressing <f1>. This feature is designed to give you a quick reminder of the function of each module.

```
1:          P&T CP/M 2 Module Selection Program - Ver 1.xx
2:
3:
4:
5:
6:
7:
8:          Disk Parameter Module for two physical floppy drives.
9:
10:         Allows single density, double density, and double sided double
11:             density on all drives.
12:
13:         Physical drive 0 is assigned to logical drive A
14:         Physical drive 1 is assigned to logical drive B
15:
16:
17:
18:
19:
20:
21:
22:
23:
24:          (press <enter> to continue)
```

Figure 6.5 Displaying a Module Description

Figure 6.6 shows the display of the utility module names. Note that, since these modules are not required in the system, "(optional)" appears on Line 6.6-10 directly under the description of the module category. This will occur for all optional categories. You may select as many modules from the utility module category as you wish.

```
1: P&T CP/M 2 Module Selection Program - Ver 1.xx
2:
3:
4:
5:
6:
7:           Utility Modules
8:           (optional)
9:
10:
11:          AUTOKEY SCRNDUMP KEYXLATE
12:
13:
14:
15:
16:
17:
18:
19:
20:
21:
22: <hold>.....select module      <f1>....show description of module
23: <esc>.....remove module      <f2>...display modules already included
24: <arrow keys>..move cursor    <enter>.include selected modules
```

Figure 6.6 Selecting Utility Modules

After you have gone through all the categories of modules, MODSEL will present you with a display of all their names as shown in Figure 6.7. You are then asked (Line 6.7-23) to verify that these are the modules you want included in the system. A negative response on Line 6.7-23 will cause MODSEL to start over. An affirmative response will cause the BIOSPARM.PNT file to be updated to include the modules displayed. As the file is updated, MODSEL will display the message "writing new system parameter file..." on the console. After the file has been updated, MODSEL will clear the screen and return you to the command level of the system.

```
1: P&T CP/M 2 Module Selection Program - Ver 1.xx
2:
3:
4:
5:
6:
7:           These modules will be included in system:
8:
9:
10:
11:          STDCORE1 STDCRT1 2FLOPPY STDWB1 S10     PPSTD     AUTOKEY
12:
13:
14:
15:
16:
17:
18:
19:
20:
21:
22:
23:           Is this correct? Y<enter>
24:
```

Figure 6.7 Reviewing Modules Selected

## Possible Error Messages

### Do you really want to quit (Y/N)?

This is not really an error message. It appears at the bottom of the screen when you press the <break> key. If you pressed it accidentally you can return to the program by giving an negative response. If you do want to abort the program and return immediately to the operating system, enter an affirmative response.

### Not a valid response, please re-enter

This message is given either by itself or in conjunction with another error message. It indicates that the response you just made is not valid for some reason (for example, answering a yes/no question with Q). You should enter another response.

### No more room for modules in system

#### [module name] not included in system

This message is given if you try to select more than the maximum permissible number (31) of modules for inclusion in the system. As long as all the module being selected when this message is given is not a required module, the program will continue to allow you to remove other modules or take other action.

### No room for essential module, aborting operation

If you are selecting a required module and it would exceed the maximum permissible number of modules (31), this message is displayed. The program will immediately return to the operating system if this error occurs.

### \*\*\*\*\* Not enough memory for module selection \*\*\*\*\*

This message indicates that there is not enough memory available for workspace for the program. You should not encounter this error unless you have reserved a large amount of memory above the operating system causing the Transient Program Area to be smaller than 48 Kbytes.

### Enter a drive letter A to P

This message is displayed if you have specified an illegal logical drive letter.

### Drive X does not exist

This message is displayed if you enter a logical drive letter for a drive that is not defined on the system. The "X" will be replaced by the letter you specified.

### Cannot open BIOSMODS.PNT

This message indicates that the program was unable to open the file containing the library of modules available to you. This typically is caused by specifying the wrong disk drive on which to look for the file.

### A [module type] is required

This message is given if you attempt to bypass the selection of a module type that is required for the system. You will not be allowed to proceed until you have selected one of the modules whose names are presented on the console display.

### Unable to find compatible disk parameter module

#### Enter correct driver type for your system

This message is given if you attempt to select a hard disk driver module for which there are no compatible disk parameter modules in the library. You should never get this message. If you do, your module library file has probably been

damaged in some way. You should get a fresh copy of the file from your master diskette.

**Error opening BIOSPARM.PNT to read old parameters**

This message is displayed if the program cannot successfully open the parameter file in which the names of the modules to include in the system are kept. It typically results from specifying the wrong drive on which to find the file. The program will return to the system immediately after this message.

**Error closing BIOSPARM.PNT**

This message indicates that a disk error occurred when the BIOS parameter file was being closed. This error is very rare and may indicate a hardware problem. You should try making another working system diskette and using MODSEL on it.

**Module library damaged...essential module missing**

This message indicates that, while checking the module library, MODSEL found that one or more of the required module types were missing. This usually indicates that something has damaged the library file.

6

**A [module type]**

**is required and one was not included**

If MODSEL finds, during its final checking, that a required type of module was not selected for inclusion in the system, it displays this message. You should never see this message. If you do, it may indicate a problem in the module library or the MODSEL program.

## NOTES

## 7.1 Introduction

The utility modules provide for various useful functions to be loaded with the system. These functions typically take up too much memory to be included in the core since not everyone will want to use them. If you do choose to include a utility module, it essentially becomes part of the system.

The following utility modules are available:

- AUTOKEY Provides 5 programmable function keys. The keys may be programmed directly from the keyboard at any time or by sending a special string to the system console.
- KEYXLATE Allows you to translate any 16 keys on the console keyboard to other characters.
- SCRNDUMP Allows you to print the contents of the console display on the system printer by pressing <ctl-> on the system keyboard.
- ADM3A Makes the console display/keyboard look like an ADM 3A terminal.

## 7.2 AUTOKEY Module

The AUTOKEY module provides you with five programmable function keys. Each of the keys can be programmed to generate a string of up to 79 characters with a maximum of 120 characters for all strings. Any characters (including control characters) may be included in a programmed string. The keys may be programmed at any time directly from the console keyboard or by a program using an escape sequence. In addition, the string programmed into any of the keys can be viewed at any time without affecting the console display.

The five function keys are accessed by pressing <ctl-1>, <ctl-2>, <ctl-3>, <ctl-4>, and <ctl-5>. This allows the function keys to be supported without conflicting with any other control keys. As an example, the <f1> key on the keyboard is identical to <ctl-A>. If <f1> were made into a programmable key, <ctl-A> would no longer be available (it would generate the same string as <f1>).

### Programming Function Keys from the Keyboard

To program a function key from the keyboard first press <ctl-7>. The bottom line of the screen will immediately be cleared and the cursor will be moved to its beginning. You should then press the key (one of <ctl-1> through <ctl-5>) you want to program. If you press any key other than one of <ctl-1> through <ctl-5> the programming operation will be aborted and the console display will be restored to its previous condition.

After pressing <ctl-7> and one of <ctl-1> - <ctl-5>, you should enter the string to be programmed into that key. As you type, the characters will be shown on the bottom line of the screen. Typing <enter> will end the string, assign it to the specified key, and restore the screen to its previous condition. While you are entering the string, <back space> may be used to back up and delete the previous character.

You may include most keys just by typing them. If you wish to use a control key, just type the letter while holding down the <ctrl> key. When you type a control key, the character will be shown in reverse video to denote that it is a control character. In order to put <esc>, <enter>, or <back space> into the string, you must type the <esc> key first. For example to put <esc> into the string you must type <esc><esc>; to put <enter> into the string, you must type <esc><enter>. Since <esc>, <enter>, and <back space> are control characters they will be shown in reverse video.

You can program the function keys at any time from the keyboard, even while a program is running. This can be especially useful when editing text that has frequently used words and phrases. You can program the function keys to generate the words and phrases without leaving the editor and you can change them whenever you want without affecting the editor at all.

### Programming Function Keys from a Program

A program may also set up the string for any (or all) of the function keys. To do this, the program must send out an escape sequence of the form shown in Figure 7.1 to the console.

```
<esc> <esc> <key number> string <cr>

If the following special characters are to be included in the string,
they must be sent as the specified escape sequence:

      character          escape sequence
      <cr>              <esc><cr>
      <esc>              <esc><esc>
      <back space>       <esc><back space>

<key number> is replaced by 1, 2, 3, 4, or 5 according to the key
to be programmed.
<cr>           is the ASCII carriage return character. This is the
                  character generated by the <enter> key.
```

Figure 7.1 Escape Sequence to Program a Function Key

One escape sequence is required for each function key to be programmed. Figure 7.2 shows an example in Basic of programming <ctl-1> to "The quick brown fox " and <ctl-2> to "jumps over the lazy dog.<enter>".

```
10 PRINT CHR$(27);CHR$(27);"1The quick brown fox "
20 PRINT CHR$(27);CHR$(27);"2jumps over the lazy dog.";CHR$(27);CHR$(13)
```

Figure 7.2 Example of Programming Function Keys from Basic

When programming the AUTOKEY strings from a program, you may run into a problem if the <tab> character is included in a string. CP/M 2 provides two ways of sending characters to the system console. The first way automatically expands <tab> characters into one or more spaces to move the cursor to the next tab stop. (Tab stops are assumed at every eighth character position.) The second way sends the actual character to the console with no alterations.

Most programming languages use the first method of sending characters to the console (some even expand tabs on their own). Since the <tab> character gets converted into one or more spaces, you are not really sending the characters you expected to the console. Some languages that have this problem provide an

alternate function to send characters directly to the console using the second method. You should check the documentation on the language to see if it has such an alternate function. If you do not have a method of sending characters directly to the keyboard without tab expansion, you can usually send the character code 137 (89h) in place of the <tab> character. Although most programs will accept this code as a <tab> character, it will usually not be expanded.

### Displaying Programmed Strings

You may display the string programmed into any of the five function keys by pressing <ctl-8> followed by one of <ctl-1> through <ctl-5>. After pressing the function key, the bottom line of the console display will be cleared and the string (if any) programmed into that key will be displayed. The next key you press will cause the screen to be restored to its previous condition.

### Editing Programmed Strings

AUTOKEY includes a limited capability to edit a string that has been programmed into a function key. By pressing <ctl-8> followed by one of <ctl-1> through <ctl-5> the string will be displayed. If you then press <ctl-7> you will be able add to the string just as if you were programming it from the keyboard. If you need to change something in a string, you must use <back space> to delete characters back to where you want to make the change. This feature is most useful when you forgot something (like <enter>) at the end of a programmed string.

7

### Making Programmed Strings Permanent

When you program a function key with a string, that string remains in effect until it is reprogrammed or until the system is reloaded (RESET). You may make the strings you have programmed permanent by using system MENU option AK. This option will take the current strings defined for the function keys and store them in the BIOSPARM.PNT file. From that time on the function keys will be automatically loaded with the strings when the system is loaded. You can, of course, still temporarily change the programmed strings as previously described.

### Examples of Use

AUTOKEY can be very useful once you become familiar with it and learn to use it. For example, suppose that you are writing a document with one of the word processing programs that requires commands embedded in the text. You may find that you are repeating two or three commands over and over again. In one popular text processor, two typical command lines used to print numbered paragraphs are \pi-4,sp0\ and \pi0,sp1\. Rather than continuing to type these each time you need them, you could program two of the function keys with the strings. You then could get each string with a single key stroke. Since you can reprogram the functions keys without leaving the text editor it is easy to change the programmed strings when you need different ones.

As another example, consider the case of writing a Pascal/MT+ program. Suppose that the program and the sample data you are testing it with are stored on drive B, the compiler is on drive C, and the text editor is on drive A. You might program four function keys as follows:

```
A:EDIT B:PROG.PAS<enter>
C<enter>MTPLUS B:PROG $TB<enter>
C<enter>LINKMT B:PROG=B:PROG,PASLIB/S<enter>
B:<enter>PROG<enter>
```

The first of these strings invokes the editor to make changes on the program. The second makes drive C the current drive and executes the compiler to compile the program. The third makes drive C the current drive and executes the linker to create an executable file. The fourth string makes drive B the current drive and executes the program for testing. By defining these four function keys, you can save a great deal of typing as you develop the program.

### 7.3 KEYXLATE Module

The KEYXLATE module allows you to translate up to 16 keys on the console keyboard to other characters. This lets you reassign keys for use with various programs. For example, some text editors do not allow you to use the cursor arrow keys on the keyboard. They insist, instead, on using control characters. With KEYXLATE you can translate the arrow keys to the control characters the program expects for cursor movements. The arrow keys will then function just like you would expect them to.

When the KEYXLATE module is first included in the system, it is not set up to perform any translations. You may define the translations you want in one of two ways. You may use the KXEDIT utility program to edit the current translations interactively from the keyboard or you can write a program to set the translations by sending an escape sequence to the console. The KXEDIT utility program is described in Section 8.18.

Most of the non-alphabetic keys in the central typing area of the keyboard generate special codes when they are pressed in conjunction with the <ctrl> key. For example the apostrophe key generates the code 162 (A2h) when it is pressed in conjunction with the <ctrl> key. You can use this fact to translate <ctrl-> into another character (accent grave, for example). The KXEDIT program will allow you to investigate which keys generate these special codes and assign translations to them. Once you know what the translation is, you can have a program assign the translations (see the example below).

#### Setting the Translations from a Program

When the KEYXLATE module is included in the system, a program may set up the translations by sending an escape sequence to the console output. The form of the escape sequence is shown in Figure 7.3.

```
<esc><ctrl-x>(16 original codes)(16 translated codes)
```

The first original code will be translated to the first translated code, the second original code to the second translated code, etc.

All 16 original and translated codes must be sent. If you do not have 16 translations send 128 (80h) for each unused original and translated code.

Figure 7.3 Form of the Escape Sequence to Assign Key Translations

Note that you must send all 16 original codes and all 16 translated codes. If you do not need to use all 16 translations, you should send the character code 128 (80h) for

the original and translated codes you are not using. Figure 7.5 gives an example in Basic of installing the translations shown in Figure 7.4.

from	to
<left arrow> (1Ch)	<ctl-A>
<right arrow> (1Dh)	<ctl-D>
<up arrow> (1Eh)	<ctl-W>
<down arrow> (1Fh)	<ctl-X>
<ctl-`> (A2h)	` (accent grave)
{	{
}	}
{	
}	

Figure 7.4 Translations Assigned by Example in Figure 7.5

```

10 ! Program to assign key translations
20 !
30 ! ----- start with <esc><ctl-X>
40 PRINT CHR$(27);CHR$(24);
50 !
60 ! ----- send out original codes
70 PRINT CHR$(&H1C);CHR$(&H1D);CHR$(&H1E);CHR$(&H1F);
80 PRINT CHR$(&HA2);"[15†";
90 FOR I=1 TO 7 : PRINT CHR$(128); : NEXT      ' fill unused spaces
100 !
110 ! ----- send out translated codes
120 PRINT CHR$(1);CHR$(4);CHR$(23);CHR$(24);
130 PRINT CHR$(96);"$†[1]";
140 FOR I=1 TO 7 : PRINT CHR$(128); : NEXT      ' fill unused spaces

```

Figure 7.5 Example of Assigning the Key Translations from Basic

When setting the key translations from a program, you may run into one or two problems. The first may occur when you want to have the <tab> character involved in the translation. CP/M 2 provides two ways of sending characters to the system console. The first way automatically expands <tab> characters into one or more spaces to move the cursor to the next tab stop. (Tab stops are assumed at every eighth character position.) The second way sends the actual character to the console with no alterations.

Most programming languages use the first method of sending characters to the console (some even expand tabs on their own). Since the <tab> character gets converted into one or more spaces, you are not really sending the characters you expected to the console. Some languages that have this problem provide an alternate function to send characters directly to the console using the second method. You should check the documentation on the language to see if it has such an alternate function. If you do not have a method of sending characters directly to the keyboard without tab expansion, KEYXLATE will recognize the character code 137 (89h) in an assignment string as a <tab> character.

The second potential problem arises if the programming language you use automatically sets the high bit of each character to 0 before sending it to the console. This makes it impossible to send characters whose codes are greater than 127 to the console. Fortunately most languages do not do this. If you are using a language that does perform this way, you will need to find some alternate means of sending codes greater than 127 to the console, should you need to do so.

In addition, if you use KEYXLATE in conjunction with AUTOKEY, you will not be able to translate the keys <ctl-1>, <ctl-2>, <ctl-3>, <ctl-4>, <ctl-5>, <ctl-7>, and <ctl-8>. You can assign translations to these keys if you wish but AUTOKEY will not allow

the translated character through since it uses these keys itself. Similarly, if you use KEYXLATE with SCRNDUMP, you will not be able to translate <ctl=>.

## 7.4 SCRNDUMP Module

The SCRNDUMP module allows you to print the current contents of the console display by pressing <ctl=>. All characters from the display will be printed on the system printer (the LST: device) except for graphics characters. Any graphics characters will be replaced by spaces so that the spacing on the printed copy will be the same as on the screen. Reverse video is ignored during the print process so characters that are in reverse video will be printed the same as normal characters.

## 7.5 ADM3A Module

The ADM3A module makes the console display and keyboard emulate an ADM 3A terminal. With this module included in the system, the display will respond to the standard ADM 3A cursor positioning codes and the cursor arrow keys will generate the same codes as an ADM 3A terminal. In some cases, control sequences were taken from Soroc terminals for functions that are not available with the ADM 3A.

This feature is useful when running programs that make use of cursor addressing and other terminal features but do not support P&T CP/M 2 directly. If you can configure the program for P&T CP/M 2 you should do so to insure the maximum use of the features of the video display. For programs that cannot be configured to work with P&T CP/M 2, the ADM3A module should allow you to configure them for an ADM 3A terminal and use them.

Figure 7.6 shows the codes recognized by the ADM 3A terminal emulator module to perform various functions. Note that these are the codes that would be sent to the console by a program to accomplish the listed action.

character sequence	numeric codes (dec)	numeric codes (hex)	function
<ctrl-Z>	31	1Fh	new line (same as <cr><lf>)
<ctrl-H>	26	1Ah	clear screen, home cursor
<ctrl-J>	8	08h	move cursor left
<ctrl-K>	10	0Ah	move cursor down
<ctrl-L>	11	0Bh	move cursor up
<ctrl-M>	12	0Ch	move cursor right
	30	1Eh	home cursor
<esc>[	27, 91	1Bh, 5Bh	enable emulation
<esc>]	27, 93	1Bh, 5Dh	disable emulation
<esc>+	27, 43	1Bh, 2Bh	clear screen, home cursor
<esc>*	27, 42	1Bh, 2Ah	clear screen, home cursor
<esc>T	27, 84	1Bh, 54h	clear from cursor to end of line
<esc>Y	27, 89	1Bh, 59h	clear from cursor to end of screen
<esc>)	27, 41	1Bh, 29h	begin reverse video
<esc>(<	27, 40	1Bh, 28h	end reverse video
<esc>=rc	27, 61, r, c	1Bh, 3Dh, r, c	position cursor to row and column specified by "r" and "c". r = 32 + row number c = 32 + column number

Figure 7.6 Code Sequences Recognized by ADM3A Module

Figure 7.7 shows the character codes that are generated by the arrow keys on the keyboard when the ADM 3A terminal emulator module is in effect.

key	code (dec)	code (hex)
<left arrow>	8	08h
<right arrow>	12	0Ch
<up arrow>	11	0Bh
<down arrow>	10	0Ah

Figure 7.7 Character Codes Generated by Arrow Keys with ADM3A

With the ADM3A module included in the system, any programs that use the standard P&T CP/M 2 console control codes (see Chapter 10) will not work properly. There are two ways to disable the module so that the standard P&T CP/M 2 control codes will be recognized. Escape sequences, as shown in Figure 7.8, are recognized to turn the module on and off. Also Special System Function 31 has been included in P&T CP/M 2 to enable and disable the ADM3A module and others like it.

character sequence	numeric codes (dec)	numeric codes (hex)	function
<esc>[	27, 91	1Bh, 5Bh	enable emulation
<esc>]	27, 93	1Bh, 5Dh	disable emulation

Figure 7.8 Escape Sequences to Enable and Disable ADM3A

Programs that use the standard P&T CP/M 2 console control codes can use these escape sequences or this special system function to disable the module when they begin execution and re-enable it before returning to the system. The special system functions are discussed in Chapter 16. An example of how this can be done from a Basic program is shown in Figure 7.9. All of the P&T CP/M 2 utility programs that make use of the console control codes use this special system function to insure that they will work even when the ADM3A module is included in the system.

```

10 SYSENT%=&H43           'entry point for special system functions
20 !
30 !  turn off terminal emulator if one is installed
40 POKE &H46,0 : POKE &H47,31   'setup registers B & C
50 CALL SYSENT%             'call special system function
.
.
.
60000 !
60010 !  turn terminal emulation back on before exiting program
60020 POKE &H46,&HFF : POKE &H47,31   'setup registers B & C
60030 CALL SYSENT%           'call special system function
60040 STOP

```

Figure 7.9 Using Special System Function to Turn ADM3A Module On and Off

## NOTES

## 8.1 Introduction

P&T CP/M 2 includes a variety of utility programs to perform useful and needed system functions. These include the standard CP/M 2 utility programs, as well as a number of special purpose programs developed by Pickles & Trout especially for the TRS-80 hardware.

Most of the utility programs have an interactive mode of operation in which the program asks you various questions to determine what you want done. For example the FORMAT program will ask for the drive on which to format a diskette, the density at which to format it, whether you want to make a quick check for flaws, and so forth. When you wish to use the interactive mode of a utility program you merely type the name of the program at the command level of the system and answer the questions it asks.

Although we have tried to show as many "screen images" as possible in the descriptions of the utility programs, it is very difficult to learn how to use a program just from reading a manual. We strongly suggest that you actually execute the programs as you read about them. You will learn faster if you actually see what happens as the program runs. Note that some programs have the potential for erasing or altering information on disks. You should make up some "learning" diskettes that contain no important information to use while you learn about the utility programs.

## 8.2 Command Line Mode

Some of the utility programs have a command line mode. In this mode of operation, all instructions about what the program is to do are given on the command line that executes the program. This mode was developed specifically for use in integrated applications. By using SUBMIT, some sort of menu program, or a chaining facility, you can organize your system so that a user can perform system functions (like formatting a diskette, changing the serial port setup, copying diskettes, etc.) without ever dealing directly with the utility programs involved.

When the command line mode is used, the output from a utility program includes messages indicating what the program is doing and reporting any errors that occur. The command line option allows you to direct this output from a program to the console, a printer, a disk file, or any combination of them. This feature gives you a way to log exactly what happens when the utility program is executed. It also allows you to detect errors and deal with them in your own way. You can, for example, direct the output from a utility program to a disk file and then have your program look at the contents of the file to check for error messages. When the output is directed to a disk file, it will not appear on the console unless you specifically direct it to the console as well.

Since the output from several utility programs may be directed to the same disk file, the beginning of each program is marked by a line beginning with two asterisks ( \*\* ). This allows a program looking at the file to easily find the beginning of the output from each utility program. Error messages always appear on lines that begin with four greater than signs ( >>> ). A program can check for errors simply by scanning for lines that begin with >>>.

Section 8.34 of this chapter gives several examples of using the command line options.

### 8.3 List of Utility Programs

The distribution diskette you received should contain the following utility programs:

<u>page</u>	<u>program</u>	<u>description</u>
8.4	ASM	The standard CP/M 2 assembler.
8.9	ASSIGN	P&T CP/M 2 utility to quickly change the I/O device assignment (e.g. change the system printer from the parallel port to a serial port).
8.13	CLEAN	P&T CP/M 2 utility for use when cleaning diskette drive heads.
8.15	CLONE	P&T CP/M 2 utility to make an image copy of a diskette. When a diskette is copied with this program random access files will be properly copied.
8.27	DATIME	P&T CP/M 2 utility to display the current system date and time.
8.29	DDT	Standard CP/M debugger.
8.36	DENSITY	P&T CP/M 2 utility to report and, optionally, change the density flag on a diskette.
8.39	DISKCHK	P&T CP/M 2 utility to check a diskette for flaws. This program does not affect the data on a diskette and may be used at any time.
8.26	DISKTEST	P&T CP/M 2 utility to test a diskette for flaws. This program destroys any data on a diskette and should be used only on a new diskette or when reusing a diskette.
8.55	DUMP	Standard CP/M 2 program to display a disk file in hexadecimal format.
8.56	ED	Standard CP/M 2 line oriented editor.
8.57	ERROR	P&T CP/M 2 utility to explain error messages given by the system.
8.58	FASTCOPY	P&T CP/M 2 utility to make a file by file copy of all files under all user numbers from one drive to another. FASTCOPY reads as many files from the source drive into memory as will fit before beginning to write them out to the destination drive. Since it performs file copies, FASTCOPY may not copy random access files correctly.
8.69	FORMAT	P&T CP/M 2 utility to format diskettes.
8.78	KXEDIT	P&T CP/M 2 utility to edit the key translations performed by the KEYXLATE module when it is included in the system.
8.82	LOAD	Standard CP/M 2 utility that creates an executable "COM" file from the "HEX" file generated by an assembler.
8.84	MODSEL	P&T CP/M 2 utility to select the modules that are to be included in the system when it is loaded.
8.85	PATCH	P&T CP/M 2 utility for installing P&T supplied patches to programs.
8.90	PIP	Standard CP/M 2 program for file transfer. Since PIP performs a file copy operation, it may not copy random access files correctly.

8.104 SETCCB	P&T CP/M 2 utility for setting the P&T CCB clock, calendar, and bell board.
8.108 SETDATE	P&T CP/M 2 utility for setting the system date.
8.111 SETMISC	P&T CP/M 2 utility for setting the following miscellaneous I/O parameters: CRT parameters, floppy drive head step rate, and parallel printer port options.
8.120 SETTIME	P&T CP/M 2 utility for setting the system time.
8.122 SETUP	P&T CP/M 2 utility for setting the serial port parameters and the I/O device assignments.
8.130 STAT	Standard CP/M 2 utility for displaying statistics on disk drives and files.
8.137 SUBMIT	Standard CP/M 2 utility for submitting multiple command lines for batch processing.
8.140 SYNCRO	P&T CP/M 2 utility for synchronizing the system date and time to the P&T CCB clock, calendar, and bell board.
8.141 TRS2CPM	P&T CP/M 2 utility to transfer files from a TRSDOS diskette to a CP/M diskette.
8.149 VERIFY	P&T CP/M 2 utility to verify that a copy of a utility program is correct. This program may be used periodically to insure that nothing has damaged the copies of programs you are using.
8.151 XSUB	Standard CP/M 2 utility that allows a submit file to feed console input into running programs.

**8.4 Utility name: ASM**

**Purpose:** Translation of 8080 assembly language source files into object code represented in HEX format.

**General Description**

ASM is a system utility which reads 8080 assembly language source files and translates them into 8080 machine language instructions. The resulting machine code is stored in Intel hex format in a disk file. The HEX file may be read by other programs (e.g. DDT) or may be converted to an executable COM file by the LOAD utility. ASM may also generate a PRN file, which contains the original source code with the assembled machine code. The format of this source file is compatible with the Intel 8080 assembler, except that no macros are implemented.

A complete discussion of assembly language programming is beyond the scope of this section. Many books are available on the subject, a few of which are listed in the references at the end of this section. Note that special steps must be taken to assemble Z-80 code with ASM. See Appendix B of this manual for details.

**Using ASM**

There are two forms of command lines which may be used to execute ASM, as shown below in Figure 8.1. Both specify the primary name of the file containing the source code to be assembled. ASM requires that the extension of this file name be "ASM".

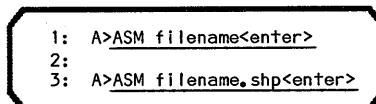
- 
- 1: A>ASM filename<enter>
  - 2:
  - 3: A>ASM filename.shp<enter>

Figure 8.1 Command lines for executing ASM

When ASM is executed by the command line shown on Line 8.1-1, it assumes that the source code file will be found on the current default disk drive and that the HEX and PRN files are to be saved on that same drive. The second command line, shown on Line 8.1-3, allows you to specify the drives to be accessed for each of the three files involved (source, HEX, and PRN). The first letter following the period (which replaces the "s") indicates the drive from which to get the source file, the second letter (which replaces the "h") indicates the drive on which to save the HEX output file, and the third letter (which replaces the "p") indicates the drive on which to save the PRN output file. The valid drive indicators for each file are given in Figure 8.2.

<u>position</u>	<u>valid letters</u>	<u>meaning</u>
s	A, B, ..., P	Indicates logical drive on which the source file resides.
h	A, B, ..., P or Z	Indicates logical drive on which to save the HEX file. Indicates that no HEX file is to be generated.
p	A, B, ..., P or X or Z	Indicates logical drive on which to save the PRN file. Sends the PRN file to the console. Indicates that no PRN file is to be generated.

Figure 8.2 Valid drive designations for ASM command line

Figure 8.3 shows a short assembly language program which merely prints a message to the system console, waits for <enter> to be typed, and then returns to the system.

```
; Program to wait for operator input
org 100h ;locate at beginning of TPA
lxi d,msg ;send message to console
mvi c,9
call 5
loop: mvi c,1 ;get char from console
call 5
cpi 0dh ;check for <enter>
jnz loop ;loop if not <enter>
jmp 0 ;return to system on <enter>
msg: db 'Press <enter> to continue.$'
end
```

Figure 8.3 Source code of sample assembler program

Figure 8.4 shows the console dialog which will assemble (Line 8.4-1), load (Line 8.4-7), and execute (Line 8.4-15) the sample program shown in Figure 8.3. Note that the source code file is named "ASMDEMO.ASM" and is located on logical drive C. Also note that the HEX and PRN files are routed to logical drive C.

```
1: A>ASM C:ASMDEMO.CCC<enter>
2: CP/M ASSEMBLER - VER 2.0
3: 0130
4: 000H USE FACTOR
5: END OF ASSEMBLY
6:
7: A>LOAD C:ASMDEMO<enter>
8:
9: FIRST ADDRESS 0100
10: LAST ADDRESS 012F
11: BYTES READ 0030
12: RECORDS WRITTEN 01
13:
14:
15: A>C:ASMDEMO<enter>
16: Press <enter> to continue.<enter>
17: A>
```

Figure 8.4 Console dialog to assemble, load, and execute sample program

Line 8.4-2 shows ASM's sign-on message, and Line 8.4-3 gives the last address in memory that was used by the assembled program. In this example, the last address used is 130h. Line 8.4-4 indicates the amount of symbol table space during the assembly. The larger the number specified, the more symbol table space that was used. Since this example uses a very short program, the use factor is shown as 0. When it is finished, ASM displays the message shown on Line 8.4-5 just before returning to the operating system.

Figure 8.5 shows the PRN file that resulted from the assembly shown in Figure 8.4.

```
; Program to wait for operator input
0100      org    100h ;locate at beginning of TPA
0100 111501  lxi   d,msg ;send message to console
0103 0E09    mvi   c,9
0105 C05000  call  5
0108 0E01    loop: mvi   c,1 ;get char from console
010A C0D500  call  5
010D FE0D    cpi   0dh ;check for <enter>
010F C20801  jnz   loop ;loop if not <enter>
0112 C30000  jmp   0     ;return to system on <enter>
0115 5072657373msg: db    end   'Press <enter> to continue.$'
0130
```

Figure 8.5 PRN file generated by the assembly of the sample program

### Possible Error Messages

If an error in the source code is discovered in the course of assembly, the source line containing the error is displayed on the system console. A single character error indicator is shown at the leftmost end of the displayed line and is also inserted at the leftmost end of the line in the PRN file (if one is generated). The assembler will continue execution until the assembly is finished, even when errors have been reported. In many instances, some of the error messages will be superfluous. A single error in the source code can cause a large number of errors to be reported.

Figure 8.6 shows the error indicators used by ASM.

<u>Indicator</u>	<u>meaning</u>
D	Data error: An element in a data statement does not fit in the specified data area. This typically is caused by specifying a 16 bit quantity in a DB statement.
E	Expression error: An expression has been found that does not conform to the syntax which is accepted by ASM and hence cannot be evaluated.
L	Label error: A label is used in an illegal context or a duplicate label exists.
N	Not implemented: Some features (such as macros) are recognized by ASM as valid assembly language statements but are not implemented by the current version of the assembler.
O	Overflow: An expression has been found that is too complex (has too many pending operators) to be evaluated.
P	Phase error: A label does not have the same value on two successive passes through the source code.
R	Register error: A register has been specified that is not consistent with the machine instruction used. For example, specifying register C in a PUSH instruction.
V	Value error: An operand has been found which is inconsistent with its use.

Figure 8.6 ASM error indicators

8

Several conditions related to the operating system environment may prevent continuation of the assembly. If any of these conditions is encountered, one of the following messages is displayed on the console and the assembly is aborted.

#### NO SOURCE FILE PRESENT

This message indicates that ASM could not find the source file on the specified source drive. This is typically caused by one of three errors:

1. The file does not exist.
2. You have specified the wrong source drive.
3. The file does exist but does not have the extension "ASM".

#### NO DIRECTORY SPACE

The directory on the drive(s) specified for the "HEX" and "PRN" files is full, making it impossible to create the output files.

#### SOURCE FILE NAME ERROR

The source file name you have given contains invalid characters. Note: Wildcard characters are not allowed.

#### SOURCE FILE READ ERROR

An error was encountered while reading the source file.

#### OUTPUT FILE WRITE ERROR

An error was encountered while writing to one of the output files. Typically, it results from a disk that has become full.

#### CANNOT CLOSE FILE

One of the output files could not be successfully closed.

Some reference material for assembly language programming is listed in Figure 8.7.

CP/M Assembler (ASM) User's Guide, Digital Research  
MCS-80/85 Family User's Manual, Intel Corp.  
8080/8085 Reference Card, Intel Corp.  
8080/8085 Assembly Language Programming Manual, Intel Corp.  
Zilog Z80-CPU Programming Reference Card, Zilog Corp.

Figure 8.7 References for assembly language programming

**8.5 Utility name: ASSIGN**

**Purpose:** To change the logical to physical I/O device assignments. (See Section 5.7 for an introduction to device assignments.)

**General Description**

ASSIGN provides a quick and easy way to change the current logical to physical I/O device assignments. It has both an interactive mode and a command line mode. In the interactive mode, ASSIGN shows a visual representation of the current device assignments and allows you to change them as you wish. The command line mode provides a quick way of setting the device assignments which can be used directly from the keyboard, from a submit file, or from a menu program.

Note that ASSIGN allows you to change the physical device assignment for the CON: logical device. This device is usually set to the CRT which is the built-in display and standard keyboard. If you change this assignment, some other physical device (a serial port, for example) will become the system console. If you have nothing connected to that device or if that device is connected to an output only device (like a printer) you will lose control of the system and need to reload the system to regain control. If you switch the CON: device to a serial port which has a terminal connected to it, that terminal will become the system console.

The function performed by ASSIGN is also performed by part of SETUP. If all you want to do is change the I/O assignment, using ASSIGN is easier and quicker than using SETUP. Note that the changes made to the I/O assignment by ASSIGN are made in the running system only. They will not remain in effect when the system is reloaded (RESET). If you want to make them permanent, you must use the FR option of MENU after running ASSIGN.

**Using ASSIGN - Interactive Mode**

The command line to execute ASSIGN in the interactive mode is shown in Figure 8.8.

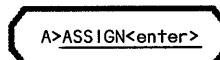


Figure 8.8 Command Line to Execute ASSIGN

When ASSIGN begins execution, it will present a display like the one shown in Figure 8.9. Lines 8.9-8 through 8.9-11 will show the current I/O assignments for each of the logical devices CON:, LST:, PUN:, and RDR: by highlighting in reverse video the physical device assigned to each one. The cursor will be positioned at SIOA in Line 8.9-8. Line 8.9-15 gives a brief summary of what you can do. (Note: an average user will never need to bother with the PUN: and RDR: logical devices.)

```
1:  
2:          ====== ASSIGN ======  
3:  
4:          I/O Assignment Utility - ver 2.3  
5:          Copyright 1980,83 Pickles & Trout  
6:  
7:  
8:          CON:      SIOA      CRT      CENTRON   SIOB  
9:          LST:      SIOA      CRT      CENTRON   SIOB  
10:         PUN:     SIOA      UPUN1    UPUN2     SIOB  
11:         RDR:     SIOA      URDR1    URDR2     SIOB  
12:  
13:  
14:  
15:   Arrows move cursor | <hold> - set field | <esc> - accept | <break> - quit
```

Figure 8.9 Interactive Mode of ASSIGN

You may move the cursor to any of the physical devices by using the arrow keys on the keyboard. To make an assignment, move the cursor to the desired physical device on the line for the logical device and press **<hold>**. The physical device name will then be highlighted and any other name on that line that had previously been highlighted reverts to normal characters.

Note that the actual assignment does not take place immediately. The assignment will take effect only after you press **<esc>** to accept the assignments. Upon your pressing **<esc>** the assignments shown in the display will take effect and you will be returned to the command level of the system. The assignments will be made in the system running in memory only; they will not be in effect after the system is reloaded (RESET). If you decide that you don't want to make any changes in the assignments, press the **<break>** key to return to the command level of the system with no action being taken.

The meanings of the various physical device names are shown in Figure 8.10.

```
for the CON: (system console) logical device:  
  SIOA    Use serial port A for input and output.  
  CRT     Use system keyboard for input and system display for output.  
  CENTRON Use system keyboard for input and parallel printer port for output.  
  SIOB    Use serial port B for input and output.  
  
for the LST: (system printer) logical device:  
  SIOA    Use serial port A for output.  
  CRT     Use system display for output.  
  CENTRON Use parallel printer port for output.  
  SIOB    Use serial port B for output.  
  
for the PUN: (punch) logical device:  
  SIOA    Use serial port A for output.  
  UPUN1   User supplied output routine.  
  UPUN2   User supplied output routine.  
  SIOB    Use serial port B for output.  
  
for the RDR: (reader) logical device:  
  SIOA    Use serial port A for input.  
  URDR1   User supplied input routine.  
  URDR2   User supplied input routine.  
  SIOB    Use serial port B for input.
```

Figure 8.10 Meanings of Physical Device Names

## Using ASSIGN - Command Line Mode

The command line mode of ASSIGN allows you to set one or more I/O assignments by typing a single command line. The general form of the command line is shown in Figure 8.11. There may be up to four assignments specified on the command line. The assignments take effect immediately and you are returned to the command level of the system.

```
ASSIGN [assignment-1] [assignment-2] [assignment-3] [assignment-4]
       (required)      (optional)     (optional)     (optional)

each assignment has the form:
Logical Device Name = Physical Device Name

Assignments must be separated from one another by at least one space.
```

Figure 8.11 Form of Command Line Mode for ASSIGN

For ease of typing, ASSIGN accepts a short synonym for each logical device name in the command line mode. These synonyms are shown in Figure 8.12.

logical device name	synonym
CON:	C:
LST:	L:
RDR:	R:
PUN:	P:

Figure 8.12 Synonyms Accepted by ASSIGN for Logical Device Names

Several synonyms are recognized for each of the physical device names. You may use the complete name or any of the synonyms listed in Figure 8.13. See Figure 8.10 for the meaning of each physical device name.

physical device name	synonyms
SIOA	SA, A
SIOB	SB, B
CRT	C
CENTRON	CENT, CE, E
UPUN1	UP1
UPUN2	UP2
URDR1	UR1
URDR2	UR2

Figure 8.13 Synonyms Accepted by ASSIGN for Physical Device Names

Examples of using the command line mode of ASSIGN are:

### ASSIGN LST:=SIOA

Assigns serial port A to be the system printer.

### ASSIGN RDR:=SIOB

Assigns serial port B to the reader device.

### ASSIGN R:=SA P:=SB L:=CE

Assigns serial port A to the reader device, serial port B to the punch device, and the parallel printer port to the system printer.

## Possible Error Messages

### Invalid Assignment.

This message is given when an assignment is given on the command line that cannot be deciphered. You must use only the logical and physical device names and their valid synonyms as previously described. Note that all valid assignments appearing on the command line before the invalid assignment will be made.

**8.6 Utility name: CLEAN**

**Purpose:** To load the heads on a floppy drive for cleaning with a cleaning diskette.

### General Description

The CLEAN utility program is designed to provide an easy way to clean the read/write heads on floppy diskette drives. It will load the heads on the specified drive and move them around in a random pattern for about 20 seconds. After the random movements cease, the heads will be unloaded in about 2.5 seconds.

Note that for thinline drives (like those used on the Models 12 and 16), the heads are loaded as soon as the diskette is inserted and the bar on the front of the drive rotated (ie. the drive is closed). Since the diskette will remain turning for about 20 seconds after the light on the front of the drive goes out, you should remove the cleaning diskette as soon as the message instructing you to do so appears on the console.

### Using CLEAN

The CLEAN program has only an interactive mode. Figure 8.14 shows a sample dialog from using CLEAN.

```
1: A>CLEAN<enter>
2:
3:          Pickles & Trout Head Cleaning Utility Ver 2.xx
4:          (c) copyright 1982,83 Pickles & Trout
5:          all rights reserved
6:
7:          (Hit <BREAK> at any time to quit)
8:
9:
10: Enter drive to clean (0-3 or A-P) : 1<enter>
11:
12: Mount the cleaning disk on drive B and press <ENTER> when ready : <enter>
13: xx
14:
15: Remove cleaning diskette now, press <ENTER> : <enter>
16:
17: A>
```

Figure 8.14 Example of Using CLEAN

On Line 8.14-1 is the command line which executes the CLEAN program. After displaying its initial messages (Lines 8.14-3 to 8.14-7) CLEAN asks you for the drive to clean (Line 8.14-10). Note that you may enter either the physical floppy drive number or the logical drive letter by which it is referred to with P&T CP/M 2. If you specify a drive that is not defined for the system, an error message will be displayed and you will be asked again for the drive to clean.

After entering a valid drive to clean, you will be asked to mount the cleaning diskette and press <enter> (Line 8.14-12). At this time you should prepare the cleaning diskette, mount it on the drive and press the <enter> key. Note that on Line 8.14-12 the drive is always referred to by the logical drive letter. If you have

specified a physical drive number, CLEAN will determine the logical drive letter assigned to it and use that letter for this prompt line.

After you press <enter> on Line 8.14-12, CLEAN will load the heads on the selected drive and begin to move them in a random pattern. On Line 8.14-13, CLEAN shows a running time for the cleaning as it proceeds. After 20 seconds, CLEAN will flash the message shown on Line 8.14-15 indicating that the cleaning operation is complete. At this time you should remove the diskette from the drive and press <enter>.

If a system diskette is required for a warm boot, CLEAN will prompt you to mount one on the system drive before returning to the system. If you see this prompt, insure that a system diskette is mounted and press <enter>. If a system diskette is not required for the warm boot, CLEAN will return immediately to the operating system after you press <enter> on Line 8.14-15.

### Possible Error Messages

#### Please respond with A-P or 0-3 only.

This message is displayed if you enter an illegal character when specifying the drive to clean. You will then be prompted again for the drive. Use only characters in the specified range in response to this question.

#### That drive is not on the system or is a hard disk drive.

This message indicates that you have specified a drive that is not defined for the system or is a hard disk drive. CLEAN works only with floppy diskette drives. Double check which drive you want to clean and enter its physical drive number (0 - 3) or its logical drive letter.

#### >>> Home error.

This message is given if the CLEAN program encountered an error while attempting to restore the read/write head to the home position. This can be caused by too fast a head stepping rate or, in some cases, not closing the drive door after the cleaning diskette is mounted.

**8.7 Utility name: CLONE****Purpose:** To make an image copy of one diskette to another.**General Description**

CLONE makes image copies from one diskette to another. An image copy results in two identical diskettes. CLONE is the fastest way with P&T CP/M 2 to make a backup copy of a diskette. Figure 8.15 shows typical times for copying diskettes using CLONE.

type of copy	no verification	with verification
single density	53 sec	78 sec
double density	53 sec	82 sec
double sided	97 sec	154 sec

Figure 8.15 Copying Times for CLONE

Since CLONE makes an exact copy of the diskette, it will not correct any fragmentation of files that has been created by using the diskette. In order to remove fragmentation, you must copy the files from one diskette to another using a program that does file by file copying like PIP or FASTCOPY. CLONE does have the advantage of speed and, because it makes an image copy, any random access files on the diskette will not be affected by the copying process. (A file copy program like PIP or FASTCOPY can result in loss of data when random files are copied.)

CLONE gives you the option of copying the data tracks, the system tracks, or all tracks of a diskette. You also have the option of verifying each track as it is copied, although this will slow down the copy process somewhat. You can also instruct CLONE to stop copying when it reaches the first empty track (a track containing nothing but E5h data bytes, the pattern left by the FORMAT program). When copying diskettes containing random access files, it is usually best to copy the entire diskette.

CLONE has both an interactive and a command line mode of operation.

**Using CLONE - Interactive Mode**

CLONE's interactive mode of operation prompts you for all the information necessary to specify the copy operation to be performed. It also provides the option of performing the same operation repetitively without re-entering the information. Figure 8.16 shows a typical console dialog for using CLONE in the interactive mode.

Line 8.16-1 shows the command line that will execute CLONE in the interactive mode. After CLONE displays its opening messages (Lines 8.16-4 to 8.16-6) it reminds you (Lines 8.16-9 to 8.16-11) that any data on the destination diskette will be destroyed by CLONE unless you choose the "system tracks only" option. Copying only the system tracks will not affect the data stored on the diskette.

```
1: A>CLONE<enter>
2:
3:
4:           1D/2D/2S Disk Clone Utility - Ver 1.xxx
5:           (C) copyright 1982,83 Pickles & Trout
6:           all rights reserved
7:
8:
9: Caution:
10:   This routine will destroy any & all existing files on the destination disk.
11:   (except if the "System Tracks Only" option is selected)
12:
13:
14:           Press <BREAK> at any time to quit.
15:
16: Enter source drive (A-P or 0-3) : B<enter>
17:   NOTE: Logical drive B is the same as physical drive (1).
18:
19: Enter destination drive (A-P or 0-3) : 2<enter>
20:   NOTE: Logical drive C: is the same as physical drive (2).
21:
22:
23: Which tracks would you like to copy?
24:   1. Data tracks only
25:   2. System tracks only
26:   3. Data and system tracks
27: Enter your choice by number : 3<enter>
28:
29: Do you want each track verified as it's copied? . . . (y/n) : Y<enter>
30:
31: Do you want copying to stop if an empty track is found? (y/n) : N<enter>
32:
33: Mount source disk on drive B:
34: Mount destination disk on drive C:
35:   Press <ENTER> when ready : <enter>
36:
37:
38: Copying from drive B: to drive C: at Double Density.
39: ++++++...
40: Data and System tracks copied.
41:
42:
43: Caution:
44:   This routine will destroy any & all existing files on the destination disk.
45:   (except if the "System Tracks Only" option is selected)
46:
47:
48:           Press <BREAK> at any time to quit.
49:
50: Enter source drive (A-P or 0-3) or (R) to repeat : <break>
51:
52: A>
```

Figure 8.16 Console Dialog for an Example of Using CLONE

On Line 8.16-14 CLONE reminds you that you may abort the program at any time (even in the middle of copying) by pressing the <break> key. Note that if you do press <break> during the copy process, the copy will not be completed and the state of the destination diskette will, in general, be unknown. You may CLONE to the diskette again or erase any files that show up in the directory and use it as an empty diskette. Attempting to use a partially CLONE'd diskette may cause unpredictable results.

On Line 8.16-16 CLONE asks you for the source drive. This is the drive on which you will mount the diskette to be copied. Note that you can enter the drive specification as either a physical drive number or a logical drive letter. The physical floppy drive numbers are always 0 - 3 but in a hard disk system the logical drive letters assigned to the physical drive may change depending on how you

configure the system. You may choose the method of specifying the drive which is most comfortable for you. In this example, logical drive B is selected for the source drive. On Line 8.16-17 CLONE shows you that on this system logical drive B is equivalent to physical drive 1.

A similar query is made for the destination drive on 8.16-19. The destination drive is the drive on which you will mount the diskette to which the copy will be made. In this example the destination drive is specified as physical drive 2. On Line 8.16-20, CLONE shows that logical drive C corresponds to physical drive 2.

After you have specified the source and destination drives, CLONE will ask you what sort of copy you would like as shown on Lines 8.16-23 to 8.16-27. Copying the data tracks only (option 1) will copy all files to the destination diskette but will leave the system area of the destination diskette unaffected. This can be useful when you want to copy only the data area from your system diskette to a backup diskette. Many people do not keep the system on backup diskettes since it then becomes one more system diskette to keep track of. Remember you are responsible under the P&T CP/M 2 license to know the whereabouts of ALL system diskettes.

Copying the system tracks only (option 2) is useful when you need to make a double density diskette into a bootable system diskette. This operation will not affect the data stored on the destination diskette in any way. Copying all tracks (option 3) will make a complete image copy of the source diskette on the destination diskette. The copy will include the operating system if it was present on the source diskette. This operation is one way to make additional working system diskettes.

In this example, option 3 is chosen to copy all tracks on the diskette.

After specifying the type of copy operation, CLONE asks you if you want the copy verified as it is made (Line 8.16-29). If you respond with an affirmative answer, each track will be read back from the destination diskette and checked for errors. A negative response will omit the verification. When verification is selected, the copy operation will take about twice as long as when verification is not performed. In this example verification is selected.

Next CLONE asks you if you want to stop the copy operation when the first empty track is found (Line 8.16-31). CLONE considers a track to be empty if all of the data areas on that track contain E5h bytes (a newly formatted diskette has E5h in all its data areas). This option is useful in certain copy operations since, by stopping when it encounters an empty track, it can take noticeably less time than copying the entire diskette. You might want to use this feature if you are making copies of programs you have written for distribution. If you copy all the program files onto a newly formatted diskette with a file copy program such as PIP, they will fill the data area of the diskette sequentially. If you then use this diskette as your duplication master and tell CLONE to stop when it finds an empty track, you will minimize the duplication time.

You should not use this option, however, when you are not certain that all data on the diskette is at the beginning of the data area. It is possible, in rare circumstances, for data to be recorded on the diskette in such a manner that a track is skipped. In this case, if you tell CLONE to stop copying when an empty track is encountered, it will not copy all of the data on the diskette.

After you have entered all the necessary information, CLONE will ask you to mount the source and destination diskettes as shown on Lines 8.16-33 to 8.16-35. At this point you should mount the diskettes and press <enter>.

After you press <enter>, CLONE will make a variety of checks on the diskettes to insure that a copy can be made. It checks that both diskettes are of the same density since it is not possible to make an image copy of one density diskette onto a different density diskette. Note: double sided diskettes are considered to be of a different density than single sided diskettes.

CLONE also checks whether files exist on the destination diskette. If there are files on the destination diskette, CLONE reports the fact and asks you to verify that you want to continue with the copy process. This helps to protect you if you forget to mount the diskettes at the proper time. In addition CLONE checks for files on the source diskette. If no files are found on the source diskette, CLONE informs you of the fact and asks you to confirm that you want to continue with the copy operation. Mounting the diskettes on the wrong drives or mis-specifying the source and destination drives are frequent causes of these warnings. If you get either warning, you should check to insure that you have not made a mistake.  
**Remember that CLONE'ing a newly formatted diskette onto your source diskette will irrevocably destroy the information recorded on the source diskette!!**

Once CLONE has made its initial tests, it displays a message to indicate what it is doing (Line 8.16-38) and begins the copy operation. As each track is read from the source diskette, CLONE displays a period ( . ) on the console display. As the tracks are written to the destination diskette, the periods are replaced by plus signs ( + ). Line 8.16-39 shows the copy operation after 11 tracks have been read and 8 tracks have been written. If an entire diskette is being CLONE'd, the plus signs will extend across the console display by the time the copy is completed.

When the copy is finally complete, CLONE displays a message indicating what has been accomplished (Line 8.16-40). CLONE then starts over again as shown in Lines 8.16-43 to 8.16-50. Note that on Line 8.16-50 a new option is available to you. By entering "R" instead of a drive designation, you may repeat exactly the same copy operation you just completed. This is useful when making multiple copies in a production environment. In this example the <break> key is pressed to return to the operating system. In this example, the system does not require a system diskette for a warm boot so CLONE returns immediately to the operating system after the <break> key is pressed. If a system diskette were required to accomplish the warm boot, a prompt to mount a system diskette on the system drive would have been given before returning to the system command level.

You may specify the same drive for both the source and destination drives. This will cause CLONE to issue prompts to mount either the source or destination diskettes as needed for the copy operation. An example of a prompt for mounting the source diskette is given in Figure 8.17. Specifying the same drive for source and destination should be avoided unless there is no other choice (in a single floppy drive system, for example) since it will require quite a bit of disk swapping. The periods and pluses are not displayed in the single drive mode of CLONE since they tend to get lost in the swapping messages.

Mount source disk on drive X: and press <ENTER> when ready:

Figure 8.17 CLONE Message to Mount Source Diskette

### Using CLONE - Command Line Mode

In the command line mode of CLONE, you must specify all of the information regarding the copy operation you want performed on the command line that you use to execute CLONE. Note that in the command line mode, the copying normally begins automatically as soon as CLONE begins execution (except on a single floppy drive system). You should insure that the source and destination diskettes are mounted before executing CLONE in the command line mode. (See the PMOUNT instruction below for an exception to this.)

The general form of the CLONE command line is shown in Figure 8.18.

```
CLONE [source and destination drives] [output destinations] [options]
       (required)           (optional)        (optional)

All instructions on the command line must be separated from one another
by a comma and/or one or more spaces.
```

Figure 8.18 General Form of CLONE Command Line

You must specify the source and destination drives when using the command line option of CLONE. You may use either physical drive numbers or logical drive letters when specifying the drives. The source drive is specified using an expression like those shown in Figure 8.19. Note that the drive designation may come before or after the equal sign.

```
SOURCE=[drive]          SRC=[drive]
[drive]=SOURCE          [drive]=SRC

[drive] should be replaced by a logical drive letter
or a physical drive number.
```

Figure 8.19 Specifying the Source Drive

In a similar manner, the destination drive is specified as shown in Figure 8.20.

```
DESTINATION=[drive]      DEST=[drive]      DST=[drive]
[drive]=DESTINATION     [drive]=DEST     [drive]=DST

[drive] should be replaced by a logical drive letter
or a physical drive number.
```

Figure 8.20 Specifying the Destination Drive

There are also several short forms for specifying both the source and destination drives in a single statement. Examples of these short forms are given in Figure 8.21.

```
A>B      A is source, B is destination
B TO C   B is source, C is destination
B<A      A is source, B is destination
B=A      A is source, B is destination

Note: Physical drive numbers may be used
Instead of logical drive letters.
```

Figure 8.21 Short Forms of Specifying Source and Destination Drives

The command line mode of CLONE allows you to direct the console output from the program to the console, the system printer, or a disk file. The output defaults to the console if no output direction is specified. You may direct the output to any

combination of these three destinations. Any destination(s) you specify will be the only destination(s) for the console output. For example, if you specify a disk file for the destination, no output will appear on the console. If you want output to both a disk file and the console you must specify both as output destinations. A special output destination (NUL:) is provided for cases where you want no output at all from the program. An output destination is specified by a slash ( / ) followed by the destination name. Note that you cannot direct the console output to a disk file on a single floppy drive system unless there is also a hard disk on the system. Examples are given in Figure 8.22.

/CON:	sends output to the console (default if no output instructions are given).
/LST:	sends output to system printer.
/(valid file name)	sends output to the specified file. If a file with the same name already exists, it is replaced.
/(valid file name)+	appends output to the specified file. If the file does not already exist, it is created. The + may be preceded by a space or comma.
/NUL: or /NULL:	causes no output to be generated. If any other output instructions appear, they will supersede /NUL:.

Figure 8.22 Instructions to Direct Console Output

There are several additional instructions that can appear on the command line of CLONE. They are listed in Figure 8.23. In the command line mode, CLONE normally does not prompt for disks to be mounted; it immediately begins the copy operation specified on the command line. When it is finished, it does not prompt for a system diskette to be mounted either.

If you wish for CLONE to prompt for the source and destination diskettes to be mounted, you may use the "PMOUNT" instruction (or one of its synonyms) on the command line. This can be useful when CLONE is used from a SUBMIT file because it eliminates the need for a separate program to prompt for the diskettes to be mounted. If you want CLONE to prompt for a system diskette at the end of the copy operation you should include the "PSYS" instruction on the command line. This instruction causes CLONE to issue a prompt for the system diskette if one is required for a warm boot operation. No prompt will be issued for systems that do not require a system diskette for a warm boot. You may instruct CLONE to prompt both for the source and destination diskettes and for the system diskette by using the "PBOTH" instruction on the command line.

CLONE will always issue prompts for the source and destination diskettes and for the system diskette (if one is needed for a warm boot) if the system has only a single floppy drive.

PMOUNT or PM	causes CLONE to prompt for the source and destination diskettes to be mounted before beginning the copy.
PSYS or PS	causes CLONE to prompt for a system diskette to be mounted after the copy is completed.
PBOTH or PB	causes CLONE to prompt both for the source and destination diskettes and for a system diskette after the copy is completed.
V or VERIFY	causes CLONE to perform a read back verification of each track as it is written to the destination diskette.
SYS or SYSTEM	causes only system tracks to be copied.
ALL	causes all tracks (system and data) to be copied.
T	causes CLONE to stop when it reaches the first empty track.

Figure 8.23 Additional Command Line Instructions for CLONE

The "V" or "VERIFY" instruction allows you to tell CLONE to perform a read back verify on each track it copies. If you do not use this instruction, CLONE will default to no verification.

CLONE defaults to copying all data tracks of the DISKETTE. The "T" instruction tells CLONE to stop the copy operation after encountering a empty track. The "SYS" or "SYSTEM" instruction allows you to tell CLONE to copy only the system tracks. If you use the "SYS" or "SYSTEM" instruction, CLONE will ignore the "T" instruction. The "ALL" instruction tells CLONE to copy all tracks (system and data) on the diskette.

If two conflicting instructions appear on the command line ("ALL" and "SYS", for example) CLONE will follow the last one on the line.

The following examples show command lines that could be used with CLONE:

#### **CLONE SOURCE=B, DESTINATION=C**

Copy data tracks from the diskette on drive B to the diskette on drive C. The diskettes are assumed to be mounted already and the copy will begin immediately. Messages are sent to the console, no verification is done, and all data tracks are copied regardless of whether they are empty or not. CLONE will not prompt for a system disk after the copy.

#### **CLONE C>D,ALL,V**

Copy all tracks from the diskette on drive C to the diskette on drive D. The diskettes are assumed to be mounted already and the copy will begin immediately. Messages are sent to the console, verification is performed, and all tracks are copied regardless of whether they are empty or not. CLONE will not prompt for a system disk after the copy.

#### **CLONE C:=B: V T**

Copy data tracks from the diskette on drive B to the diskette on drive C. The diskettes are assumed to be mounted already and the copy will begin immediately. Messages are sent to the console, verification is done, and the copy will stop when the first empty track is encountered. CLONE will not prompt for a system disk after the copy.

#### **CLONE C>B,/CON:/,LST:**

Copy data tracks from the diskette on drive C to the diskette on drive B. The diskettes are assumed to be mounted already and the copy will begin immediately. Messages are sent to the console and the system printer, no

verification is done, and all data tracks are copied regardless of whether they are empty or not. CLONE will not prompt for a system disk after the copy.

**CLONE A>C,PBOTH,/B:LOG,V**

Copy data tracks from the diskette on drive A to the diskette on drive C. CLONE will prompt for the diskettes to be mounted before beginning the copy. Messages are sent to a file named LOG on drive B, verification is done, and all data tracks are copied regardless of whether they are empty or not. CLONE will prompt for a system disk after the copy if one is required for a warm boot.

**CLONE 1=2,V,T,/LOG+,/CON:,PSYS**

Copy data tracks from the diskette on physical drive 2 to the diskette on physical drive 1. The diskettes are assumed to be mounted already and the copy will begin immediately. Messages are sent to the console and appended to a file named LOG on the current default drive, verification is done, and the copy is ended when the first empty track is encountered. CLONE will prompt for a system disk after the copy.

### Possible Warning Messages

**Source disk on drive X: has no files.**

**Do you want to copy from it? (y/n) :**

When CLONE checked the source diskette it found that there were no files on it. This could indicate that you mounted the wrong diskette or specified the wrong source drive. CLONE issues this message and query to inform you of the situation and allow you to abort the operation if you have made a mistake. You should respond affirmatively if you want to continue with the copy and negatively if you want to stop it.

**Destination disk on drive X: is not empty.**

**Do you want to copy to it? (y/n) :**

When CLONE checked the destination diskette it found that there was some information recorded on it. This could indicate that you mounted the wrong diskette or specified the wrong source drive. Of course this message will be given any time you CLONE to a diskette that has been used. If you are reusing a diskette, you will probably want to continue with the copy even if you do get this message. CLONE issues this message and query to inform you of the situation and allow you to abort the operation if you have made a mistake. You should respond affirmatively if you want to continue with the copy and negatively if you want to stop it.

### Possible Error Messages

**Please answer with 1, 2 or 3 :**

This message is displayed if you did not enter one of the three valid numbers when specifying what tracks on the disk you want copied. You should re-enter your choice using one of the indicated numbers.

**Non-modular system, not compatible with this program.**

You are trying to run this program with a previous version of P&T CP/M 2. The program makes use of features found only in the modular version of P&T CP/M 2 and hence cannot run on this system.

**>>> Error: Not enough memory to run CLONE**

This message is displayed if there is not enough memory available for CLONE to run. It is very unlikely that you will encounter this error. If you do, you should check the amount of memory reserved above the operating system (MENU option LA). Reducing this reserved memory should allow CLONE to run.

**Copy aborted by user.**

This message is displayed if you decide to abort the copy before it starts (perhaps in response to one of the warning messages).

**Verify error on Track XXX**

If you requested verification of the copy and you are copying a single sided diskette, this message is displayed if CLONE finds a discrepancy when it reads back the destination diskette. This indicated a faulty copy; you should make another copy on another diskette or reformat the destination diskette and try again. The "XXX" will be replaced by the track number on which the error was encountered.

**Verify error on Cylinder XXX, Side Y**

This message has the same meaning as the previous one except that it is displayed if a double sided diskette is being copied. The "XXX" will be replaced by the cylinder number and "Y" will be replaced by the side number on which the error was encountered.

**Read error on Track XXX, Sector YYY**

When copying a single sided diskette, an error occurred while reading a diskette. This usually occurs on the source diskette. It can also occur on the destination diskette when verification is performed. "XXX" and "YYY" indicate the track and sector, respectively, at which the error occurred. If a read error occurs on the source diskette, the sector at which the error occurred may not be copied properly. In some cases mounting the diskette on a different drive may alleviate the problem. If a read error occurs on the destination diskette, you should make another copy to another diskette or reformat the diskette and try again.

**Read error on Cylinder xxx, Side y, Sector zzz**

This message has the same meaning as the previous one except that it is displayed if a double sided diskette is being copied. "xxx", "y", and "zzz" indicate the cylinder, side, and sector, respectively, at which the error occurred.

**Write error on Track xxx, Sector yyy**

An error occurred while writing to the destination diskette during a copy of a single sided diskette. If this error occurs, the copy probably will not be valid. If you requested verification, a verify error will also be reported on this sector. You should try the copy operation again with another destination diskette or reformat the destination diskette and try it again. The "xxx" and "yyy" indicate the track and sector, respectively, at which the error occurred.

**Write error on Cylinder xxx, Side y, Sector zzz**

This message has the same meaning as the previous one except that it is displayed if a double sided diskette is being copied. "xxx", "y", and "zzz" indicate the cylinder, side, and sector, respectively, at which the error occurred.

**Bad drive name, please re-enter (A-P or 0-3) :**

The logical drive letter you entered was not in the range A to P. Respecify the drive using one of the valid letters or a physical drive number in the range 0 to 3.

**Bad drive number, please re-enter (A-P or 0-3) :**

The physical drive number you entered was not in the range 0 to 3. Respecify the drive using one of the valid numbers or a logical drive letter in the range A to P.

**That drive is not on the system, please re-specify :**

The drive you have specified is not available on the system. Re-enter the drive specification indicating one of the drives that is available.

**That drive is a hard disk, please re-specify :**

The drive you have specified is assigned to a hard disk. CLONE does not work with hard disk drives. Re-enter the specification giving a floppy drive. Note that if you use a physical drive number, it will always refer to a floppy drive.

**Please answer with 'Y' for yes or 'N' for no :**

The response you gave to a yes/no message was not acceptable. Enter your response again using one of the characters indicated. Note that both upper and lower case are acceptable.

**Found empty track — Copy terminated.**

If you instruct CLONE to stop after the first empty track, and the first track it encounters is empty, this message will be given. In this case no copying was done.

**Copy aborted with <break>.**

**Last track written = zzz**

This message is displayed if you press the <break> key during a copy. The "zzz" will be replaced by the number of the last track that was copied before the operation was aborted. Note that a partially copied diskette may not be usable.

**Disk s are different formats — cannot copy.**

CLONE has found that the diskettes mounted on the source and destination drives are not of the same format. Since CLONE makes an image copy, it cannot copy from one format to another. Note that double sided diskettes are considered to be a different format than single sided even though they may both be double density. If you wish to copy from one format to another, you must use a file copy program like PIP or FASTCOPY.

**Unable to determine density of disk on drive X:**

CLONE could not find the density of the disk on the drive indicated. This may result from using an unformatted diskette or a diskette with an unsupported format. Check to see that the diskette is properly formatted. The "X" will be replaced by the logical drive letter of the drive on which the problem occurred.

**Disk x: is write protected — cannot copy to it.**

CLONE found that the destination diskette is write protected. Since CLONE must write to the destination diskette, the copy operation cannot proceed. This may occur if you mount the diskettes on the wrong drives or specify the wrong drive for source and destination. If the diskettes are mounted on the correct drives, you must write enable the destination diskette before the copy can be made. "x" is replaced by the letter of the logical drive on which the protected diskette was found.

**Drive x: is not ready.**

When CLONE attempted to access a diskette, it was found not to be ready. This typically occurs when the diskette is not mounted or the drive door is not closed.

**Seek error.**

An error occurred when CLONE was moving a read/write head. This may be caused by an improper drive step rate (see the system MENU, Chapter 4, or the SETMISC program, Section 8.25 for setting the step rate). It may also be caused by an unformatted or improperly formatted diskette.

**Home error.**

An error occurred when CLONE was restoring a read/write head to the home position. This may be caused by an improper drive step rate (see the system MENU, Chapter 4, or the SETMISC program, Section 8.25 for setting the step rate). It may also be caused by an unformatted or improperly formatted diskette.

**>>> Error: Logged-on drive same as destination drive**

(Command Line Mode) Since CLONE normally begins the copy immediately when used in the command line mode, it does not allow you to specify the current default drive as the destination for the copy operation. It makes the assumption that the current drive will have valid information on it. You should log on to another drive before executing CLONE. Note that if you use the PMOUNT instruction, CLONE will prompt for diskettes to be mounted and you will be able to specify the current default drive as the destination.

**>>> Error: Bad source drive name**

(Command Line Mode) CLONE found a source logical drive letter that is not in the range A to P on the command line.

**>>> Error: Bad source drive number**

(Command Line Mode) CLONE found a source physical drive number that is not in the range 0 to 3 on the command line.

**>>> Error: Source drive not on system**

(Command Line Mode) CLONE found a valid source drive designation on the command line but that drive is not available on the system.

**>>> Error: Bad destination drive name**

(Command Line Mode) CLONE found a destination logical drive letter that is not in the range A to P on the command line.

**>>> Error: Bad destination drive number**

(Command Line Mode) CLONE found a destination physical drive number that is not in the range 0 to 3 on the command line.

**>>> Error: Destination drive not on system**

(Command Line Mode) CLONE found a valid destination drive designation on the command line but that drive is not available on the system.

**>>> Error: Missing drive name**

(Command Line Mode) CLONE could not find the designations for one or both of the source or destination drives.

**>>> Error: Source drive is a hard disk drive**

(Command Line Mode) The drive you have specified for the source drive is assigned to a hard disk. CLONE only works with floppy drives. Check the command line to make sure you entered the drive correctly.

**>>> Error: Destination drive is a hard disk drive**

(Command Line Mode) The drive you have specified for the destination drive is assigned to a hard disk. CLONE only works with floppy drives. Check the command line to make sure you entered the drive correctly.

**>>> Error: Bad output designation**

(Command Line Mode) CLONE has detected something wrong with one of the outputs you have specified for the console output. Check to see that you are using the proper form of the output direction instruction.

**>>> Error: Output-file drive same as source drive**

(Command Line Mode) You cannot specify the (optional) file for console output to be on the same drive that is used for the source of the copy operation.

**>>> Error: Output-file drive same as destination drive**

(Command Line Mode) You cannot specify the (optional) file for console output to be on the same drive that is used for the destination of the copy operation.

**>>> Error: Output-file cannot be a floppy on a single-floppy system.**

(Command Line Mode) If you have a single floppy system, you will be required to swap the source and destination diskettes on the floppy drive as CLONE makes the copy. In this case it is unreasonable to direct console output to a disk file on yet a third floppy. If you have a hard disk on your system, you may direct console output to a disk file on it otherwise you cannot direct console output to a disk file.

**>>> Error: Output-file drive not on system**

(Command Line Mode) The disk drive you specified for the output disk file does not exist on the system. Check the command line you used to insure that it is correct.

**>>> Error: Bad output-file name**

(Command Line Mode) Something is wrong with the output file you specified to receive console output. This can be caused by invalid characters in the file name or by specifying a drive for the file with an invalid character.

**8.8 Utility name: DATIME****Purpose:** To display the current system date and time.**General Description**

DATIME displays the current system date and time. The date is displayed as day of week, month, day, and year. The time is displayed in hours, minutes and seconds in a 24 hour format.

DATIME allows you to direct its output to the system printer and/or a disk file. This feature can be useful for time stamping printed output, dating backup copies of files, etc.

**Using DATIME**

Figure 8.24 shows the general form of the command line to execute DATIME. It is not necessary that any output destinations be specified on the command line. If none are specified, the output defaults to the console.

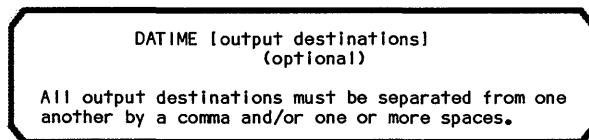


Figure 8.24 General Form of DATIME Command Line

Figure 8.25 shows the console display which results from running DATIME with no output destinations specified.

```
1: A>DATIME<enter>
2:
3:
4: System date: Thursday May 12, 1983 (5/12/83)
5: System time: 10:09:05
6:
7:
8: A>
```

Figure 8.25 Example of Using DATIME

The output from DATIME may be directed to the system console, the system printer, a disk file, or any combination of these. If you specify any destinations, they will be the only destinations for the output. For example, if you specify /LST: to send the output to the printer, no output will appear on the console. If you want output on both the console and the printer you must specify both /LST: and /CON:. The special destination /NUL: is provided in the (unlikely) event that you want no output at all from the program. If /NUL: is specified with any other destinations, the other destinations will supersede /NUL:. Figure 8.26 shows the various command line instructions that can be used to direct the output of DATIME.

/CON:	sends output to the console (default if no output instructions are given).
/LST:	sends output to system printer.
/(valid file name)	sends output to the specified file. If a file with the same name already exists, it is replaced.
/(valid file name)+	appends output to the specified file. If the file does not already exist, it is created. The + may be preceded by a space or comma.
/NUL: or /NULL:	causes no output to be generated. If any other output instructions appear, they will supersede /NUL:.

Figure 8.26 Instructions to Direct Console Output

The following examples illustrate using DATIME in the command mode:

**DATIME /LST:**

Prints the system date and time on the system printer.

**DATIME /CON: /A:JOBLOG**

Displays the system date and time on the system console and puts it in a file named JOBLOG on drive A.

**DATIME /A:JOBLOG +**

Appends the current system date and time onto the file named JOBLOG on drive A.

### Possible Error Messages

**>>> Error: Bad output designation**

This message indicates that an instruction you gave on the command line to redirect the output from DATIME was not valid. This can be caused by ending the command line with a slash (/).

**8.9 Utility name: DDT**

**Purpose:** An interactive debugging tool for programs running under CP/M 2.

### General Description

DDT is an interactive debugging tool for testing and debugging programs written for the CP/M 2 operating system. It is usually used on assembly language programs, but it is also effective for programs compiled from a high level language. In addition, it provides the ability to apply patches to programs and disk files.

DDT is written for the 8080 instruction set only. It can be used with programs that make use of Z-80 instructions, but certain precautions must be taken for successful operation.

The discussion of DDT presented in this manual is of a summary nature. For further information see Chapter 4 of the CP/M Operating System Manual

### Using DDT

**Note:** DDT displays and accepts numbers exclusively in hexadecimal representation. All numbers shown in this section should be interpreted as hexadecimal unless explicitly stated to be otherwise.

When DDT is executed, it moves itself into the high area of the transient program area (TPA) just below the beginning of BDOS and overlaying the console command processor (CCP). The 4 Kbytes of memory required by the core of DDT are removed from the available TPA when the routine is loaded.

Several commands (A, L, T, and X) require DDT to use additional memory from the TPA. Following any one of these, DDT will remove an additional 275 Kbytes from the TPA. If a program is loaded which uses some of the additional space required for these functions, A and L will not be available, and T and X will not display instructions in mnemonic form.

The command line for the execution of DDT may contain a file name to be loaded. DDT will load any file specified into memory before prompting for commands. If the file name has the extension HEX, DDT expects to find a standard Intel HEX file and will attempt to read it as such. HEX files are loaded into memory at the addresses specified within the file unless specific commands are given to DDT to load it at another address. Files with other extensions are loaded starting at 100h unless DDT is instructed otherwise. Figure 8.27 shows the command line that executes DDT and causes it to load SUBMIT.COM into memory.

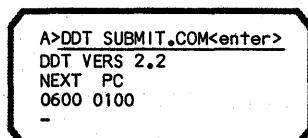


Figure 8.27 Running DDT

After a file is loaded, DDT displays two numbers. NEXT is the address of the next memory location following the loaded file, indicating where free memory begins. PC is the current contents of the program counter. It is the memory location at which execution will begin if the G command is given. PC will typically be set to 100h when a file is loaded unless the file was a HEX file that specified an execution address.

Figure 8.28 shows a summary of the DDT commands. Arguments for each command are shown in parentheses ( ) if they are required and in brackets [ ] if they are optional. The first argument must immediately follow the command character. Any additional arguments must be preceded and followed by one space or one comma. For some commands, the first argument may be omitted; in this case, a comma must immediately follow the command letter to indicate the omission.

command	function
A(x)	Assemble assembly language instructions entered from the console and place in memory beginning at location x.
D[x],[y]	Display contents of memory from x to y in hex and ASCII form.
F(x),(y),(z)	Fill memory from location x to location y with byte z.
G[x],[y],[z]	Begin execution at location x with breakpoints at y and z.
H(x),(y)	Display x-y and x-y as hexadecimal numbers.
I[filename.ext]	Initialize DDT to read the file specified.
L[x],[y]	List the instructions from location x to location y in mnemonic form.
M(x),(y),(z)	Move the block of memory beginning at location x and ending at location y to memory beginning with location z.
R[x]	Read a previously initialized file into memory with offset x.
S(x)	Examine and optionally alter the contents of memory starting at location x.
T[x]	Trace execution for x instructions.
U[x]	Execute x instructions and then return control to DDT.
X[x]	Examine and optionally alter machine registers.

Figure 8.28 DDT Command Summary

### D[x],[y] command

The D command is used to display a section of memory on the console. The memory is displayed in both hexadecimal and ASCII form with 16 bytes of memory per line, as shown in Figure 8.29. The D command can have 0, 1, or 2 arguments. If no arguments are given, 12 lines of 16 bytes are displayed starting at the current display address. The display address is initialized to the contents of the program counter but is advanced by the D command so that successive D commands will display successive sections of memory. If a single argument is given with the D command, it indicates the address in memory at which to begin the display. A second argument (if present) indicates the last address in memory to be displayed. If the first argument is omitted, memory is displayed from the current display address up to the address given by the second argument.

```
-D<enter>
0100 C3 DF 01 20 63 6F 70 79 72 69 67 68 74 28 63 29 ... copyright(c)
0110 20 31 39 37 37 2C 20 64 69 67 69 74 61 6C 20 72 1977, digital r
0120 65 73 65 61 72 63 68 20 0D 0A 24 45 72 72 6F 72 esearch ..$Error
0130 20 4F 6E 20 4C 69 6E 65 20 24 53 55 42 4E 6F 20 On Line $SUBN0
0140 27 53 55 42 27 20 46 69 6C 65 20 50 72 65 73 65 'SUB1 File Prese
0150 6E 74 24 44 69 73 6B 20 57 72 69 74 65 20 45 72 nt$Disk Write Er
0160 72 6F 72 24 43 6F 6D 6D 61 6E 64 20 42 75 66 66 ror$Command Buff
0170 65 72 20 4F 76 65 72 66 6C 6F 77 24 43 6F 6D 6D er Overflow$Comm
0180 61 6E 64 20 54 6F 6F 20 4C 6F 6E 67 24 50 61 72 and Too Long$Par
0190 61 6D 65 74 65 72 20 45 72 72 6F 72 24 49 6E 76 ameter Error$Inv
01A0 61 6C 69 64 20 43 6F 6E 74 72 6F 6C 20 43 68 61 alid Control Cha
01B0 72 61 63 74 65 72 24 44 69 72 65 63 74 6F 72 79 racter$Directory
-
```

Figure 8.29 Example of Using the D Command of DDT

**L[x],[y] command**

DDT can also display the contents of memory as 8080 instructions if the L command is used. L can have 0, 1, or 2 arguments. If no arguments are given, 12 lines of 8080 instructions will be displayed from memory starting from the current list/assembly address. The list/assembly address is initially set equal to the program counter but is modified by the L and A commands. If arguments are present, they specify the beginning and ending addresses of memory to be listed as described for the D command. Figure 8.30 shows an example of the L command. Note that the first instruction listed is a jump which is followed by ASCII text. L is used again to display the listing at the address to which the jump is made. If the first argument is omitted, the contents of memory are listed from the current list/assembly address up to the address given by the second argument.

```
-L100<enter>
0100 JMP 01DF
0103 ??= 20
0104 MOV H,E
0105 MOV L,A
0106 MOV M,B
0107 MOV A,C
0108 MOV M,D
0109 MOV L,C
010A MOV H,A
010B MOV L,B
010C MOV M,H
-L1DF<enter>
01DF LXI H,0000
01E2 DAD SP
01E3 SHLD 05F0
01E6 LXI H,0E93
01E9 SPHL
01EA CALL 02CC
01ED CALL 038A
01F0 CALL 04FE
01F3 CALL 0587
01F6 RET
01F7 LXI H,05DD
```

Figure 8.30 Example of Using the L Command of DDT

**A(x) command**

The A command allows you to type in 8080 assembly language statements which will be converted to machine code. They are placed in memory at the address specified by the argument in the command. The address at which the code is stored is

advanced after each instruction so that successive instructions can be entered.  
(Note: If the L command is used after the A command, the listing produced will begin at the address where the next instruction would have been stored.) To return to the command mode of DDT, merely press <enter> instead of entering an assembly language instruction. Figure 8.31 gives an example of the A command.

```
-A800<enter>
0800 MVI A,12<enter>
0802 STA 810<enter>
0805 RST 7<enter>
0806 <enter>
-
```

Figure 8.31 Example of using the A command of DDT

### S(x) command

The S command can be used to examine and modify single bytes in memory. It displays the memory address given by the argument and the contents of memory at that address. You may enter a new value to be stored at that location, or you may advance to the next location by pressing <enter>. To return to the command mode of DDT, type a period (.) followed by <enter>. In the example of the S command is shown in Figure 8.32, the 18h at location 301h is replaced by 20h.

```
A>S300<enter>
0300 C2 <enter>
0301 18 20<enter>
0302 05 4F<enter>
0303 0D <enter>
-
```

Figure 8.32 Example of using the S command of DDT

### G[x],[y],[z] command

The G command causes execution of the instructions at certain locations in memory. 0, 1, 2, or 3 arguments are accepted. If no arguments are specified, execution is started at the address specified by the program counter. The first argument to the G command specifies the address in memory at which to begin execution. The second and third arguments, if given, specify break point locations. Break points are RST 7 instructions which are placed over the contents of memory at the address given. If the address of a break point is encountered during execution, the RST 7 instruction will cause control to be returned to DDT. (Note: Break points can also be inserted manually into the program; see Figure 8.31 for an example.)

If you wish, you may specify break points after omitting the first argument; this will cause execution of the instructions at the address given by the program counter. Break points or RST 7 instructions are the only means of returning control to DDT once the G command is given. When DDT does regain control, it displays an asterisk followed by the address at which the break point was encountered, and it clears any break points specified in the G command. Figure 8.33 shows the use of the G command to execute the code assembled in Figure 8.31.

```
-G800<enter>
*0805
-
```

Figure 8.33 Example of using the G command of DDT

**X[x] command**

The X command allows you to examine and modify the contents of the 8080 registers. If the X command is given with no argument, all of the registers and flags are displayed. In addition, the contents of the memory location given by the program counter is displayed in mnemonic form. An argument in the command specifies one of the registers or flags as the only one to be displayed. In this case, its contents may be modified by the user. Figure 8.34 shows the various flags and registers that may be displayed and modified with the X command, and Figure 8.35 gives an example of its use. In Figure 8.35, all of the registers are displayed, the program counter is modified (perhaps in preparation for using the G command), and the registers are redisplayed.

<u>designation</u>	<u>meaning</u>	<u>allowed values</u>
C	carry flag	0 - 1
Z	zero flag	0 - 1
M	sign flag	0 - 1
E	parity flag	0 - 1
I	half carry flag	0 - 1
A	accumulator	0 - FFF
B	BC register	0 - FFFF
D	DE register	0 - FFFF
H	HL register	0 - FFFF
S	stack pointer	0 - FFFF
P	program counter	0 - FFFF

Figure 8.34 Register and Flag Designations for the X Command of DDT

```
-X<enter>
COZOMOE010 A=12 B=C3C5 D=2446 H=4624 S=0126 P=8000 NOP
-X<enter>
P=8000 100<enter>
-X<enter>
COZOMOE010 A=12 B=C3C5 D=2446 H=4624 S=0126 P=0100 JMP 01DF
-
```

Figure 8.35 Example of Using the X Command of DDT

8

**F(x),(y),(z) command**

The F command of DDT fills a section of memory with a particular byte. Three arguments are required. The first two specify, respectively, the beginning and ending addresses of the section of memory to be filled. The third gives the byte to be used. In Figure 8.36 the F command is used to fill memory from 100h up to and including 8000h with the byte "00h".

```
-F100 8000 0<enter>
-
```

Figure 8.36 Example of Using the F Command of DDT

**M(x),(y),(z) command**

The M command of DDT is used to copy the contents of one section of memory to another. It also requires three arguments. The first two are, respectively, the beginning and ending addresses of the section of memory to be copied. The contents of that section are copied byte-by-byte into successive memory locations starting at the address given by the third argument. Figure 8.37 shows the use of the M command to copy the 512 (decimal) bytes of memory from 100h through 2FFh to 8000h through 81FFh.

```
-M100 2FF 8000<enter>
```

Figure 8.37 Example of Using the M Command of DDT

**T[x] and U[x] commands**

The T and U commands allow you to execute a program while retaining control from DDT. The T command displays the machine registers and instructions as they are executed, while the U command displays nothing at all. With the T command, the registers and machine instructions are displayed together before execution of the instructions begins. During T or U commands, you may regain control of the computer by typing a DEL character (<ctrl-minus>). Figure 8.38 shows the use of both commands. Note the action of DDT when it regains control: it displays an asterisk followed by the address of the next instruction in memory to be executed.

```
-T5<enter>
COZOM0E010 A=12 B=C3C5 D=2446 H=4624 S=0126 P=0100 JMP 01DF
COZOM0E010 A=12 B=C3C5 D=2446 H=4624 S=0126 P=01DF LXI H,0000
COZOM0E010 A=12 B=C3C5 D=2446 H=0000 S=0126 P=01E2 DAD SP
COZOM0E010 A=12 B=C3C5 D=2446 H=0126 S=0126 P=01E3 SHLD 05F0
COZOM0E010 A=12 B=C3C5 D=2446 H=0126 S=0126 P=01E6 LXI H,0E93*01E9
-U5<enter>
COZOM0E010 A=12 B=0081 D=2446 H=0E93 S=0E8F P=02D0 MV1 E,7F*02D2
```

Figure 8.38 Example of Using the T and U Commands of DDT

**I(filename.ext) and R[x] commands**

It is possible to load disk files into memory for examination while DDT retains control. A two-step procedure using the I and R commands of DDT is required.

First, the I command inserts a file name into the system default file control block at 5Ch. The command may also be used to initialize the default file control block for a program being tested. This action is very similar to that taken by the console command processor (CCP) in loading and executing a COM file. Unlike the CCP, however, the I command can accept only one file name and does not set up a command line at location 80h.

Next, the R command is used to read the file into memory. If the file name specified in the I command has the extension HEX, DDT tries to read it as an Intel hex format file. The hex file is converted to binary and is loaded into memory at the addresses specified in the hex format. Files with other file name extensions are loaded at 100h.

An optional argument in the R command specifies an offset to be added to the address at which a file would normally be loaded. That number is added to the

normal load address, and any overflow past the 16th bit is ignored. Using an appropriate argument, a file can be loaded to any location in memory. For example, to load a HEX file whose internal load address is 8000h into memory starting at 9180h, you would give an offset of 1180h. To load the same file into memory at 100h, the needed argument is 8100h.

Figure 8.39 shows examples of the I and R commands.

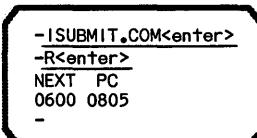


Figure 8.39 Example of Using the I and R commands of DDT

### H(x),(y) command

DDT provides a simple hexadecimal calculator mode for computing offsets and other hex numbers. Two numbers to be added and subtracted are entered as the two required arguments in the command. DDT then displays their sum and difference (argument 1 minus argument 2). All carries and borrows beyond the 16th bit are ignored. Figure 8.40 shows the H command used to compute the sum and difference of EC00h and 100h.

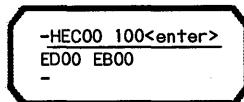


Figure 8.40 Example of Using the H Command of DDT

Since DDT is designed for the 8080 only, care must be taken in running it with Z-80 programs. In particular, if breakpoints are used in the G command, it is essential that one does not fall on a Z-80 instruction. It is not possible to use a T and U command with Z-80 instructions, so the command must be stopped before a Z-80 instruction is encountered. The G command could then be used to proceed past the Z-80 code. Also, the L command will not be able to decipher the Z-80 instructions and may be confused by them.

### Possible Error Messages

DDT has only one error indicator. If it cannot decode a command or if the command cannot be executed, it displays a question mark on the console.

**8.10 Utility name: DENSITY**

**Purpose:** To display and optionally alter the density flag on a diskette.

**General Description**

Whenever a floppy diskette is formatted by the P&T CP/M 2 FORMAT utility, a density flag is set. This flag is then used by the operating system in determining the density at which the diskette was formatted. The DENSITY program allows you to inspect the density flag and modify it if necessary.

Normally, there is no need to alter the density flag on a diskette. The FORMAT routine automatically sets the flag to agree with the actual formatting procedure. However, it is possible that a diskette formatted by another system will be inadvertently flagged at the wrong density. Since most interchange between systems takes place on single density diskettes, the most likely error is a single density diskette flagged as double density or double sided. (Even this case should be extremely rare.) If you are having trouble reading a diskette and you are certain of its actual density, you should use the DENSITY program to check the flag. If it is set incorrectly, DENSITY can reset it according to your instructions.

**Using DENSITY**

A sample running of the DENSITY program is shown in Figure 8.41. The program is executed by the command on Line 8.41-1. After displaying its opening message (Lines 8.41-4 to 8.41-6), it reminds you that pressing the <break> key will terminate the program (Line 8.41-8).

On Line 8.41-10, DENSITY requests the drive on which the test is to take place. Note that you may specify the drive by either a logical drive letter or a physical drive number. After you enter the drive specification (drive B in this case), DENSITY checks that the drive is actually on the system and is a floppy drive. If it is not on the system or is not a floppy drive, an error message will be displayed and you will be asked to specify the drive again. If the specified drive is a valid floppy drive, DENSITY will display the correspondence between the logical and physical drive designations as shown on Line 8.41-11.

Next, you are asked to mount the diskette to be tested (Line 8.41-13). At this time the diskette in the selected drive may be changed if desired.

After you press <enter> (Line 8.41-13), DENSITY will read the density flag and report its value. In our example, a double density value is found (Line 8.41-15). You will then be asked if you want to change the flag (Line 8.41-17). If you answer negatively, DENSITY will start over, allowing you to check the flag on another diskette. If you answer affirmatively, as in this example, you will be asked to specify the new density setting (Line 8.41-19). After you enter the type of flag, DENSITY will modify it accordingly and start over.

```
1: A>DENSITY<enter>
2:
3:
4: Disk density test routine. Ver 2,xxx
5: Copyright 1980,82,83 by PICKLES & TROUT
6: All rights reserved
7:
8: Hit BREAK to quit.
9:
10: Enter drive on which to test (0-3 or A-P) : B<enter>
11:     NOTE: Logical drive B: is the same as physical drive (1).
12:
13: Mount disk to test on drive B and hit ENTER when ready : <enter>
14:
15: The disk is tagged double density
16:
17: Do you want to change the disk tag? Y<enter>
18:
19: Tag disk as Single or Double density or 2-sided double density (S,D,2)? : S<enter>
20:
21:
22: Disk density test routine. Ver 2,xxx
23: Copyright 1980,82,83 by PICKLES & TROUT
24: All rights reserved
25:
26: Hit BREAK to quit.
27:
28: Enter drive on which to test (0-3 or A-P) : B<enter>
29:     NOTE: Logical drive B: is the same as physical drive (1).
30:
31: Mount disk to test on drive B and hit ENTER when ready : <enter>
32:
33: The disk is tagged single density
34:
35: Do you want to change the disk tag? Y<enter>
36:
37: Tag disk as Single or Double density or 2-sided double density (S,D,2)? : D<enter>
38:
39:
40: Disk density test routine. Ver 2,xxx
41: Copyright 1980,82,83 by PICKLES & TROUT
42: All rights reserved
43:
44: Hit BREAK to quit.
45:
46: Enter drive on which to test (0-3 or A-P) : <break>
47: A>
```

Figure 8.41 Example Console Dialog for Using DENSITY

In this example, the flag is changed to single density and then back to double density (Lines 8.41-22 to 8.41-37). After you have finished checking and possibly changing density flags, you should press <break> when asked for a drive specification as shown on Line 8.41-46. DENSITY will then return you to the command level of the system. If your system requires a system diskette for a warm boot, DENSITY will prompt you to mount a system diskette before returning to the system.

### Possible Error Messages

#### Bad drive name, please re-enter (0-3 or A-P) :

The letter you entered to specify the logical drive on which to test was not in the range A - P. You should re-specify the drive using a valid logical drive letter or a physical drive number.

**Bad drive number, please re-enter (0-3 or A-P) :**

The number you entered to specify the physical drive on which to test was not in the range 0 - 3. You should re-specify the drive using a valid physical drive number or a logical drive letter.

**That drive is not on the system, please re-specify :**

You have specified a valid drive designation but the drive is not available on your system. For example, you might have specified drive D on a two-drive system. Re-enter a drive designation using one of the drives that is available on your system.

**That drive is a hard disk drive, please re-specify :**

The drive you have specified is a hard disk drive. DENSITY can work only on floppy disk drives. You should re-enter a drive designation for one of the floppy drives on the system. If you do not know which logical drives are assigned to the physical floppy drives on the system you can specify a physical drive number.

**Please re-enter specifying D for double density, S for single density,  
or 2 for 2-sided double density:**

You have not entered a valid selection for the new density flag after electing to change it. Re-enter your selection using one of the three characters D, S, or 2.

**BDOS error while reading first sector on track 0.**

DENSITY could not read the sector on the diskette that contains the density flag. This may indicate that the diskette is not formatted on track 0, is incorrectly formatted on track 0 (ie. it might have been formatted by another operating system), or track 0 has been damaged in some way. All diskettes used with P&T CP/M 2 must be formatted at single density on track 0 of side 0 regardless of the density at which the rest of the diskette is formatted.

**8.11 Utility name: DISKCHK**

Purpose: To perform a non-destructive test for flaws on a diskette.

**General Description**

DISKCHK performs a non-destructive test of the information recorded on a diskette. It reads all of the sectors on each track of the diskette and reports any errors that occur. This test is by no means comprehensive; it is intended merely as a quick check on the current state of a diskette. However, it is a good idea to replace a diskette if DISKCHK starts to report a significant number of soft errors. Frequently, the errors can be eliminated by erasing and reformatting the diskette, but, of course, all data on it would then be lost.

Figure 8.42 shows how long it takes to perform a check on each format of diskette.

format	time
single density	19 sec
double density	19 sec
double sided	33 sec

Figure 8.42 Testing Times for DISKCHK

The command line mode of DISKCHK allows you to execute it from a SUBMIT file or a menu program. This will allow you to embed DISKCHK in an integrated system of programs and execute it several times a day. Since DISKCHK takes relatively little time, the user of the system need not even know that DISKCHK is being used. Such a system could automatically keep track of the state of each diskette and suggest that the user make backups should soft errors begin to occur. See the examples at the end of this chapter for a simple approach to doing this.

The DISKCHK test is very simple, and it may miss problems that occur in normal (more complicated) usage. Remember to keep adequate backups at all times, even if DISKCHK reports no errors on the diskette.

**Using DISKCHK - Interactive Mode**

Figure 8.43 shows a typical console dialog that might take place when using DISKCHK.

DISKCHK is executed by the command line shown on Line 8.43-1. After displaying its sign-on messages (Lines 8.43-4 to 8.43-6), DISKCHK reminds you that it performs a non-destructive test and that you may press the <break> key at any time to abort the program and return to the operating system (Lines 8.43-9 and 8.43-10).

DISKCHK then asks you for the drive on which the test is to be performed (Line 8.43-12). Note that you may specify the drive as either a physical drive number or a logical drive letter. After you have entered the drive (B in this example), DISKCHK checks that it is a valid floppy drive on the system. It then shows you the correspondence between the logical drive letter and the physical drive number (Line 8.43-13). It is not necessary that the diskette to be tested be mounted on the drive at this time. If the drive you specify is not a floppy drive or is not defined

for the system DISKCHK will display an error message and ask you again for the drive.

```
1: A>DISKCHK<enter>
2:
3:
4:          1D/2D/2S Disk Check Utility - Ver 2.xxx
5:          copyright (C) 1980,82,83 Pickles & Trout
6:          all rights reserved
7:
8:
9:          This is a non-destructive disk check routine.
10:         Press <BREAK> at any time to quit.
11:
12: Enter drive to check (A-P or 0-3) : B<enter>
13:     NOTE: Logical drive B: is the same as physical drive (1).
14:
15: Mount disk to check on drive B: and press <ENTER> when ready to start : <enter>
16:
17: Checking Double-density disk in drive B:
18: .....
19:
20: Checking complete.
21:
22:
23:          This is a non-destructive disk check routine.
24:          Press <BREAK> at any time to quit.
25:
26: Enter drive to check (A-P or 0-3) or (R) to repeat : <break>
27:
28: A>
```

Figure 8.43 Example Console Dialog for Using DISKCHK

After you have specified the drive, DISKCHK asks you to mount the diskette to be tested on the drive and press the <enter> key (Line 8.43-15). At this time you may change diskettes, if you wish, before pressing <enter>. After you press <enter> the test will begin.

As the test begins, DISKCHK checks the density of the diskette and reports it as shown on Line 8.43-17. As the test proceeds a dot is displayed on the console display for each track tested (Line 8.43-18). Note: if a double sided diskette is being tested, DISKCHK first checks side 0 and then side 1 at the same head location (cylinder). It displays a dot as side 0 is tested and then replaces it with a colon (:) as side 1 is tested. When the test of a double sided diskette is completed, a row of colons will appear on Line 8.43-18 instead of dots.

After the test is completed, DISKCHK starts over again as shown in Lines 8.43-23 to 8.43-26. Note that the prompt for the drive is slightly different now. You have the option of entering "R" to repeat exactly the same test again. This is useful when you want to test several diskettes. DISKCHK begins the test immediately after you enter an "R" on Line 8.43-26 so you should mount the diskette to be tested before entering the "R". With the "R" option you can easily test several diskettes by merely mounting them on the drive you initially specified and repeating the test.

When you have finished testing, press the <break> key when asked for a drive, as shown on Line 8.43-26. Pressing the <break> key at other times during the test will cause DISKCHK to abandon the test in progress and return to the beginning (it does not return to the system).

For systems that require a system diskette for warm booting, DISKCHK will prompt you to mount a system diskette on the system drive before returning to the system

command level. If you encounter this prompt, insure that a system diskette is mounted on the system drive and press <enter>.

### Using DISKCHK - Command Line Mode

In the command line mode of DISKCHK, you must specify all of the information regarding the test you want performed on the command line that you use to execute DISKCHK. Note that in the command line mode, the test normally begins automatically as soon as DISKCHK begins execution (except on a single floppy drive system). You should insure that the diskette to be tested is mounted before executing DISKCHK in the command line mode. (See the PMOUNT instruction below for an exception to this.)

The general form of the DISKCHK command line is shown in Figure 8.44.

```
DISKCHK [drive to test] [output destinations] [options]
          (required)           (optional)        (optional)
```

All instructions on the command line must be separated from one another by a comma and/or one or more spaces.

Figure 8.44 General Form of DISKCHK Command Line

You must specify the drive on which the test is to be made when using the command line option of DISKCHK. You may use either a physical drive number or a logical drive letter when specifying the drives. The drive is specified using an expression like those shown in Figure 8.45. Note that the drive designation may come before or after the equal sign.

```
DRIVE=[drive]      DRV=[drive]      DR=[drive]
[drive]=DRIVE     [drive]=DRV       [drive]=DR
```

[drive] should be replaced by a logical drive letter or a physical drive number.

Figure 8.45 Specifying the Drive on Which to Test

The command line mode of DISKCHK allows you to direct the console output from the program to the console, system printer, or a disk file. The output defaults to the console if no output direction is specified. You may direct the output to any combination of these three destinations. Any destination(s) you specify will be the only destination(s) for the console output. For example, if you specify a disk file for the destination, no output will appear on the console. If you want output to both a disk file and the console you must specify both as output destinations. A special output destination (NUL:) is provided for cases where you want no output at all from the program. An output destination is specified by a slash ( / ) followed by the destination name. Examples are given in Figure 8.46.

/CON:	sends output to the console (default if no output instructions are given).
/LST:	sends output to system printer.
/(valid file name)	sends output to the specified file. If a file with the same name already exists, it is replaced.
/(valid file name)+	appends output to the specified file. If the file does not already exist, it is created. The + may be preceded by a space or comma.
/NUL: or /NULL:	causes no output to be generated. If any other output instructions appear, they will supersede /NUL:.

Figure 8.46 Instructions to Direct Console Output

There are several additional instructions that can appear on the command line of DISKCHK. They are listed in Figure 8.47. In the command line mode, DISKCHK normally does not prompt for disks to be mounted; it immediately begins the test. When it is finished, it does not prompt for a system diskette to be mounted either.

If you wish for DISKCHK to prompt for a diskette to be mounted, you may use the "PMOUNT" instruction (or one of its synonyms) on the command line. This can be useful when DISKCHK is used from a SUBMIT file because it eliminates the need for a separate program to prompt for a diskette to be mounted. If you want DISKCHK to prompt for a system diskette at the end of the test you should include the "PSYS" instruction on the command line. This instruction causes DISKCHK to issue a prompt for the system diskette if one is required for a warm boot operation. No prompt will be issued for systems that do not require a system diskette for a warm boot. You may instruct DISKCHK to prompt both for a diskette to test and for the system diskette by using the "PBOTH" instruction on the command line.

DISKCHK will always issue prompts for a diskette to test and for the system diskette (if one is needed for a warm boot) if the system has only a single floppy drive and no hard disk drives.

PMOUNT or PM	causes DISKCHK to prompt for a diskette to be mounted before beginning the test.
PSYS or PS	causes DISKCHK to prompt for a system diskette to be mounted after the test is completed.
PBOTH or PB	causes DISKCHK to prompt both a diskette to be tested and for a system diskette after the test is completed.

Figure 8.47 Additional Command Line Instructions for DISKCHK

The following examples show command lines that could be used with DISKCHK:

**DISKCHK DRIVE=B**

Tests the diskette mounted on drive B. The diskette is assumed to be mounted already and the test will begin immediately. Messages are sent to the console. DISKCHK will not prompt for a system disk after the test.

**DISKCHK DR=C PSYS**

Tests the diskette mounted on drive C. The diskette is assumed to be mounted already and the test will begin immediately. Messages are sent to the console. DISKCHK will prompt for a system disk after the test if one is required for a warm boot.

**DISKCHK DR=B,/A:LOG,PMOUNT**

Tests the diskette on drive B. DISKCHK will prompt for the diskette to be mounted before the test begins. Messages are sent to a disk file named LOG on drive A. DISKCHK will not prompt for a system disk after the test.

**DISKCHK DRV=B,/CON:/,LST:**

Tests the diskette on drive B. The diskette is assumed to be mounted already and the test will begin immediately. Messages are sent to the console and the system printer. DISKCHK will not prompt for a system disk after the test.

**DISKCHK DR=C,/LOG+**

Tests the diskette on drive C. The diskette is assumed to be mounted already and the test will begin immediately. Messages are appended to a file named LOG on the current default drive. DISKCHK will not prompt for a system disk after the test.

### Possible Error Messages

**Non-modular system, not compatible with this program.**

You are trying to run this program with a previous version of P&T CP/M 2. The program makes use of features found only in the modular version of P&T CP/M 2 and hence cannot run on this system.

8

**Bad drive name, please re-enter (0-3 or A-P) :**

The letter you entered to specify the logical drive on which to test was not in the range A - P. You should respecify the drive using a valid logical drive letter or a physical drive number.

**>>> Error: Bad drive name**

Same meaning as the previous message but given in the command line mode only.

**Bad drive number, please re-enter (0-3 or A-P) :**

The number you entered to specify the physical drive on which to test was not in the range 0 - 3. You should respecify the drive using a valid physical drive number or a logical drive letter.

**>>> Error: Bad drive number**

Same meaning as the previous message but given in the command line mode only.

**That drive is not on the system, please re-specify:**

You have specified a valid drive designation but the drive is not available on your system. For example, you might have specified drive D on a two-drive system. Re-enter a drive designation using one of the drives that is available on your system.

**>>> Error: Drive not on system**

Same meaning as the previous message but given in the command line mode only.

**That drive is a hard disk drive, please re-specify:**

The drive you have specified is a hard disk drive. DENSITY can work only on floppy disk drives. You should re-enter a drive designation for one of the floppy drives on the system. If you do not know which logical drives are assigned to the physical floppy drives on the system you can specify a physical drive number.

**>>> Error: Test target drive is a hard disk drive**

Same meaning as the previous message but given in the command line mode only.

**>>> Error: Missing drive name**

Given in the command line mode if the program could not find a specification on the command line of the drive on which to perform the test. Recheck the command line you used to be sure you specified the drive.

**>>> Error: Bad output designation**

Given in the command line mode if you attempted to redirect the console output but the program could not figure out where you wanted it sent. This can be caused by an illegal character in a file name or by ending the command line with a slash (/).

**>>> Error: Output-file drive same as test target drive**

Given in the command line mode if you directed the console output to a disk file on the same drive that is being tested. This is not allowed because it interferes with the test.

**>>> Error: Bad output-file drive name**

Given in the command line mode if you directed console output to a disk file and used an illegal character (ie. not A - P) to specify the drive on which to place the file.

**>>> Error: Output-file drive is not on system**

Given in the command line mode if you directed console output to a disk file on a drive that is not present on the system. Recheck the command line you used to make sure you specified the correct drive.

**>>> Error: Output-file drive cannot be a floppy on a single-floppy system**

Given in the command line mode if you directed console output to a disk file on a floppy diskette on a system that has a single floppy drive. This is not allowed even if disk swapping is in effect to make the single floppy drive appear as if it were multiple drives. Swapping diskettes during the test would not only be tedious but also would interfere with the test.

**>>> Error: Drive X: is not ready.**

When the program started the test, it found that the drive specified for the test was not ready. This is usually caused by not having a diskette mounted when the test begins. The "X" will be replaced by the logical drive letter of the drive that was found not to be ready.

**>>> Error: Home error.**

A disk error occurred when the program attempted to restore the read/write head to the home position. This can be caused by an unformatted or improperly formatted diskette or an improper drive step rate.

**>>> Error: Seek error.**

A disk error occurred when the program was moving the read/write head to a particular location. This can be caused by an unformatted or improperly formatted diskette or an improper drive step rate.

**>>> Error: Unable to determine density of disk.**

The program was unable to determine the density of the diskette being formatted. Although it is rare to see this error, it can be caused by an unformatted or improperly formatted diskette.

**>>> Error: Check aborted with <BREAK>.**

This message is issued if you abort the test by pressing the <break> key while the test is running.

**>>> Error: xx Soft error(s): Track yy, Sector zz**

This message is given when soft errors are encountered. Whenever a sector on the disk cannot be read successfully, the program will attempt to read it again. If a subsequent attempt is successful, each previous failure is called a soft error. In the error message, "xx" would be replaced by the number of times the attempt failed before a successful read was performed. "yy" and "zz" would be replaced by the track and sector numbers, respectively, where the soft errors occurred.

**>>> Error: xx Soft error(s): Cylinder yy, Side s, Sector zz**

Same meaning as the previous message except that it is used when a double sided diskette is being tested. In the error message, "xx" would be replaced by the number of times the attempt failed before a successful read was performed. "yy", "s", and "zz" would be replaced by the cylinder, side and sector numbers, respectively, where the soft errors occurred.

**>>> Error: Permanent error: Track yy, Sector zz**

A permanent error has been encountered. A permanent error is a failure to read a sector on the disk successfully in 10 consecutive attempts. The message reports the track and sector number at which the error took place.

8

**>>> Error: Permanent error: Cylinder yy, Side s, Sector zz**

Same meaning as the previous message except that it is used when a double sided diskette is being tested. In the error message, "yy", "s", and "zz" would be replaced by the cylinder, side and sector numbers, respectively, where the permanent error was found.

**>>> Error: Time-out error: Track yy, Sector zz**

In some cases, when reading a diskette, if the desired sector cannot be found, the disk controller circuit will try to locate it on several successive revolutions of the diskette before reporting an error. If the sector could not be found on the very first attempt, a problem could exist on the diskette. This message is given if too much time elapses from when the "read" command is given to the disk controller to when the sector was actually read. This indicates that the controller required more than one revolution to find the sector meaning that the sector was missed at least once. This may indicate the very early stages of degradation of a sector.

**>>> Error: Time-out error: Cylinder yy, Side s, Sector zz**

Same meaning as the previous message except that it is used when a double sided diskette is being tested. In the error message, "yy", "s", and "zz" would be replaced by the cylinder, side and sector numbers, respectively, where the error was found.

**8.12 Utility name: DISKTEST**

**Purpose:** To test a diskette for flaws. (Note: this test destroys any data on the diskette.)

**General Description**

DISKTEST performs a media test on diskettes. The complete test writes and then reads back three different data patterns to each of the sectors on each track of the diskette and reports any errors that occur. You may optionally select a short test which writes and reads back only one data pattern. The single data pattern used is a "worst case" pattern for double density formats.

DISKTEST is intended as a check of the media prior to use and before reuse.  
**NOTE: DISKTEST WILL ERASE ALL INFORMATION FROM THE DISKETTE - DO NOT USE IT ON A DISKETTE THAT HAS IMPORTANT INFORMATION STORED ON IT.**

Errors reported by DISKTEST may be caused either by the media or by the disk system hardware. If only on an occasional diskette shows errors, the diskettes themselves are probably bad. But if errors are reported for most of the diskettes tested, it is possible that a hardware problem exists. In this case, you should first try cleaning the read/write heads on your disk drives. You can also try bulk erasing the diskettes before formatting them (a bulk tape eraser works just fine for this), or even try a different brand of diskette. If none of these actions eliminates the errors, a hardware problem is very likely.

Any diskette showing a permanent error should be replaced, but a few soft errors may be tolerated. Deciding on an acceptable level of soft errors requires a personal decision. If a diskette shows any soft errors, you would be wise not to use it for very important information. For less important data and/or scratch usage, a few soft errors are usually tolerable. In addition, a diskette that is not satisfactory at double density will often perform perfectly at single density.

The approximate times to complete the long and short disk tests for each of the diskette formats is shown in Figure 8.48.

<u>format</u>	<u>full test</u>	<u>short test</u>
single density	3 min, 16 sec	1 min, 5 sec
double density	3 min, 23 sec	1 min, 8 sec
double sided	6 min, 43 sec	2 min, 16 sec

Figure 8.48 Copying Times for DISKTEST

DISKTEST also has a command line mode so that you can use it from SUBMIT files and menu programs.

**Using DISKTEST - Interactive Mode**

Figure 8.49 shows a typical console dialog that might take place when using DISKTEST.

```
1: A>DISKTEST<enter>
2:
3:
4:                               1D/2D/2S Disk Test Utility - Ver 2.0xx
5:                               copyright (C) 1980,82,83 Pickles & Trout
6:                               all rights reserved
7:
8:
9:               Caution: This test destroys any data on the disk.
10:                  Press <BREAK> at any time to quit.
11:
12: Enter drive to test (A-D or 0-3) : B<enter>
13:     NOTE: Logical drive B: is the same as physical drive (1).
14:
15: This test normally makes 3 passes.
16: Do you want the single-pass option? (y/n) : N<enter>
17:
18: Mount disk to test on drive B: and press <ENTER> when ready to start : <enter>
19:
20: Testing Double-density disk in drive B:
21:
22: Beginning pass with pattern = 00
23: .....
24:
25: Beginning pass with pattern = FF
26: .....
27:
28: Beginning pass with pattern = DBB6
29: .....
30:
31: Testing complete.
32:
33:
34:               Caution: This test destroys any data on the disk.
35:                  Press <BREAK> at any time to quit.
36:
37: Enter drive to test (A-D or 0-3) or (R) to repeat : <break>
38:
39: A>
```

Figure 8.49 Example Console Dialog for Using DISKTEST

DISKTEST is executed by the command line shown on Line 8.49-1. After displaying its sign-on messages (Lines 8.49-4 to 8.49-6), DISKTEST reminds you that the test it performs will destroy any data already on the diskette and that you may press the <break> key at any time to abort the program and return to the operating system (Lines 8.49-9 and 8.49-10).

DISKTEST then asks you for the drive on which the test is to be performed (Line 8.49-12). Note that you may specify the drive as either a physical drive number or a logical drive letter. After you have entered the drive (B in this example), DISKTEST checks that it is a valid floppy drive on the system. It then shows you the correspondence between the logical drive letter and the physical drive number (Line 8.49-13). It is not necessary that the diskette to be tested be mounted on the drive at this time. If the drive you specify is not a floppy drive or is not defined for the system DISKTEST will display an error message and ask you again for the drive.

DISKTEST then reminds you that the test normally makes three passes over the entire diskette and asks if you want the short (single pass) test (Lines 8.49-15 and 8.49-16). In this example, a negative response was entered to indicate that the long test should be used.

DISKTEST then asks you to mount the diskette to be tested on the drive and press the <enter> key (Line 8.49-18). At this time you may change diskettes, if you wish, before pressing <enter>. After you press <enter> the test will begin.

As the test begins, DISKTEST checks the density of the diskette and reports it as shown on Line 8.49-20. As the test proceeds each data pattern is announced and a dot is displayed on the console display as each track is tested with that pattern (Lines 8.49-22 to 8.49-29). Note: If a double sided diskette is being tested, DISKTEST first tests side 0 and then side 1 at the same head location (cylinder). It displays a dot as side 0 is tested and then replaces it with a colon (:) as side 1 is tested. When the test of a double sided diskette is completed, rows of colons will appear on Lines 8.49-23, 8.49-26, and 8.49-29 instead of dots.

After the test is completed, DISKTEST starts over again as shown in Lines 8.49-34 to 8.49-37. Note that the prompt for the drive is slightly different now. You have the option of entering "R" to repeat exactly the same test again. This is useful when you want to test several diskettes. DISKTEST begins the test immediately after you enter an "R" on Line 8.49-37 so you should mount the diskette to be tested before entering the "R". With the "R" option you can easily test several diskettes by merely mounting them on the drive you initially specified and repeating the test.

When you have finished testing, press the <break> key when asked for a drive, as shown on Line 8.49-37. Pressing the <break> key at other times during the test will cause DISKTEST to abandon the test in progress and return to the beginning (it does not return to the system).

For systems that require a system diskette for warm booting, DISKTEST will prompt you to mount a system diskette on the system drive before returning to the system command level. If you encounter this prompt, insure that a system diskette is mounted on the system drive and press <enter>.

### Using DISKTEST - Command Line Mode

In the command line mode of DISKTEST, you must specify all of the information regarding the test you want performed on the command line that you use to execute DISKTEST. Note that in the command line mode, the test normally begins automatically as soon as DISKTEST begins execution (except on a single floppy drive system). You should insure that the diskette to be tested is mounted before executing DISKTEST in the command line mode. (See the PMOUNT instruction below for an exception to this.)

The general form of the DISKTEST command line is shown in Figure 8.50.

```
DISKTEST (drive to test) [output destinations] [options]
          (required)           (optional)        (optional)
```

All instructions on the command line must be separated from one another by a comma and/or one or more spaces.

Figure 8.50 General Form of DISKTEST Command Line

You must specify the drive on which the test is to be made when using the command line option of DISKTEST. You may use either a physical drive number or a logical drive letter when specifying the drive. The drive is specified using an expression like those shown in Figure 8.51. Note that the drive designation may come before or after the equal sign.

DRIVE=[drive]	DRV=[drive]	DR=[drive]
[drive]=DRIVE	[drive]=DRV	[drive]=DR

[drive] should be replaced by a logical drive letter or a physical drive number.

Figure 8.51 Specifying the Drive on Which to Test

The command line mode of DISKTEST allows you to direct the console output from the program to the console, system printer, or a disk file. The output defaults to the console if no output direction is specified. You may direct the output to any combination of these three destinations. Any destination(s) you specify will be the only destination(s) for the console output. For example, if you specify a disk file for the destination, no output will appear on the console. If you want output to both a disk file and the console you must specify both as output destinations. A special output destination (NUL:) is provided for cases where you want no output at all from the program. An output destination is specified by a slash ( / ) followed by the destination name. Examples are given in Figure 8.52.

/CON:	sends output to the console (default if no output instructions are given).
/LST:	sends output to system printer.
/(valid file name)	sends output to the specified file. If a file with the same name already exists, it is replaced.
/(valid file name)+	appends output to the specified file. If the file does not already exist, it is created. The + may be preceded by a space or comma.
/NUL: or /NULL:	causes no output to be generated. If any other output instructions appear, they will supersede /NUL:.

Figure 8.52 Instructions to Direct Console Output

There are several additional instructions that can appear on the command line of DISKTEST. They are listed in Figure 8.53. In the command line mode, DISKTEST normally does not prompt for a diskette to be mounted; it immediately begins the test. When it is finished, it does not prompt for a system diskette to be mounted either.

If you wish for DISKTEST to prompt for a diskette to be mounted, you may use the "PMOUNT" instruction (or one of its synonyms) on the command line. This can be useful when DISKTEST is used from a SUBMIT file because it eliminates the need for a separate program to prompt for a diskette to be mounted. If you want DISKTEST to prompt for a system diskette at the end of the copy operation you should include the "PSYS" instruction on the command line. This instruction causes DISKTEST to issue a prompt for the system diskette if one is required for a warm boot operation. No prompt will be issued for systems that do not require a system diskette for a warm boot. You may instruct DISKTEST to prompt both for a diskette to test and for the system diskette by using the "PBOTH" instruction on the command line.

DISKTEST will always issue prompts for a diskette to test and for the system diskette (if one is needed for a warm boot) if the system has only a single floppy drive and no hard disk drives.

The Q instruction tells DISKTEST to perform the short one pass test. If the Q instruction does not appear on the command line, DISKTEST will perform the long three pass test.

PMOUNT or PM	causes DISKTEST to prompt for a diskette to be mounted before beginning the test.
PSYS or PS	causes DISKTEST to prompt for a system diskette to be mounted after the test is completed.
PBOTH or PB	causes DISKTEST to prompt both a diskette to be tested and for a system diskette after the test is completed.
Q	causes DISKTEST to perform the short one pass test instead of the standard three pass test.

Figure 8.53 Additional Command Line Instructions for DISKTEST

The following examples show command lines that could be used with DISKTEST:

**DISKTEST DRIVE=B**

Tests the diskette mounted on drive B with the full three pass test. The diskette is assumed to be mounted already and the test will begin immediately. Messages are sent to the console. DISKTEST will not prompt for a system disk after the test.

**DISKTEST DR=C,Q**

Tests the diskette mounted on drive C with the short one pass test. The diskette is assumed to be mounted already and the test will begin immediately. Messages are sent to the console. DISKTEST will not prompt for a system disk after the test.

**DISKTEST DR=C PSYS**

Tests the diskette mounted on drive C with the full three pass test. The diskette is assumed to be mounted already and the test will begin immediately. Messages are sent to the console. DISKTEST will prompt for a system disk after the test if one is required for a warm boot.

**DISKTEST DR=B,/A:LOG,PBOTH**

Tests the diskette on drive B with the full three pass test. DISKTEST will prompt for the diskette to be mounted before the test begins. Messages are sent to a disk file named LOG on drive A. DISKTEST will prompt for a system disk after the test if one is required for a warm boot.

**DISKTEST DRV=B,/CON:/LST:**

Tests the diskette on drive B with the full three pass test. The diskette is assumed to be mounted already and the test will begin immediately. Messages are sent to the console and the system printer. DISKTEST will not prompt for a system disk after the test.

**DISKTEST DR=C,/LOG+,Q**

Tests the diskette on drive C with the short one pass test. The diskette is assumed to be mounted already and the test will begin immediately. Messages are appended to a file named LOG on the current default drive. DISKTEST will not prompt for a system disk after the test.

## Possible Error Messages

### **Non-modular system, not compatible with this program.**

You are trying to run this program with a previous version of P&T CP/M 2. The program makes use of features found only in the modular version of P&T CP/M 2 and hence cannot run on this system.

### **Bad drive name, please re-enter (0-3 or A-P) :**

The letter you entered to specify the logical drive on which to test was not in the range A - P. You should respecify the drive using a valid logical drive letter or a physical drive number.

#### **>>> Error: Bad drive name**

Same meaning as the previous message but given in the command line mode only.

### **Bad drive number, please re-enter (0-3 or A-P) :**

The number you entered to specify the physical drive on which to test was not in the range 0 - 3. You should respecify the drive using a valid physical drive number or a logical drive letter.

#### **>>> Error: Bad drive number**

Same meaning as the previous message but given in the command line mode only.

### **That drive is not on the system, please re-specify:**

You have specified a valid drive designation but the drive is not available on your system. For example, you might have specified drive D on a two-drive system. Re-enter a drive designation using one of the drives that is available on your system.

#### **>>> Error: Drive not on system**

Same meaning as the previous message but given in the command line mode only.

### **That drive is a hard disk drive, please re-specify:**

The drive you have specified is a hard disk drive. DISKTEST can work only on floppy disk drives. You should re-enter a drive designation for one of the floppy drives on the system. If you do not know which logical drives are assigned to the physical floppy drives on the system you can specify a physical drive number.

#### **>>> Error: Test target drive is a hard disk drive**

Same meaning as the previous message but given in the command line mode only.

#### **>>> Error: Missing drive name**

Given in the command line mode if the program could not find a specification on the command line of the drive on which to perform the test. Recheck the command line you used to be sure you specified the drive.

#### **>>> Error: Bad output designation**

Given in the command line mode if you attempted to redirect the console output but the program could not figure out where you wanted it sent. This can be caused by an illegal character in a file name or by ending the command line with a slash (/).

#### **>>> Error: Output-file drive same as test target drive**

Given in the command line mode if you directed the console output to a disk file on the same drive that is being tested. This is not allowed because it interferes with the test.

**>>> Error: Bad output-file drive name**

Given in the command line mode if you directed console output to a disk file and used an illegal character (ie. not A - P) to specify the drive on which to place the file.

**>>> Error: Output-file drive not on system**

Given in the command line mode if you directed console output to a disk file on a drive that is not present on the system. Recheck the command line you used to make sure you specified the correct drive.

**>>> Error: Output-file drive cannot be a floppy on a single-floppy system**

Given in the command line mode if you directed console output to a disk file on a floppy diskette on a system that has a single floppy drive. This is not allowed even if disk swapping is in effect to make the single floppy drive appear as if it were multiple drives. Swapping diskettes during the test would not only be tedious but also would interfere with the test.

**>>> Error: Logged-on drive same as test target drive**

This message is given in the command line mode if the program is instructed to test the drive which is the current default drive when the program was executed. The program assumes that the current default drive contains important files and should not be tampered with. This message is not given, however, if the PMOUNT instruction is given on the command line so that the program prompts for a diskette to be mounted before beginning testing. If you get this message and are sure that you are specifying the correct drive to be tested, make another drive the current default drive before executing the program or use the PMOUNT instruction.

**Disk to test on drive X: is not empty.**

**Do you want to continue with the test? (y/n) :**

When the program started to test the diskette, it found that the diskette had files on it. Since this test is destructive, the program warns you that you are about to destroy all information on the diskette and asks you whether you want to continue. This is done as a protective measure to help you to avoid erasing an important diskette. Check whether the diskette is really the one you want to test and respond accordingly. The "X" is replaced by the logical drive letter of the drive being used for the test.

**>>> Error: Test aborted by user.**

This message is displayed if you respond negatively to the question in the previous message.

**Please answer with (Y) for yes or (N) for no :**

This message is displayed if you answer a yes/no question with an invalid response. Enter your response again using one of the letters specified.

**>>> Error: Disk on drive X: is write protected — cannot test.**

When the program checked the diskette on the specified drive, it found that the diskette was write protected. Since this test involved writing to the diskette, the test could not proceed. Check that the diskette is really the one you want to test and, if it is, cover the write protect notch and try again. The "X" is replaced by the logical drive letter of the drive being used for the test.

**>>> Error: Drive X: is not ready.**

When the program started the test, it found that the drive specified for the test was not ready. This is usually caused by not having a diskette mounted when

the test begins. The "X" will be replaced by the logical drive letter of the drive that was found not to be ready.

**>>> Error: Home error.**

A disk error occurred when the program attempted to restore the read/write head to the home position. This can be caused by an unformatted or improperly formatted diskette or an improper drive step rate.

**>>> Error: Seek error.**

A disk error occurred when the program was moving the read/write head to a particular location. This can be caused by an unformatted or improperly formatted diskette or an improper drive step rate.

**>>> Error: Unable to determine density of disk.**

The program was unable to determine the density of the diskette being tested. Although it is rare to see this error, it can be caused by an unformatted or improperly formatted diskette.

**>>> Error: Check aborted with <BREAK>.**

This message is issued if you abort the test by pressing the <break> key while the test is running.

**>>> Error: xx Soft read error(s): Track yy, Sector zz**

This message is given when soft errors are encountered. Whenever a sector on the disk cannot be read successfully, the program will attempt to read it again. If a subsequent attempt is successful, each previous failure is called a soft error. In the error message, "xx" would be replaced by the number of times the attempt failed before a successful read was performed. "yy" and "zz" would be replaced by the track and sector numbers, respectively, where the soft errors occurred.

**>>> Error: xx Soft read error(s): Cylinder yy, Side s, Sector zz**

Same meaning as the previous message except that it is used when a double sided diskette is being tested. In the error message, "xx" would be replaced by the number of times the attempt failed before a successful read was performed. "yy", "s", and "zz" would be replaced by the cylinder, side and sector numbers, respectively, where the soft errors occurred.

**>>> Error: Permanent read error: Track yy, Sector zz**

A permanent error has been encountered. A permanent read error is a failure to read a sector on the disk successfully in 10 consecutive attempts. The message reports the track and sector number at which the error took place.

**>>> Error: Permanent read error: Cylinder yy, Side s, Sector zz**

Same meaning as the previous message except that it is used when a double sided diskette is being tested. In the error message, "yy", "s", and "zz" would be replaced by the cylinder, side and sector numbers, respectively, where the permanent error was found.

**>>> Error: Permanent write error: Track yy, Sector zz**

Same as a permanent read error except caused by a failure to write a sector on the disk successfully in 10 consecutive attempts. The message reports the track and sector number at which the error took place.

**>>> Error: Permanent write error: Cylinder yy, Side s, Sector zz**

Same meaning as the previous message except that it is used when a double sided diskette is being tested. In the error message, "yy", "s", and "zz" would

be replaced by the cylinder, side and sector numbers, respectively, where the permanent error was found.

**>>> Error: Compare error: . . . . Track yy, Sector zz**

This error message is displayed when no read or write errors have occurred yet the data written out to the diskette does not agree with the data read back. This usually is not due to a disk media problem. It might be due to hardware problems with the disk system but it is more likely a problem with the memory used to store the two sets of data before they are compared. The "yy" and "zz" will be replaced by the track and sector numbers, respectively, at which the error occurred.

**>>> Error: Compare error: . . . . Cylinder yy, Side s, Sector zz**

Same meaning as the previous message except that it is used when a double sided diskette is being tested. In the error message, "yy", "s", and "zz" would be replaced by the cylinder, side and sector numbers, respectively, where the permanent error was found.

**>>> Error: Read Time-out error: Track yy, Sector zz**

In some cases, when reading a diskette, if the desired sector cannot be found, the disk controller circuit will try to locate it on several successive revolutions of the diskette before reporting an error. If the sector could not be found on the very first attempt, a problem could exist on the diskette. This message is given if too much time elapses from when the "read" command is given to the disk controller to when the sector was actually read. This indicates that the controller required more than one revolution to find the sector meaning that the sector was missed at least once. This may indicate the very early stages of degradation of a sector.

**>>> Error: Read Time-out error: Cylinder yy, Side s, Sector zz**

Same meaning as the previous message except that it is used when a double sided diskette is being tested. In the error message, "yy", "s", and "zz" would be replaced by the cylinder, side and sector numbers, respectively, where the error was found.

**>>> Error: Write Time-out error: Track yy, Sector zz**

Same meaning as a Read Time-out error except that it occurred while writing to a diskette.

**>>> Error: Write Time-out error: Cylinder yy, Side s, Sector zz**

Same meaning as the previous message except that it is used when a double sided diskette is being tested. In the error message, "yy", "s", and "zz" would be replaced by the cylinder, side and sector numbers, respectively, where the error was found.

**8.13 Utility name: DUMP**

Purpose: To display a disk file in hexadecimal form.

**General Description**

DUMP is a very simple program that reads a disk file and displays it on the console in hexadecimal format. The file is displayed in lines of 16 hex numbers, each representing one byte of the file. The number (relative to the beginning of the file) of the first byte of each line is shown at the beginning of the line. The source code of DUMP is supplied with the system (filename: DUMP.ASM) as an example of an assembly language program.

**Using DUMP**

Figure 8.54 shows a sample running of the DUMP program. The name of the file to be displayed is specified on the command line that executes DUMP. Line 8.54-1 shows the command line which will cause DUMP to display the file DATIME.COM on the console in hexadecimal form. The resulting output is shown on Lines 8.54-3 to 8.54-15. After displaying the file, DUMP returns to the operating system.

```
1: A>DUMP DATIME.COM<enter>
2:
3: 0000 18 32 43 6F 70 79 72 69 67 68 74 20 31 39 38 30
4: 0010 20 62 79 20 50 69 63 6B 6C 65 73 20 26 20 54 72
5: 0020 6F 75 74 20 20 72 65 76 69 73 65 64 20 31 30 2F
6: 0030 33 2F 38 30 ED 73 C2 03 31 0B 04 06 18 CD 40 00
7:   .   .   .   .   .   .   .   .
8:   .   .   .   .   .   .   .   .
9:   .   .
10: 02A0 03 51 03 5A 03 60 03 66 03 6A 03 6F 03 74 03 7B
11: 02B0 03 85 03 8D 03 96 03 00 00 00 00 00 00 00 00 00
12: 02C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
13: 02D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
14: 02E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
15: 02F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
16:
17: A>
```

Figure 8.54 Example of using DUMP

**Possible Error Messages****NO INPUT FILE PRESENT ON DISK**

DUMP could not locate the file name specified on the command line.

**8.14 Utility name: ED**

**Purpose: A line oriented text editor.**

### **General Description**

ED is a general purpose, line oriented text editor which is a part of the Digital Research standard release of CP/M 2. Since ED is line oriented (rather than screen oriented), it is somewhat harder to use than more modern text editors. ED is adequate for small jobs, such as creating SUBMIT files or making small changes to programs. If you intend to do a lot of text editing, however, a more powerful, screen oriented editor is highly recommended.

### **Using ED**

Chapter 2 of CP/M Operating System Manual gives full instructions on the use of ED. The limited discussion below touches mainly on points not mentioned in that manual or ones that deserve emphasis.

One characteristic of ED is frequently overlooked. When ED is first executed with an existing file, none of the file is read into ED's memory buffer. It is necessary to use the append command (A) to "append" lines from the file into the empty buffer. If you do not use the A command, it will appear that nothing exists in the file you are editing.

Another noteworthy feature of ED is that it responds differently to commands entered in upper and lower case letters. When commands are given in upper case letters, ED converts all text inserted into the file to upper case, so the U command will have no effect. If lower case letters are used in commands, lower case characters will not be converted to upper case unless a +U command is given.

The type command (T) does not advance the character pointer, so repetitions of the command simply display the same lines of the file. One way to display successive lines is to change the pointer's position with the L command before each new T command. Alternately, parameters can be included with the T command when the line numbers are known. For example, the command 43:62T will move the character pointer to line 43 and then type lines 43 through 62.

The "sleep" command (nZ) causes ED to suspend operation (go to sleep) for a period of time. The number "n" specifies the number of time periods (each about .3 seconds on a 4 Mhz Z-80) to pause. This command can be used in a macro that prints the entire edit buffer in successive groups of lines, with ED pausing after each group. For example, the macro #m20t20l50z will type 20 lines, advance the character pointer 20 lines, and wait for about 15 seconds. This action will be continued until the end of the edit buffer is reached.

### **Possible Error Messages**

See pp. 43,44 of the CP/M Operating System Manual for an explanation of the error messages.

**8.15 Utility name: ERROR**

Purpose: To explain the error codes displayed by the BIOS portion of the system.

**General Description**

The BIOS portion of P&T CP/M 2 reports errors that occur when performing various disk functions. In addition to a descriptive name for the message (e.g. Read error) an error code is also given. This error code provides more detailed information about the error that occurred. It is very difficult to present in a manual an explanation of all the error codes that might occur.

The ERROR program is provided to interpret the error codes and give you a more detailed explanation of them.

**Using ERROR**

ERROR is very easy to use. All you need to do is type ERROR followed by a space followed by the error code you want explained. Figure 8.55 shows an example of using ERROR.

```
1: Read error, code = R08
2: Bdos Err on B: Bad Sector <break>
3: A>ERROR R08<enter>
4:
5: A disk error occurred while trying to read a floppy diskette.
6: The following error conditions were detected :
7:
8: A CRC error was reported. This is typically due to a bad spot on the disk.
9: It may also occur if the disk has been abused. In some cases it may be caused
10: by hardware problems.
11:
12: A>
```

Figure 8.55 Example of Using ERROR

**Possible Error Messages****>>> No error code specified**

This message is displayed if you did not type an error code on the command line of ERROR. You will be returned to the command level of the system immediately following this message.

**[err code] ?**

If you ask ERROR to explain an error code that it does not know about, it will display the code on the console followed by a question mark. You will be returned to the command level of the system immediately following this message. This is usually caused by mis-typing the error code.

**8.16 Utility name: FASTCOPY**

**Purpose:** To copy all files on one or more user numbers from one disk to another while maintaining file attributes.

**General Description**

FASTCOPY is a file transfer program designed to copy all files on one or more user numbers from one disk to another in a rapid and efficient manner. Unlike PIP, which transfers only one file at a time, FASTCOPY reads files from the source disk until the memory available for temporary storage is full. Only then does FASTCOPY begin to write the file out to the destination disk. Transferring files in this way is especially advantageous on a single-drive system, as it limits the number of times the diskettes will need to be swapped.

Since FASTCOPY does a file-by-file copy, copying files to an empty disk will eliminate any file fragmentation that may exist on the source disk.

It is important to note that FASTCOPY treats all files as being sequential and may not properly transfer random access files.

FASTCOPY will optionally transfer files of zero length (i.e. files which have been created but have never had data written to them). It will also optionally read back the files from the destination diskette to verify that they were copied properly.

FASTCOPY will transfer ALL files on the selected user number(s) from the source disk to the destination disk. All file attributes such as SYS, DIR, R/W, and R/O will be maintained. Any file on the destination disk having the same name as one on the source disk will be deleted before the transfer is made. You should assure that none of these files has important information stored in it. Other files on the destination disk will not be disturbed. FASTCOPY will continue until all files have been copied or until available space on the destination disk is exhausted.

**Using FASTCOPY - Interactive Mode**

FASTCOPY's interactive mode of operation prompts you for all the information necessary to specify the copy operation to be performed. It also provides the option of performing the same operation repetitively without re-entering the information. Figure 8.56 shows a typical console dialog for using FASTCOPY in the interactive mode.

On a single drive system, you must make use of the system's disk swapping to use FASTCOPY (see Section 3.3 for an explanation of disk swapping). You would use FASTCOPY as shown in the example below (where a multiple drive system is assumed) but would specify two logical drives that are assigned to the single floppy drive. The system will then automatically flash messages on the console asking you to mount the different diskettes as they are needed.

Note that if you need to copy files from one diskette to another on a hard disk system with a single floppy drive, it is usually easier to copy all files from the source diskette to the hard disk and then back to the destination diskette than it is to use disk swapping. This assumes that at least one of the following is true:

1. You have an empty logical drive on the hard disk.
2. You have an empty user area on one of the logical drives on the hard disk. In this case you should use the PIP program with the G option to transfer the files to the empty user area and then back to the new diskette.
3. The names of the files are all similiar enough so that you can transfer them with a wildcard file name using PIP. This assumes that there are no files with names that also match the wildcard name on the hard disk drive being used.

Line 8.56-1 shows the command line that will execute FASTCOPY in the interactive mode. After FASTCOPY displays its opening messages (Lines 8.56-4 to 8.56-6) it reminds you (Lines 8.56-8 to 8.56-9) of the type of copy operation it performs.

On Line 8.56-11 FASTCOPY reminds you that the <break> key will abort the copy process. If you press the <break> key while FASTCOPY is copying files, it will stop as soon as it has finished reading or writing the current file.

On Line 8.56-13 FASTCOPY asks you for the source drive. This is the drive on which you will mount the disk to be copied. Note that you can enter the drive specification as either a physical drive number or a logical drive letter. The physical drive numbers always refer to the floppy disk drives. The logical drive letters can refer either to a floppy drive or a hard disk drive. You may choose the method of specifying the drive which is most comfortable for you. In this example, logical drive D is selected for the source drive. On Line 8.56-14 FASTCOPY shows you that on this system logical drive D is equivalent to physical drive 3. If the logical drive you specify is a hard disk drive, the message displayed on Line 8.56-14 will indicate that this is the case.

A similar query is made for the destination drive on 8.56-16. The destination drive is the drive on which you will mount the disk to which the copy will be made. In this example the destination drive is specified as logical drive C. On Line 8.56-17, FASTCOPY shows that logical drive C corresponds to physical drive 2. If the logical drive you specify is a hard disk drive, the message displayed on Line 8.56-17 will indicate that this is the case.

After you have specified the source and destination drives, FASTCOPY asks you for the user areas to transfer (Line 8.56-19). In response you can enter a list of user area numbers separated by a comma and/or one or more spaces. Only files in the specified user areas will be transferred. If you want to transfer files from all user areas, you may enter "ALL" or "all" instead of a list of user numbers. If you want to transfer files from the current user area only, you need merely press <enter> as shown on Line 8.56-19. Note that files are always copied to the same user area on the destination drive as they came from on the source drive.

If you do not understand user areas and/or are not using them merely press <enter> on Line 8.56-19.

Next FASTCOPY asks you if you want to transfer zero length files. A file that has zero length is one that was created but never had any data written to it. Some sloppy programs create files on a disk but never use them (for example if the program is aborted before it finishes) and never delete them. Usually these files are just a nuisance so you do not want them copied. In some cases, however, the zero length files are actually required by the programs. In these cases you would want them transferred. On Line 8.56-21 this example elects not to have zero length files transferred.

```
1: A>FASTCOPY<enter>
2:
3:
4:                               1D/2D/2S  Fastcopy Utility - Ver 2.xxx
5:                               copyright (C) 1980,82,83  Pickles & Trout
6:                               all rights reserved
7:
8: This program copies all files from specified user areas on the source
9: disk to the destination disk while maintaining all file attributes.
10:
11: (Press <BREAK> at any time to quit)
12:
13: Enter source drive (A-P or 0-3) : D<enter>
14:     NOTE: Logical drive D: is the same as physical drive (3).
15:
16: Enter destination drive (A-P or 0-3) : C<enter>
17:     NOTE: Logical drive C: is the same as physical drive (2).
18:
19: Enter user areas to transfer (<ENTER> for current area only) : <enter>
20:
21: Do you want to transfer zero-length files? (y/n) : N<enter>
22:
23: Do you want each file verified as it's written? (y/n) : N<enter>
24:
25: Once copying has started, you may abort it at any time by pressing <BREAK>.
26:
27: Mount source disk on drive D:
28: Mount destination disk on drive C:
29:     Press <ENTER> when ready :
30:
31: Copying from drive D: to drive C:
32: -----
33: Reading SOURCE .ASM R/W, DIR, user= 0
34: -----
35: Writing SOURCE .ASM R/W, DIR, user= 0
36: -----
37: Reading SOURCE .ASM R/W, DIR, user= 0
38: Reading PROGRAM .ASM R/W, DIR, user= 0
39: -----
40: Writing SOURCE .ASM R/W, DIR, user= 0
41: Writing PROGRAM .ASM R/W, DIR, user= 0
42: -----
43: Reading PROGRAM .ASM R/W, DIR, user= 0
44: Reading PROGRAM .HEX R/W, DIR, user= 0
45: Reading PROGRAM .COM R/W, DIR, user= 0
46: Reading ARRAY .LIB R/W, DIR, user= 0
47: Reading SCRN .LIB R/W, DIR, user= 0
48: Reading STRCONVT.LIB R/W, DIR, user= 0
49: Reading PIP .COM R/W, DIR, user= 0
50: -----
51: Writing PROGRAM .ASM R/W, DIR, user= 0
52: Writing PROGRAM .HEX R/W, DIR, user= 0
53: Writing PROGRAM .COM R/W, DIR, user= 0
54: Writing ARRAY .LIB R/W, DIR, user= 0
55: Writing SCRN .LIB R/W, DIR, user= 0
56: Writing STRCONVT.LIB R/W, DIR, user= 0
57: Writing PIP .COM R/W, DIR, user= 0
58: -----
59: Reading PIP .COM R/W, DIR, user= 0
60: -----
61: Writing PIP .COM R/W, DIR, user= 0
62:
63:
64: (Press <BREAK> at any time to quit)
65:
66: Enter source drive (A-P or 0-3) or (R) to repeat : <break>
67:
68: A>
```

Figure 8.56 Example Console Dialog for Using FASTCOPY

On Line 8.56-23, FASTCOPY asks you if you want each file verified as it is written.

If you respond affirmatively, FASTCOPY will read back each file to check for errors. This verification slows down the copy process but it does provide checking for errors in the copy. In this example a negative response is given indicating that no verification is to be done.

After you have entered all the necessary information, FASTCOPY will ask you to mount the source and destination disks as shown on Lines 8.56-27 to 8.56-29. At this point you should mount the disks (if necessary) and press <enter>. If diskettes are involved in the copy operation, they need not be of the same density. However, you must be sure that enough space is available on the destination disk to hold all of the transferred files. After you have mounted the disks and pressed the <enter> key, the copying will begin.

FASTCOPY reports each file name on the console as it is being read or written. If several files will fit into the available memory at one time, they will all be read before writing them to the destination disk. If FASTCOPY runs out of memory in the middle of reading a file, it stops reading the file and begins to write out to the destination disk. After the file data stored in memory is written out, FASTCOPY resumes reading the file where it left off. Each time FASTCOPY switches between reading and writing, it displays a line of dashes on the console.

In this example, the file SOURCE.ASM is larger than the available memory so only part of it is read at first (Line 8.56-33). After that part is written (Line 8.56-35) the remainder of the file is read and FASTCOPY begins reading PROGRAM.ASM (Lines 8.56-37 and 8.56-38). These are then written out (Line 8.56-40 and 8.56-41) and FASTCOPY continues reading and writing files until all of the files in the specified user areas have been transferred. In Lines 8.56-43 to 8.56-49 several shorter files are read into memory at once and are then written out on Lines 8.56-51 to 8.56-57.

When the copy is finally complete, FASTCOPY starts over again as shown in Lines 8.56-64 to 8.56-66. Note that on Line 8.5366 a new option is available to you. By entering "R" instead of a drive designation, you may repeat exactly the same copy operation you just completed. This is useful when making multiple copies. In this example the <break> key is pressed to return to the operating system. In this example, the system does not require a system diskette for a warm boot so FASTCOPY returns immediately to the operating system after the <break> key is pressed. If a system diskette were required to accomplish the warm boot, a prompt to mount a system diskette on the system drive would have been given before returning to the system command level.

If you request that FASTCOPY verify the files that are written to the destination drive, the console display during the copy operation will behave a little differently. As a file is written to the destination drive, the message "Writing ..." will be displayed as shown on Line 8.56-35. As FASTCOPY is verifying the file, the word "Writing" will be replaced by "Verifying". When the verification is completed, the word "Verifying" will be replaced by "Verified". If the example shown in Figure 8.56 had specified verification, the display would look exactly the same except that everywhere "Writing" appears, "Verified" would appear instead.

### Using FASTCOPY - Command Line Mode

In the command line mode of FASTCOPY, you must specify all of the information regarding the copy operation you want performed on the command line that you use to execute FASTCOPY. Note that in the command line mode, the copy normally begins automatically as soon as FASTCOPY begins execution (except on a single floppy drive system). You should insure that the source and destination disks are

mounted before executing FASTCOPY in the command line mode. (See the PMOUNT instruction below for an exception to this.)

The general form of the FASTCOPY command line is shown in Figure 8.57.

```
FASTCOPY [src and dest drives] [user numbers] [output destinations] [options]
          (required)           (optional)        (optional)        (optional)
```

All instructions on the command line must be separated from one another by a comma and/or one or more spaces.

Figure 8.57 General Form of FASTCOPY Command Line

You must specify the source and destination drives when using the command line option of FASTCOPY. You may use either physical drive numbers or logical drive letters when specifying the drives. The source drive is specified using an expression like those shown in Figure 8.58. Note that the drive designation may come before or after the equal sign.

```
SOURCE=[drive]      SRC=[drive]
[drive]=SOURCE      [drive]=SRC
```

[drive] should be replaced by a logical drive letter  
or a physical drive number.

Figure 8.58 Specifying the Source Drive

In a similar manner, the destination drive is specified as shown in Figure 8.59.

```
DESTINATION=[drive]    DEST=[drive]    DST=[drive]
[drive]=DESTINATION   [drive]=DEST   [drive]=DST
```

[drive] should be replaced by a logical drive letter  
or a physical drive number.

Figure 8.59 Specifying the Destination Drive

There are also several short forms for specifying both the source and destination drives in a single statement. Examples of these short forms are given in Figure 8.60.

A>B	A is source, B is destination
B TO C	B is source, C is destination
B<A	A is source, B is destination
B=A	A is source, B is destination

Note: Physical drive numbers may be used  
instead of logical drive letters.

Figure 8.60 Short Forms of Specifying Source and Destination Drives

In the command line mode, FASTCOPY will default to copying files from the current user area only. If you wish to copy files from another user area, from several user areas, or from all user areas, you must use the "USER" instruction on the command line. Figure 8.61 shows the form of this instruction. Note that files are always copied to the same user area on the destination disk as they came from on the source disk.

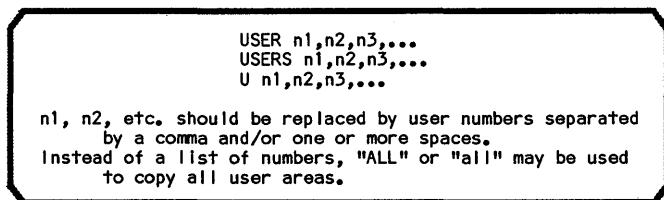


Figure 8.61 Specifying the User Areas to Copy

The command line mode of FASTCOPY allows you to direct the console output from the program to the console, the system printer, or a disk file. The output defaults to the console if no output direction is specified. You may direct the output to any combination of these three destinations. Any destination(s) you specify will be the only destination(s) for the console output. For example, if you specify a disk file for the destination, no output will appear on the console. If you want output to both a disk file and the console you must specify both as output destinations. A special output destination (NUL:) is provided for cases where you want no output at all from the program. An output destination is specified by a slash ( / ) followed by the destination name. Examples are given in Figure 8.62.

/CON:	sends output to the console (default if no output instructions are given).
/LST:	sends output to system printer.
/(valid file name)	sends output to the specified file. If a file with the same name already exists, it is replaced.
/(valid file name)+	appends output to the specified file. If the file does not already exist, it is created. The + may be preceded by a space or comma.
/NUL: or /NULL:	causes no output to be generated. If any other output instructions appear, they will supersede /NUL:.

Figure 8.62 Instructions to Direct Console Output

There are several additional instructions that can appear on the command line of FASTCOPY. They are listed in Figure 8.63. In the command line mode, FASTCOPY normally does not prompt for disks to be mounted; it immediately begins the copy operation specified on the command line. When it is finished, it does not prompt for a system diskette to be mounted either.

If you wish for FASTCOPY to prompt for the source and destination diskettes to be mounted, you may use the "PMOUNT" instruction (or one of its synonyms) on the command line. This can be useful when FASTCOPY is used from a SUBMIT file because it eliminates the need for a separate program to prompt for the diskettes to be mounted. If you want FASTCOPY to prompt for a system diskette at the end of the copy operation you should include the "PSYS" instruction on the command line. This instruction causes FASTCOPY to issue a prompt for the system diskette if one is required for a warm boot operation. No prompt will be issued for systems that do not require a system diskette for a warm boot. You may instruct FASTCOPY to prompt both for the source and destination diskettes and for the system diskette by using the "PBOTH" instruction on the command line.

FASTCOPY will always issue prompts for the source and destination diskettes and for the system diskette (if one is needed for a warm boot) if the system has only a single floppy drive and no hard disk drives.

PMOUNT or PM	causes FASTCOPY to prompt for the source and destination diskettes to be mounted before beginning the copy.
PSYS or PS	causes FASTCOPY to prompt for a system diskette to be mounted after the copy is completed.
PBOTH or PB	causes FASTCOPY to prompt both for the source and destination diskettes and for a system diskette after the copy is completed.
V or VERIFY	causes FASTCOPY to perform a read back verification of each file as it is written to the destination diskette.
Z	causes FASTCOPY to copy zero length files (they are normally not copied).
Q	causes the names of the files being copied to be removed from console output that is directed to a disk file.

Figure 8.63 Additional Command Line Instructions for FASTCOPY

The "V" or "VERIFY" instruction allows you to tell FASTCOPY to verify each file that is transferred to insure a correct copy. If you do not use this instruction, FASTCOPY will default to no verification.

The "Z" instruction tells FASTCOPY to copy zero length files as well as other files in the specified user area(s). If you do not use this instruction, FASTCOPY will default to not copying zero length files.

The "Q" instruction allows you to specify an abbreviated form of the console output when you direct it to a disk file. If you direct the output to a disk file and do not use the "Q" instruction, FASTCOPY will report all the files it copies to the disk file. This may result in a rather large output file and can slow the copy process down noticeably. The "Q" instruction causes FASTCOPY to report only its beginning message and any error messages to the console output file. Not only does this make the output file smaller but also makes it easier to scan for error messages.

The following examples show command lines that could be used with FASTCOPY:

#### **FASTCOPY SOURCE=B, DESTINATION=C**

Copy all files from the current user area on drive B to the current user area on drive C. The disks are assumed to be mounted already and the copy will begin immediately. Messages are sent to the console. No verification is done. FASTCOPY will not prompt for a system disk after the copy.

#### **FASTCOPY C>D,Z,V**

Copy all files from the current user area on drive C to the current user area on drive D. The disks are assumed to be mounted already and the copy will begin immediately. Messages are sent to the console. Verification is performed. Zero length files are copied. FASTCOPY will not prompt for a system disk after the copy.

#### **FASTCOPY C:=B: V U 0,1,9**

Copy all files from user areas 0, 1, and 9 on drive B to the same user areas on drive C. The disks are assumed to be mounted already and the copy will begin immediately. Messages are sent to the console. Verification is done. FASTCOPY will not prompt for a system disk after the copy.

**FASTCOPY C>B,/CON:,/LST:**

Copy all files from the current user area on drive C to the current user area on drive B. The disks are assumed to be mounted already and the copy will begin immediately. Messages are sent to the console and the system printer. No verification is done. FASTCOPY will not prompt for a system disk after the copy.

**FASTCOPY A>C,PBOTH,/B:LOG,V,Q**

Copy all files from the current user area on drive A to the current user area on drive C. FASTCOPY will prompt for the disks to be mounted before beginning the copy. Beginning and error messages only are sent to a file named LOG on drive B. Verification is done. FASTCOPY will prompt for a system disk after the copy if one is required for a warm boot.

**FASTCOPY 1=2,V,/LOG+,/CON:,PSYS,USER ALL**

Copy all files from all user areas of the diskette on physical drive 2 to the same user areas of the diskette on physical drive 1. The diskettes are assumed to be mounted already and the copy will begin immediately. Messages are sent to the console and appended to a file named LOG on the current default drive. Verification is done. FASTCOPY will prompt for a system disk after the copy if one is required for a warm boot.

### Possible Error Messages

**Non-modular system, not compatible with this program.**

You are trying to run this program with a previous version of P&T CP/M 2. The program makes use of features found only in the modular version of P&T CP/M 2 and hence cannot run on this system.

**Bad drive name, please re-enter :**

The logical drive letter you entered was not in the range A to P. Respecify the drive using one of the valid letters or a physical drive number in the range 0 to 3.

**Bad drive number, please re-enter :**

The physical drive number you entered was not in the range 0 to 3. Respecify the drive using one of the valid numbers or a logical drive letter in the range A to P.

**That drive is not on the system, please re-specify :**

The drive you have specified is not available on the system. Re-enter the drive specification indicating one of the drives that is available.

**Source and destination cannot be the same logical drive.**

**Please re-specify.**

If you have more than one physical drive on your system, FASTCOPY will not allow you to specify the same drive for both source and destination. To do so would cause you to do a lot of disk swapping. FASTCOPY will prompt you again for the source and destination drives; specify different drives for each.

**>>> Error: Source drive same as destination drive**

Same meaning as previous message but given in the command line mode.

**Please answer with 'Y' for yes or 'N' for no :**

The response you gave to a yes/no message was not acceptable. Enter your response again using one of the characters indicated. Note that both upper and lower case are acceptable.

**>>> Error: Missing drive name**

(Command Line Mode) This message indicates that FASTCOPY could not find either the source drive or destination drive designation.

**>>> Error: Bad source drive name**

(Command Line Mode) This message is displayed if the letter you used on the command line to specify the source drive was not in the range A - P. This might be caused by an improper command line that causes FASTCOPY to interpret the wrong character as a drive letter.

**>>> Error: Bad destination drive name**

(Command Line Mode) Same as previous message except for destination drive.

**>>> Error: Bad source drive number**

(Command Line Mode) This message is displayed if the number you used on the command line to specify the source drive was not in the range 0 - 3. This might be caused by an improper command line that causes FASTCOPY to interpret the wrong number as a drive number.

**>>> Error: Bad destination drive number**

(Command Line Mode) Same as previous message except for the destination drive.

**>>> Error: Source drive not on system**

(Command Line Mode) This message indicates that you have used a proper letter or number to specify the source drive but that drive does not exist on the system. Check that you are really specifying the drives you want to use for the copy.

**>>> Error: Destination drive not on system**

(Command Line Mode) Same as previous message except for the destination drive.

**>>> Error: Logged-on drive same as destination drive**

(Command Line Mode) In the command line mode, FASTCOPY usually begins the copy operation immediately upon execution. In this case, it does not allow the current default drive to be the destination of the copy operation under the assumption that there are files on the drive that should not be disturbed. You should log onto a drive other than the destination before executing FASTCOPY in the command line mode. This message will not be given if you instruct FASTCOPY to prompt for the source and destination diskettes to be mounted (see the "PMOUNT" command).

**>>> Error: Output-file drive same as source drive**

(Command Line Mode) FASTCOPY does not allow you to direct console output to a disk file on the source drive since doing so can interfere with the copy operation. You should direct console output to a disk file on another drive.

**>>> Error: Output-file drive same as destination drive**

(Command Line Mode) FASTCOPY does not allow you to direct console output to a disk file on the destination drive since doing so can interfere with the copy operation. You should direct console output to a disk file on another drive.

**>>> Error: Bad output designation**

(Command Line Mode) The destination you specified for the console output is not valid. This can be caused by ending the command line with a slash ( / ).

**>>> Error: Output-file drive not on system**

(Command Line Mode) You attempted to direct console output to a disk file but the drive specified with the output file name does not exist on the system. Check that you are specifying the correct drive.

**>>> Error: Bad output-file drive name**

(Command Line Mode) Something is wrong with the drive designation you gave for the output file you specified to receive console output. This can be caused by specifying a drive for the file with an invalid character.

**>>> Error: Bad user number designation**

(Command Line Mode) This message indicates that you specified a number outside of the range 0 - 15 in the list of user numbers to copy from the source to the destination drive. Check the command line you gave for errors in specifying the user number.

**>>> Error: Too many files for available memory**

This message is given if insufficient memory is available to hold information about all the files being transferred and leave enough workspace for making the transfers. It is very unlikely that you will encounter this error. If you do, check the amount of reserved memory above the operating system. Reducing this amount may allow FASTCOPY to run. Alternately you should transfer files from fewer user numbers at one time.

**>>> Error: No files found on specified user(s)**

This message indicates that FASTCOPY could find no files to copy on the user numbers you specified. Check that you specified the correct user numbers.

**>>> Error: Cannot open input file - skipping it.**

When FASTCOPY went to open a file for input, it could not be found. This indicates that the file disappeared from the disk directory between the time that FASTCOPY first found it and when it tried to start reading it. This error is very rare and, should it occur, it may indicate a hardware problem.

**>>> Error: Directory full on destination volume - aborting.**

The directory on the destination disk filled up while FASTCOPY was in the process of copying file to it. This can be caused by copying to a disk that already has files on it. It can also be caused by copying to a disk with a smaller directory than the source disk (e.g. from a double density to a single density diskette). When this error occurs, FASTCOPY immediately returns to the operating system. All files transferred prior to the error will remain intact on the destination disk.

**>>> Error: Destination disk full - aborting.**

The available space on the destination disk has been exhausted. This situation can arise when copying to a disk which already has some files on it or when copying to a disk with a smaller capacity than the source disk (e.g. from a double density to a single density diskette). When this error occurs, FASTCOPY immediately returns to the operating system. All files transferred prior to the error will remain intact on the destination diskette.

**>>> Error: Error while closing file - aborting.**

BDOS has detected an error in its attempt to close a disk file. This message

may indicate that diskettes were not properly swapped when using FASTCOPY in a single drive system. FASTCOPY immediately returns to the operating system upon this error. All files transferred prior to the error will remain intact on the destination diskette.

**>>> Error: Unexpected end of file during file verify - aborting.**

This message indicates that, while FASTCOPY was re-reading the destination file for verification, the end of file was encountered before it should have been. This is a very rare error and you should never see this message. If you do, it could indicate hardware problems. FASTCOPY immediately returns to the operating system upon this error. All files transferred prior to the error will remain intact on the destination diskette.

**>>> Error: Compare error during file verify - aborting.**

This message indicates that the data FASTCOPY read back from the destination file does not match what was written. In addition, the disk system did not report an error. This is also a very rare error. Most typically this error occurs after a disk error was reported and you pressed the <Enter> key. Recall that pressing the <Enter> key after a disk error instructs the system to proceed as if no error had occurred. In this case FASTCOPY has no means of knowing that the data read is in error. If you get this error message without an accompanying disk error, as just described, it probably indicates a problem with the RAM in your system. FASTCOPY immediately returns to the operating system upon this error. All files transferred prior to the error will remain intact on the destination diskette (including any portion of the file containing the error that has already been copied).

**>>> Error: Directory update error during file verify - aborting.**

A disk system error occurred when FASTCOPY attempted to close the destination file before verifying it. This close is done to insure that the directory on the destination disk is completely up to date before beginning to verify a file. This is a relatively rare error; its most common cause is improper diskette swapping on a single floppy drive system. FASTCOPY immediately returns to the operating system upon this error. All files transferred prior to the error will remain intact on the destination diskette.

**>>> Error: Re-opening error during file verify - aborting.**

A disk system error occurred when FASTCOPY attempted to reopen the destination file to verify it. This should not occur in a properly functioning system. FASTCOPY immediately returns to the operating system upon this error. All files transferred prior to the error will remain intact on the destination diskette.

**>>> Error: FASTCOPY aborted with <BREAK>.**

This message is displayed if you press the <Break> key while the copy operation is in progress. FASTCOPY will not stop until it has finished reading or writing the file on which it is currently working. Note that the last file transferred to the destination disk may be incomplete if you abort the copy by pressing <Break>.

**Bdos Err on x: Read Only**

This error message actually comes from BDOS rather than FASTCOPY. It should occur only when running FASTCOPY on a single drive system, and it almost certainly indicates that you have made an error in diskette swapping. If you should encounter this message, the best course of action is to mount your system disk and hit the RESET switch.

**8.17 Utility name: FORMAT****Purpose:** To format (prepare) diskettes for use.**General Description**

Before a disk can be used to store data, certain information must be written on it. This function, called formatting, is the task of the FORMAT program.

FORMAT can format single sided diskettes at single or double density and double sided diskettes at double density. Formatting a diskette at double density does not require one that is certified at double density. However, using one certified as a double density diskette is highly recommended to avoid potential loss of data due to diskette failure.

Since the single density format is standardized, most single density diskettes may be used with P&T CP/M 2 exactly as purchased from the manufacturer. However, all diskettes used at double density must be reformatted to be compatible with P&T CP/M 2.

The capacities and number of directory entries for each of the formats is shown in Figure 8.64.

format	usable storage	Directory Entries
single density	243 Kbytes	64
double density	596 Kbytes	128
double sided	1210 Kbytes	192

Figure 8.64 Capacities and Number of Directory Entries for Disk Formats

FORMAT can perform an optional verification pass on a diskette after it is formatted. This verification consists of reading all sectors from all tracks on the diskette and reporting any errors that are encountered. This is the same test that is performed by the DISKCHK program. The amount of time to format diskettes in each of the supported formats is shown in Figure 8.65.

format	no verification	with verification
single density	29 sec	47 sec
double density	29 sec	47 sec
double sided	54 sec	86 sec

Figure 8.65 Format Times for FORMAT

FORMAT also has a command line mode so that it can be executed from a SUBMIT file or a menu system and not require any user input.

**NOTE: FORMAT WILL ERASE ALL INFORMATION FROM THE DISKETTE - DO NOT USE IT ON A DISKETTE THAT HAS IMPORTANT INFORMATION STORED ON IT.**

### Using FORMAT - Interactive Mode

Figure 8.66 shows an example running of the FORMAT program.

```
1: A>FORMAT<enter>
2:
3:
4:                               1D/2D/2S Disk Format Utility - Ver 2,xxx
5:                               copyright (C) 1980,82,83 Pickles & Trout
6:                               all rights reserved
7:
8:                               Press <BREAK> at any time to quit.
9:
10: Enter drive to format (A-P or 0-3) : B<enter>
11:     NOTE: Logical drive B: is the same as physical drive (1).
12:
13: Enter 'S' for Single-density or 'D' for Double-density : D<enter>
14:
15: Do you want a fast check performed after the disk is formatted? (y/n) : Y<enter>
16:
17: Mount disk to format on drive B: and press <ENTER> when ready to start : <enter>
18:
19: Formatting Single-sided disk in drive B: at Double-density.
20: .....
21: Format complete.
22:
23: Checking disk on drive B:
24: .....
25: Checking complete.
26:
27:
28:                               Press <BREAK> at any time to quit.
29:
30: Enter drive to format (A-P or 0-3) or (R) to repeat: <break>
31:
32: A>
```

Figure 8.66 Example Console Dialog for Using FORMAT

The command which will execute FORMAT is given on Line 8.66-1. FORMAT signs on with the messages shown in Lines 8.66-4 to 8.66-6. On Line 8.66-8 FORMAT reminds you that you can abort the program by pressing the <break> key. On Line 8.66-10 FORMAT asks for the drive on which the format operation is to take place. Note that you can specify the drive as either a logical drive letter or a physical drive number. After you enter a drive specification (B in this example), FORMAT checks that the drive exists on the system and is a floppy diskette drive. If it does not exist or is not a floppy drive, an error message is displayed and you will be asked again to specify the drive. If the drive exists and is a floppy drive, FORMAT will show you the logical to physical drive assignment for that drive (Line 8.66-11).

FORMAT next asks you whether you want the diskette formatted at single or double density (Line 8.66-13). In this example, double density was selected. FORMAT then asks if you want the quick verification to be done after the diskette is formatted (Line 8.66-15). In this example an affirmative answer is given to indicate that the verification should be done.

FORMAT then prompts you to mount the diskette to be formatted on the specified drive and press <enter> (Line 8.66-17). At this time you may change diskettes, if you wish, before pressing <enter>. After you press <enter> the format will begin.

**REMEMBER: FORMAT WILL ERASE ALL INFORMATION FROM A DISKETTE - MAKE SURE THE PROPER ONE IS MOUNTED BEFORE PRESSING <enter>.**

When FORMAT begins to format the diskette, it determines whether the diskette is double sided or not. FORMAT will automatically format single and double sided

diskettes accordingly when double density is specified. A double sided single density format is not supported; hence FORMAT will give an error message if a double sided diskette is mounted when single density is specified. In this example, a single sided diskette was mounted. On Line 8.66-19 FORMAT reports the result of its test for number of sides and begins the format operation.

As each track is formatted, a dot is displayed on the console (Line 8.66-20). After the format is completed, the quick verification pass is made (Lines 8.66-23 to 8.66-25). As with the format, a dot is displayed as each track is verified.

For a double sided diskette, a dot is displayed as side 0 is formatted or verified. The dot is replaced by a colon (:) as side 1 is formatted or verified. When formatting a double sided diskette, the console display will end up with lines of colons instead of dots on Lines 8.66-20 and 8.66-24.

After the formatting and optional verification are completed, FORMAT starts over again as shown in Lines 8.66-28 to 8.66-30. Note that the prompt for the drive is slightly different now. You have the option of entering "R" to repeat exactly the same format operation again. This is useful when you want to format several diskettes. FORMAT begins the format immediately after you enter an "R" on Line yy49-31 so you should mount the diskette to be formatted before entering the "R". With the "R" option you can easily format several diskettes by merely mounting them on the drive you initially specified and repeating the format operation.

When you have finished formatting, press the <break> key when asked for a drive, as shown on Line 8.66-30. Pressing the <break> key at other times during the test will cause FORMAT to abandon the format in progress and return to the beginning (it does not return to the system).

For systems that require a system diskette for warm booting, FORMAT will prompt you to mount a system diskette on the system drive before returning to the system command level. If you encounter this prompt, insure that a system diskette is mounted on the system drive and press <enter>.

### Using FORMAT - Command Line Mode

In the command line mode of FORMAT, you must specify all of the information regarding the format operation you want performed on the command line that you use to execute FORMAT. Note that in the command line mode, the format operation normally begins automatically as soon as FORMAT begins execution (except on a single floppy drive system). You should insure that the diskette to be formatted is mounted before executing FORMAT in the command line mode. (See the PMOUNT instruction below for an exception to this.)

The general form of the FORMAT command line is shown in Figure 8.67.

```
FORMAT (drive to format) (density) [output destinations] [options]
      (required)      (required)      (optional)      (optional)
```

All instructions on the command line must be separated from one another by a comma and/or one or more spaces.

Figure 8.67 General Form of FORMAT Command Line

You must specify the drive on which the format operation is to be made when using the command line option of FORMAT. You may use either a physical drive number or a logical drive letter when specifying the drive. The drive is specified using an

expression like those shown in Figure 8.68. Note that the drive designation may come before or after the equal sign.

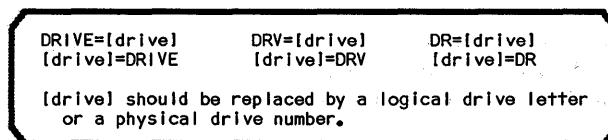


Figure 8.68 Specifying the Drive on Which to Format

The density at which the diskette is to be formatted is specified by one of the instructions shown in Figure 8.69.

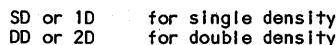


Figure 8.69 Specifying the Density at Which to Format

The command line mode of FORMAT allows you to direct the console output from the program to the console, system printer, or a disk file. The output defaults to the console if no output direction is specified. You may direct the output to any combination of these three destinations. Any destination(s) you specify will be the only destination(s) for the console output. For example, if you specify a disk file for the destination, no output will appear on the console. If you want output to both a disk file and the console you must specify both as output destinations. A special output destination (NUL:) is provided for cases where you want no output at all from the program. An output destination is specified by a slash ( / ) followed by the destination name. Examples are given in Figure 8.70.

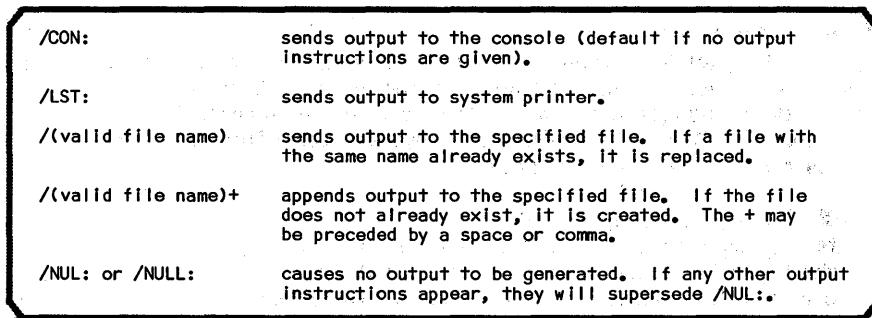


Figure 8.70 Instructions to Direct Console Output

There are several additional instructions that can appear on the command line of FORMAT. They are listed in Figure 8.71. In the command line mode, FORMAT normally does not prompt for a diskette to be mounted; it immediately begins the format operation. When it is finished, it does not prompt for a system diskette to be mounted either.

If you wish for FORMAT to prompt for a diskette to be mounted, you may use the "PMOUNT" instruction (or one of its synonyms) on the command line. This can be useful when FORMAT is used from a SUBMIT file because it eliminates the need for a separate program to prompt for a diskette to be mounted. If you want FORMAT to prompt for a system diskette at the end of the format operation you should include the "PSYS" instruction on the command line. This instruction causes FORMAT to issue a prompt for the system diskette if one is required for a warm boot operation. No prompt will be issued for systems that do not require a system

diskette for a warm boot. You may instruct FORMAT to prompt both for a diskette to format and for the system diskette by using the "PBOTH" instruction on the command line.

FORMAT will always issue prompts for the source and destination diskettes and for the system diskette (if one is needed for a warm boot) if the system has only a single floppy drive and no hard disk drives.

The V instruction tells FORMAT to perform the verification pass.

PMOUNT or PM	causes FORMAT to prompt for a diskette to be mounted before beginning the format.
PSYS or PS	causes FORMAT to prompt for a system diskette to be mounted after the format is completed.
PBOTH or PB	causes FORMAT to prompt both a diskette to be formatted and for a system diskette after the format is completed.
V	causes FORMAT to make a verification pass after formatting the diskette.

Figure 8.71 Additional Command Line Instructions for FORMAT

The following examples show command lines that could be used with FORMAT:

#### **FORMAT DRIVE=B,DD**

Formats the diskette mounted on drive B at double density. The diskette is assumed to be mounted already and the format operation will begin immediately. Messages are sent to the console. No verification is done. FORMAT will not prompt for a system disk after the format operation.

#### **FORMAT DR=C SD V PSYS**

Formats the diskette mounted on drive C at single density. The diskette is assumed to be mounted already and the format operation will begin immediately. Messages are sent to the console. A verification pass is performed. FORMAT will prompt for a system disk after the format operation if one is required for a warm boot.

#### **FORMAT DR=B,DD,/A:LOG,PMOUNT**

Formats the diskette on drive B at double density. FORMAT will prompt for the diskette to be mounted before the format operation begins. Messages are sent to a disk file named LOG on drive A. No verification is done. FORMAT will not prompt for a system disk after the format operation.

#### **FORMAT DRV=B,DD,/CON:,V,/LST:**

Formats the diskette on drive B at double density. The diskette is assumed to be mounted already and the format operation will begin immediately. Messages are sent to the console and the system printer. A verification pass is performed. FORMAT will not prompt for a system disk after the format operation.

#### **FORMAT DR=C,SD,/LOG+,V**

Formats the diskette on drive C at single density. The diskette is assumed to be mounted already and the format operation will begin immediately. Messages are appended to a file named LOG on the current default drive. A verification pass is performed. FORMAT will not prompt for a system disk after the format operation.

### Possible Error Messages

#### Non-modular system, not compatible with this program.

You are trying to run this program with a previous version of P&T CP/M 2. The program makes use of features found only in the modular version of P&T CP/M 2 and hence cannot run on this system.

#### Bad drive name, please re-enter (0-3 or A-P) :

The letter you entered to specify the logical drive on which to format was not in the range A - P. You should respecify the drive using a valid logical drive letter or a physical drive number.

#### >>> Error: Bad drive name

Same meaning as the previous message but given in the command line mode only.

#### Bad drive number, please re-enter (0-3 or A-P) :

The number you entered to specify the physical drive on which to format was not in the range 0 - 3. You should respecify the drive using a valid physical drive number or a logical drive letter.

#### >>> Error: Bad drive number

Same meaning as the previous message but given in the command line mode only.

#### That drive is not on the system, please re-specify:

You have specified a valid drive designation but the drive is not available on your system. For example, you might have specified drive D on a two-drive system. Re-enter a drive designation using one of the drives that is available on your system.

#### >>> Error: Drive not on system

Same meaning as the previous message but given in the command line mode only.

#### That drive is a hard disk drive, please re-specify:

The drive you have specified is a hard disk drive. FORMAT can work only on floppy disk drives. (Use HFORMAT for hard disks.) You should re-enter a drive designation for one of the floppy drives on the system. If you do not know which logical drives are assigned to the physical floppy drives on the system you can specify a physical drive number.

#### >>> Error: Format target drive is a hard disk drive

Same meaning as the previous message but given in the command line mode only.

#### >>> Error: Missing drive name

Given in the command line mode if the program could not find a specification on the command line of the drive on which to format. Recheck the command line you used to be sure you specified the drive.

#### >>> Error: Bad output designation

Given in the command line mode if you attempted to redirect the console output but the program could not figure out where you wanted it sent. This can be caused by an illegal character in a file name or by ending the command line with a slash (/).

#### >>> Error: Output-file drive same as format target drive

Given in the command line mode if you directed the console output to a disk file on the same drive that is being formatted. This is not allowed because it interferes with the format.

**>>> Error: Bad output-file drive name**

Given in the command line mode if you directed console output to a disk file and used an illegal character (ie. not A - P) to specify the drive on which to place the file.

**>>> Error: Output-file drive not on system**

Given in the command line mode if you directed console output to a disk file on a drive that is not present on the system. Recheck the command line you used to make sure you specified the correct drive.

**>>> Error: Output-file drive cannot be a floppy on a single-floppy system**

Given in the command line mode if you directed console output to a disk file on a floppy diskette on a system that has a single floppy drive. This is not allowed even if disk swapping is in effect to make the single floppy drive appear as if it were multiple drives. Swapping diskettes during the format operation would not only be tedious but also would interfere with the operation.

**>>> Error: Not enough memory to run FORMAT**

This message is displayed if there is not enough memory available for FORMAT to run. It is very unlikely that you will encounter this error. If you do, you should check the amount of memory reserved above the operating system (MENU option LA). Reducing this reserved memory should allow FORMAT to run.

**Please answer with 'Y' for yes or 'N' for no :**

This message is displayed if you answer a yes/no question with an invalid response. Enter your response again using one of the letters specified.

**>>> Error: Drive X: is not ready — cannot format.**

This message indicates that FORMAT found the specified drive not to be ready. This can be caused by the diskette not being mounted, the diskette being mounted backwards, or mounting a double sided diskette on a single sided drive. Correct the problem and run FORMAT again. The "X" is replaced by the logical drive letter of the drive being used for the operation.

**>>> Error: Disk is write protected — cannot format.**

When the program checked the diskette on the specified drive, it found that the diskette was write protected. Since a format operation involves writing to the diskette, the format could not proceed. Check that the diskette is really the one you want to test and, if it is, cover the write protect notch and try again.

**>>> Error: Home error.**

A disk error occurred when the program attempted to restore the read/write head to the home position. This can be caused by an improper head step rate.

**>>> Error: Seek error.**

A disk error occurred when the program was moving the read/write head to a particular location. This can be caused by an improper head step rate.

**>>> Error: Density not specified**

The command line that executed FORMAT in the command line mode did not specify the density at which the diskette was to be formatted. Check your command line to be sure that you specified a density properly.

**>>> Error: Format aborted with <BREAK>.**

This message is displayed if you press the <break> key while the diskette is being formatted. Note that if you interrupt the format operation the diskette probably will not be usable until it is reformatted.

**>>> Error: Check aborted with <BREAK>.**

This message is issued if you abort the verification pass of FORMAT test by pressing the <break> key while the verification is being done. It does not affect the diskette to interrupt the verification, however the whole diskette will not be verified.

**>>> Error: Double-sided disk detected on drive X:**

**>>> Should be formatted at Double-density.**

This message is displayed if you attempt to format a double sided diskette at single density. Either format it at double density or mount a single sided diskette. The "X" will be replaced by the logical drive letter of the drive on which the format was to have been performed.

**>>> Error: Format error, Code = N**

This message indicates that a disk error occurred while attempting to format a track on the diskette. The "N" will be replaced by a single digit number which indicates what type of error occurred. The codes have the following meanings:

- 1 An error occurred reading on side 0. This is probably a media problem.
- 2,5 FORMAT found side 1 when it expected to find side 0. This could be a hardware problem.
- 3 An error occurred reading on side 0. This is probably a media problem.
- 4 FORMAT found side 0 when it expected side 1. This could be a hardware problem.
- 6 A disk error occurred while writing the format information to the diskette. This is usually a hardware problem.

**>>> Error: Formatting error on Track zz, error code = xxxx**

An error has occurred while track "zz" (decimal number) was being formatted. The error code reported by "xxxx" gives further information about the problem. You can use the ERROR utility program (see Section 8.15) to get an explanation of the code.

**>>> Error: Formatting error on Cylinder zz, Side y, error code = xxxx**

Same meaning as the previous message except for a double sided diskette. "zz" and "y" will be replaced by the cylinder and side, respectively, where the error occurred.

**>>> Error: xx Soft read error(s): Track yy, Sector zz**

This message is given when soft errors are encountered during the verification pass. Whenever a sector on the disk cannot be read successfully, the program will attempt to read it again. If a subsequent attempt is successful, each previous failure is called a soft error. In the error message, "xx" will be replaced by the number of times the attempt failed before a successful read was performed. "yy" and "zz" will be replaced by the track and sector numbers, respectively, where the soft errors occurred.

**>>> Error: xx Soft read error(s): Cylinder yy, Side s, Sector zz**

Same meaning as the previous message except that it is used when a double sided diskette is being verified. In the error message, "xx" would be replaced by the number of times the attempt failed before a successful read was performed. "yy", "s", and "zz" would be replaced by the cylinder, side and sector numbers, respectively, where the soft errors occurred.

**>>> Error: Permanent read error: Track yy, Sector zz**

A permanent error has been encountered during the verification pass. A permanent read error is a failure to read a sector on the disk successfully in 10 consecutive attempts. The message reports the track and sector number at which the error took place.

**>>> Error: Permanent read error: Cylinder yy, Side s, Sector zz**

Same meaning as the previous message except that it is used when a double sided diskette is being verified. In the error message, "yy", "s", and "zz" would be replaced by the cylinder, side and sector numbers, respectively, where the permanent error was found.

**>>> Error: Read Time-out error: Track yy, Sector zz**

In some cases, when reading a diskette, if the desired sector cannot be found, the disk controller circuit will try to locate it on several successive revolutions of the diskette before reporting an error. If the sector could not be found on the very first attempt, a problem could exist on the diskette. This message is given if too much time elapses from when the "read" command is given to the disk controller to when the sector was actually read. This indicates that the controller required more than one revolution to find the sector meaning that the sector was missed at least once. This may indicate the very early stages of degradation of a sector.

**>>> Error: Read Time-out error: Cylinder yy, Side s, Sector zz**

Same meaning as the previous message except that it is used when a double sided diskette is being tested. In the error message, "yy", "s", and "zz" would be replaced by the cylinder, side and sector numbers, respectively, where the error was found.

**8.18 Utility name: KXEDIT**

**Purpose:** To modify the character translations performed by the KEYXLATE utility module.

### General Description

KXEDIT allows you to set and change the key code translations that are made by the KEYXLATE module. You also have the option of storing the translations in the BIOSPARM.PNT file on the system disk so that they will be in effect immediately every time the system is loaded.

### Using KXEDIT

The command line to execute KXEDIT is shown in Figure 8.72.

A0>KXEDIT<enter>

Figure 8.72 Command Line to Execute KXEDIT

When KXEDIT begins execution, it shows an editing display as depicted in Figure 8.73. The display shows the translations that are currently in effect. The "FROM" column shows the key codes from the keyboard that are being translated and the "TO" column shows the codes which are substituted for them. In this case, no translations are in effect.

```
1: [ ----- " F R O M " ----- ]      [ ----- " T O " ----- ]
2: # [ key-name   ascii  dec  hex ]      [ key-name   ascii  dec  hex ]
3:
4: 1: [ ..... [ ..... [ ..... [
5: 2: [ ..... [ ..... [ ..... [
6: 3: [ ..... [ ..... [ ..... [
7: 4: [ ..... [ ..... [ ..... [
8: 5: [ ..... [ ..... [ ..... [
9: 6: [ ..... [ ..... [ ..... [
10: 7: [ ..... [ ..... [ ..... [
11: 8: [ ..... [ ..... [ ..... [
12: 9: [ ..... [ ..... [ ..... [
13: 10: [ ..... [ ..... [ ..... [
14: 11: [ ..... [ ..... [ ..... [
15: 12: [ ..... [ ..... [ ..... [
16: 13: [ ..... [ ..... [ ..... [
17: 14: [ ..... [ ..... [ ..... [
18: 15: [ ..... [ ..... [ ..... [
19: 16: [ ..... [ ..... [ ..... [
20:
21: Use "UP" and "DOWN" arrow keys to choose selection.
22: DIRECT mode
23: Press ESC to edit selection.
24: Press ENTER for more options.
```

Figure 8.73 Editing Console Display of KXEDIT

The "FROM" and "TO" columns both consist of four columns, each displaying the key code in a different form. The key name column shows the character or the key that is pressed to generate the key code. The "ascii" column shows the ASCII character, if it is a displayable (ie. non-control) character. The "dec" column shows

the numeric value of the key code in decimal and the "hex" column shows the numeric value of the key code in hexadecimal.

When the program starts a special reverse video cursor will be positioned at the first of the translations. You may select the translation you wish to edit by using the <up arrow> and <down arrow> keys to position the special cursor at any one of the translations. After positioning the special cursor at the translation you want to edit, you should press the <esc> key to start editing. The special cursor will then move to the "FROM" column to indicate that you should enter the key code which is to be translated.

You may enter key codes in three forms. In the "DIRECT" mode, you need merely press the key on the keyboard. This is usually the easiest way to specify the key. In some cases, you may wish to actually enter the numeric code of the key rather than pressing the key. In some cases this is merely a convenience but in some cases (when entering codes in the "TO" column that cannot be generated by the keyboard), it is a necessity. The "DECIMAL" and "HEX" modes allow you to enter the key codes as decimal or hexadecimal numbers, respectively.

When you are entering a code in the direct mode, the bottom five lines of the display are changed as shown in Figure 8.74. Note that you may press <esc> to switch modes. Because of this use of <esc> you may not enter <esc> in the direct mode; you must switch to "DECIMAL" or "HEX".

- 1:
- 2:
- 3: DIRECT mode <-- (press ESC to switch modes)
- 4: Press desired "FROM" key:
- 5:

Figure 8.74 Editing in DIRECT Mode

In the "DECIMAL" mode, the bottom five lines of the display will appear as shown in Figure 8.75.

- 1: DECIMAL mode <-- (press ESC to switch modes)
- 2: Enter desired "FROM" byte in DECIMAL form: (enter 128 to delete)
- 3:

Figure 8.75 Editing in DECIMAL Mode

In the "HEX" mode, the bottom five lines of the display will appear as shown in Figure 8.76.

- 1:
- 2:
- 3: HEX mode <-- (press ESC to switch modes)
- 4: Enter desired "FROM" byte in HEX form: (enter 80 to delete)
- 5:

Figure 8.76 Editing in HEX Mode

After you enter the "FROM" code, the special cursor will move to the "TO" column. You should now enter the key code that you want to replace the code you entered in the "FROM" column. Note that if you want to remove a code from either the "FROM" or "TO" column, you must switch to the "DECIMAL" mode and enter 128 or to the "HEX" mode and enter 80.

Figure 8.77 shows the KXEDIT display after five translations have been specified. In this example, <ctl-> (holding down the control key while pressing apostrophe) has been translated to the accent grave character which is not generated by the keyboard. Also the left and right brackets and braces have been interchanged. This makes the brackets unshifted and the braces shifted (the opposite of the keyboard). You might make this swap if you frequently work on another system where the brackets are lower case or if you use brackets more frequently than braces.

1:	[ ----- " F R O M " ----- ]	[ ----- " T O " ----- ]
2:	# [ key-name    ascii    dec    hex ]	[ key-name    ascii    dec    hex ]
3:		
4:	1: [ CTRL '                         162 A2h ]	[    ` 96 60h ]
5:	2: [ {                                 { 123 7Bh ]	[      91 5Bh ]
6:	3: [ }                                 } 125 7Dh ]	[    ] 93 5Dh ]
7:	4: [ [                                 [ 91 5Bh ]	[    { 123 7Bh ]
8:	5: [ ]                                 ] 93 5Dh ]	[    } 125 7Dh ]
9:	6: [                                    ] .....	[    ] .....
10:	7: [                                    ] .....	[    ] .....
11:	8: [                                    ] .....	[    ] .....
12:	9: [                                    ] .....	[    ] .....
13:	10: [                                   ] .....	[    ] .....
14:	11: [                                   ] .....	[    ] .....
15:	12: [                                   ] .....	[    ] .....
16:	13: [                                   ] .....	[    ] .....
17:	14: [                                   ] .....	[    ] .....
18:	15: [                                   ] .....	[    ] .....
19:	16: [                                   ] .....	[    ] .....
20:		
21:	Use "UP" and "DOWN" arrow keys to choose selection.	
22:	DIRECT mode	
23:	Press ESC to edit selection.	
24:	Press ENTER for more options.	

Figure 8.77 KXEDIT Display After 5 Translations are Specified

After you have finished making all the changes you want, you should press the <enter> key and KXEDIT will present a display as shown in Figure 8.78. In order for KXEDIT to make the translations permanent, it must write them into the BIOSPARM.PNT file that is used when the system is loaded. It looks for this file on logical drive that is the current drive when KXEDIT was executed. For example, if KXEDIT is on logical drive A but logical drive D is the boot drive (this can happen in a hard disk system), you should log onto drive D and then execute KXEDIT from drive A by typing the command "A:KXEDIT". If the BIOSPARM.PNT file cannot be found, Option 2 as shown in Figure 8.78 to make the changes permanent will be omitted from the display and the "Exit" option will become Option 2.

If you choose option 1, you will be returned to the editing display (Figure 8.73) so that you can make additional changes. If you choose option 2, KXEDIT will install the translations in the BIOSPARM.PNT file so that they will be in effect the next time the system is loaded. If you choose option 3, the translations will be installed in the system in memory and you will be returned to the system command level. NOTE: if you want the translations to be permanent you must choose option 2 before exiting with option 3. If you press the <break> key, the program will return to the system command level without installing the translations in the system in memory.

```
1:  
2:  
3:  
4:  
5:  
6:           OPTIONS:  
7:           1 - Do some more editing  
8:           2 - Make changes permanent on System Disk  
9:  
10:  
11:  
12:  
13:           Please enter your choice here:
```

Figure 8.78 KXEDIT Options

### Possible Error Messages

#### No key translator residing in system.

This message is displayed if you execute KXEDIT on a system that does not include the KEYXLATE module. If you want to use KEYXLATE you should use the MODSEL utility program (see Section 6.3) to select the KEYXLATE module for inclusion in the system.

#### >>> Error: Unable to make changes permanent.

This message is displayed if a disk error occurred while attempting to store the current key translation table in the BIOSMODS.PNT file.

**8.19 Utility name: LOAD**

**Purpose:** To convert an Intel HEX format file into an object code file.

**General Description**

The output from ASM and some compilers is an Intel hex format file consisting only of ASCII characters which represent the actual machine language. Because this format was originally intended for use with paper tape, some error detection information is included. In order to execute a program, it is necessary to convert this hex format to normal object code (machine language). LOAD performs this function.

**Using LOAD**

Figure 8.79 illustrates the use of LOAD. The command that executes the routine includes the name of file to be converted, as shown on Line 8.79-1. Note that the extension ".HEX" is assumed and should not be given on the command line. In this example, the command instructs LOAD to load the file named SETTIME.HEX. (The source file SETTIME.ASM must have been previously assembled to generate the hex file).

```
1: A>LOAD SETTIME<enter>
2:
3: FIRST ADDRESS 0100
4: LAST ADDRESS 029C
5: BYTES READ    0173
6: RECORDS WRITTEN 04
7:
8:
9: A>
```

Figure 8.79 Example of Using LOAD

LOAD displays four numbers on the console which provide statistics about the file that was loaded. The first number (Line 8.79-3) gives the first address in memory at which information from the hex file appears. In most cases this number will be 100h, since the loading of all transient programs begins at that address. The second number (Line 8.79-4) shows the last address in memory at which information from the hex file appears. Note that the hex file need not specify information for all of memory between the first and last addresses.

The third number (Line 8.79-5) gives the number of bytes of data that were converted from hex format to binary. In this example, there were 173h bytes of object code in the program. The fourth number (Line 8.79-6) gives the number of logical sectors (128 bytes) written to the output file.

Output from LOAD is stored in a disk file with the same primary name as the one specified on the command line and is given a ".COM" extension. It is assumed that the object code files generated by LOAD will be used primarily as transient programs, so they will be loaded into memory starting at 100h. If the first load address in the hex file is larger than 100h, LOAD adds enough filler to the beginning of the ".COM" file so that the resulting object code will appear at the

correct address when the ".COM" file is loaded. For example, if the lowest load address is 4100h, LOAD will output 16 Kbytes at the beginning of the file.

### Possible Error Messages

#### DISK READ ERROR

An error occurred while reading the input file. It probably indicates a bad file on the disk.

#### DISK WRITE ERROR

An error occurred while writing to the output file. It usually indicates that space has been exhausted on the disk to which the output file is being written.

#### INVALID HEX DIGIT

LOAD has found a character in the input file that is not a valid hex digit or one of the delimiters used in a hex file. This error commonly occurs because of an input file that does not contain hex format data.

#### CHECK SUM ERROR

A check sum error was found in the hex format file as it was being converted to binary format. It probably indicates a bad copy of the hex file.

#### CANNOT OPEN SOURCE

The source file specified in the command line that executed LOAD cannot be found on the specified disk.

#### NO MORE DIRECTORY SPACE

The directory on the disk to which the output file is to be written cannot accommodate another needed directory entry. This can occur when the file is first created or as it grows so large that another physical extent is needed in the directory.

#### CANNOT CLOSE FILE

An error has prevented the output file from being properly closed. This is an unusual error. If it occurs it may indicate a hardware problem. Transfer the needed files to another disk and try again.

#### INVERTED LOAD ADDRESS

LOAD has found a record in the hex file specifying a load address less than the highest load address used to that point. Load addresses within a hex file must be in strictly ascending order.

**8.20 Utility name: MODSEL**

**Purpose:** To select the modules that are to be included in the system when it is loaded.

**General Description**

MODSEL allows you to select the modules that are to be included in the system when it is loaded. This allows you to easily include portions of the system you need (the serial port interface, for example) and leave out those you do not need (perhaps the parallel printer port interface).

MODSEL is described fully in Section 6.3 of this manual. Please refer to that section for details.

**8.21 Utility name: PATCH****Purpose: To install P&T supplied patches to disk files.****General Description**

The PATCH program provides an easy means of installing corrections and modifications to programs supplied with P&T CP/M 2. PATCH accepts simple patch codes and uses them to make changes to a particular program file. When changes are necessary to a P&T CP/M 2 program, you will receive a written copy of the codes which should be entered into PATCH to make the changes.

The patch codes used by PATCH consist of groups of three letters with several of the groups on each line. Since the codes contain letters only they are easy to type and they contain internal checks to insure you type them into PATCH properly. Should an error be discovered in a line you enter, PATCH will report it and ask you to enter it again.

PATCH makes changes "in place" on a disk file. This means that the copy of the file you are working on will be changed. You should always make a backup copy of the file before patching it just in case something goes wrong.

**Using PATCH**

Figure 8.80 shows the command line that executes PATCH.

A&gt;PATCH&lt;enter&gt;

Figure 8.80 Command Line to Execute PATCH

As soon as PATCH begins execution, it will clear the screen and ask you for the name of the file to be patched as shown on Line 8.81-5 of Figure 8.81. You should enter the name of the file you are patching. In this case a file named PGM.COM is specified.

```
1:                               Patch Utility - ver 3.xxx
2:       copyright 1983 Pickles & Trout, all rights reserved
3:
4:
5:   Enter name of file to be patched : PGM.COM<enter>
6:
7:   Enter name of file containing patch-codes
8:   - or -
9:   Just press <ENTER> to type patch-codes in by hand
10:
11:  Patch-codes file : <enter>
```

Figure 8.81 Entering the File to be Patched

In many cases there will be only a limited number of patch codes so there is no trouble entering them directly to PATCH. In some cases, however, the changes may be fairly extensive and you might want to use an editor to create a text file

containing the patch codes. This allows you to check the codes yourself before having PATCH install them. In addition, this allows you to easily and quickly install the patches on different copies of the file being patched. Lines 8.81-7 to 8.81-11 allow you to enter the name of a text file containing the patch codes. If you have created such a file, you should enter its name on Line 8.81-11. In this example, the patch codes will be entered directly to PATCH so just <enter> is pressed.

Figure 8.82 shows the console display presented by PATCH while you enter the patch codes. Line 8.82-5 shows the name of the file being patched. Lines 8.82-7 to 8.82-10 remind you of how to enter the patch codes. Starting on Line 8.82-12, PATCH prompts you for the patch codes one line at a time. While you are entering the patch codes, all non-alphabetic characters are ignored. Lower case letters are automatically converted to upper case and PATCH automatically inserts a space after every three characters.

```
1:           Patch Utility - ver 3.xxx
2:           copyright 1983 Pickles & Trout, all rights reserved
3:
4:
5: File being patched : PGM.COM
6:
7: Type in appropriate patch-codes at each Line prompt.
8: Press <ENTER> at the end of each Line.
9: Each Line of patch-codes is checked for typo's when <ENTER> is pressed.
10: If any errors are detected you will be re-prompted for that Line.
11:
12: Line 1 --> TAC VAB RAC AAD BDC DHD .. ok ..
13: Line 2 --> DDE QAD AAE BEC EDE ELN .. ok ..
14: Line 3 --> EFE GZU PIC BEP .. ok ..
15:
16: Patch successfully installed in PGM.COM
17:
18: A>
```

Figure 8.82 Entering Patch Codes by Hand

When you have entered all the patch codes for one line, press the <enter> key. PATCH will then check the line for errors and display and, if none are found, will display ".. ok .." at the end of the line. If an error is found, PATCH will display an error message and prompt you to enter the line again. You should double check the characters you entered against the patch notice you are working from to find the errors.

After you enter the last line of the patches, PATCH will install the changes in the specified file. It displays the message shown on Line 8.82-13 after the changes have been made and then returns to the command level of the system.

Figure 8.83 shows the console display while PATCH reads the patch codes from a text file. In this example, the text file PGM.PAT contains the patch codes. Note that there is no user interaction while PATCH reads the patch code file; everything proceeds automatically.

```
1:           Patch Utility - ver 3.xxx
2:           copyright 1983 Pickles & Trout, all rights reserved
3:
4:
5: File being patched : PGM.COM
6:
7: Reading in patch-codes file PGM.PAT
8:
9: Line 1 --> TAC VAB RAC AAD BDC DHD .. ok ..
10: Line 2 --> DDE QAD AAE BEC EDE ELN .. ok ..
11: Line 3 --> EFE GZU PIC BEP .. ok ..
12:
13: Patch successfully installed in PGM.COM
14:
15: A>
```

Figure 8.83 Reading Patch Codes from a File

### Possible Error Messages

**xxxxxxxx was originally named yyyyymm**

**Is this the right file to patch ? (y/n) :**

This is not strictly an error message. It is displayed if the file you are trying to patch has been renamed. The "xxxxxxxx" in the message will be replaced by the current file name and the "yyyyymm" will be replaced by the original file name. Since patch notices will refer to the original name of the file, this message gives you a chance to check that you are installing the patch on the correct file.

**Press <BREAK> to quit, press any other key to try again :**

This message is displayed after other error messages to give you the option of exiting the program or re-entering the information that resulted in an error.

**Exit here? (y/n) :**

This message is displayed when you press the <break> key (except in response to the previous message). An affirmative response will return you to the command level of the system while a negative response will return you to PATCH at the point you pressed <break>.

**<- Bad file name**

This message is displayed if a file name you specified is invalid. This can be caused by a file name that contains illegal characters or an illegal drive specification (ie. not in the range A - P).

**<- Error in this Line.**

This message is displayed at the right end of a line of patch codes in which an error was detected.

**Something is wrong with the patch codes you have entered.**

**One or more of the lines has an error that went undetected earlier.**

**Please check each line carefully.**

Although it is very unlikely, it is possible that the line-by-line checking that is done on the patch codes will miss a typing error. To guard against this, PATCH makes a check encompassing all of the patch codes that have been entered. If this check fails, at least one error slipped through the line-by-line checking.

After this message has been displayed, PATCH will show you each of the lines you have entered and ask you if it is OK. Carefully check each line against the patch notice you are working from. If you indicate that a line is not OK (ie. respond negatively), PATCH will prompt you to re-enter the line. You may press <break> during this process to abort PATCH and return to the command level of the system.

**Can't find file to be patched fffffff.eee**  
PATCH displays this message if it cannot find the file you specified as the file to be patched. "ffffff.eee" will be replaced by the name you specified. Check that you entered the name correctly and specified the correct drive.

**ffffff.eee is an empty file: Can't install patch.**

This message indicates that you have asked PATCH to make changes to a file that has zero length. There is something wrong with the file you specified.

**ffffff.eee has internal error(s): Can't install patch.**

If the file being patched has been set up to be verified with the VERIFY program, PATCH first verifies that the file has no errors before making changes. This message indicates that errors were detected in the file. Getting another copy of the file from the master disk should solve the problem.

**This patch is not meant to be installed in fffffff.eee**

PATCH has discovered something amiss with the file that you are attempting to modify. This usually indicates that you are trying to patch the wrong file or that the file has been corrupted in some way. "ffffff.eee" will be replaced by the name of the file you are trying to patch.

**This patch is already installed in fffffff.eee**

If PATCH finds that the changes specified by the patch codes have already been made to the file, it will display this message. "ffffff.eee" will be replaced by the name of the file you are trying to patch.

**Patches n, n, n must be installed  
before this patch can be installed.**

Each patch is given a number which is recorded in a file when the patch is applied to the file. In some cases the order in which patches are installed may be important. For this reason, PATCH has the ability to check for other patches already having been installed. This message is given if certain patches are required to have already been installed but were not present in the file. The "n"s will be replaced by the numbers of the patches that are required. You should install these patches before attempting to install the patch you have just entered.

**Error while closing fffffff.eee**

This message indicates that an error occurred when the file being patched was closed. The "ffffff.eee" will be replaced by the file name. The file may or may not still be usable after this error occurs.

**Error in Line nn of patch-codes file.**

This message is displayed if PATCH finds an error in one of the lines of patch codes read from a text file containing patch codes. The "nn" will be replaced by the number of the line on which the error was detected.

**Previously undetected error(s) somewhere in patch-codes file.**

Although it is very unlikely, it is possible that the line-by-line checking that is done on the patch codes will miss a typing error. To guard against this, PATCH

makes a check encompassing all of the patch codes that have been read from the file. If this check fails, at least one error slipped through the line-by-line checking. You should double check the entire file to find the error.

**Unexpected end of patch-codes file.**

This message is given if the end of the text file containing the patch codes encountered before all of the patch codes are read. This typically results from not entering all of the lines of patch code in the text file.

**8.22 Utility name: PIP**

**Purpose:** To transfer (copy) information between peripherals on the system.

**General Description**

PIP is a utility program for transferring data from one peripheral device to another. Virtually all of the peripheral devices on the system may be accessed by PIP, but the routine is most commonly used for transfers from disk to disk or disk to printer. Some processing may be performed on the data as it is transferred, including concatenation of two or more files.

**Using PIP**

There are two types of command lines for executing PIP, as illustrated in Figure 8.84. The first type, shown on Line 8.84-1, allows you to enter all commands for PIP operations on a single line. Line 8.84-2 shows the second type, which causes PIP to prompt you for further commands by displaying an asterisk (\*). In this mode of operation, PIP will continue to accept commands either until a blank command line is entered or the <break> key (or <ctrl-C>) is pressed.

1: A>PIP (commands)<enter>  
2: A>PIP<enter>

Figure 8.84 Command Lines to Execute PIP

The PIP commands have the general form shown in Figure 8.85. Note that the spaces shown in Figure 8.85 (e.g. between "dest" and "=") are for clarity only; they are not required. In an actual command, "dest" would be replaced by the destination of the PIP operation, which may be either a disk file or non-disk output device. "srcA," "srcB," etc., would each be replaced by a source of the data to be transferred. Each of these would be a disk file or non-disk input device. Additional processes to be performed may be specified optionally following each source indication by enclosing the option letters in square brackets.

dest = srcA[optA] , srcB[optB] , ... , srcZ[optZ]<enter>  
for example: LST:=FILE1[N],FILE2

Figure 8.85 General Form of PIP Commands

It should be noted that no PIP operations literally "move" a file from one disk to another. When the source of data in a PIP operation is a disk file, the original file will be left unchanged. In other words, PIP will simply copy its contents to the specified destination. However, if "dest" names a disk file and a file by that name already exists (on the destination drive), the previously existing file will be removed upon the successful completion of the transfer. This removal is automatic in most cases. You will be prompted for permission to remove it only if the file is marked as "read only" (R/O) and you have NOT specified PIP's W option (discussed below). The W option causes PIP to make the deletion without asking permission.

The "src" and "dest" parameters in the PIP commands may be disk files, logical I/O devices, physical I/O devices, or certain special I/O devices. A unique file name (unqfn) may be used for any "src" or "dest". In some cases, a wildcard file name (wcfn) may be used as a "src".

In addition to the general form of PIP commands shown in Figure 8.85, several shorthand forms of PIP commands are allowed, as illustrated in Figure 8.86.

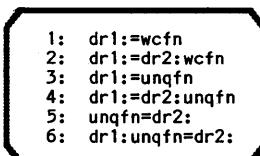


Figure 8.86 Shorthand Forms of PIP Commands

Line 8.86-1 shows a shorthand form that will transfer all files matching the wildcard file name from the current default disk to the disk specified by "dr1". (Note: "dr1" would be replaced by a single letter drive name in an actual command line.) The shorthand form in Line 8.86-2 will transfer all files matching the wildcard file name from "dr2" to "dr1". Neither "dr1" nor "dr2" are required to be the current default drive. These are the only two forms of PIP commands that allow wildcard file names. Note that no file names are specified for the "dest" parameter; the transferred files will be stored with the same name they had on the source disk.

Lines 8.86-3 to 8.86-6 show shortened PIP commands that transfer a specific file from one disk to another. Note that the file name must be specified only once, since it will be stored on the destination drive under the same name. The commands on Lines 8.86-3 and 8.86-4 are similar to those on Lines 8.86-1 and 8.86-2 except that they specify a particular file to be transferred. In fact, these commands are simply special cases of the other two, because a wildcard file name can always be replaced by a unique file name.

A unique file name may also be specified as the "dest" in shortened PIP commands, as shown in Lines 8.86-5 and 8.86-6. The command on Line 8.86-5 will transfer the file named by the "unqfn" from the drive specified by "dr2" to the current default drive. A similiar command on Line 8.86-6 will transfer the specified file from "dr2" to "dr1".

With one exception, you cannot specify the same file name on the same drive for both source and destination in a PIP command. Doing so will cause PIP to report an "INVALID FORMAT" error. The exception is that files may be transferred between users on the same disk, as explained below under the G option.

Any of the CP/M non-disk logical I/O devices may be specified as a source or a destination. In this case, the logical device's input or output is directed to the physical device assigned to it by the operating system. This assignment may be changed by the ASSIGN, SETUP and STAT utilities, as described in Sections 8.5, 8.27, and 8.28 of this manual, or by programs. The logical device names recognized by PIP are shown in Figure 8.87.

CON:	(input and output)	The currently assigned console device; typically the system keyboard and video display.
LST:	(output only)	The currently assigned system printer.
RDR:	(input only)	The currently assigned reader device.
PUN:	(output only)	The currently assigned punch device.

Figure 8.87 Logical I/O Devices Available to PIP

PIP can also reference non-disk physical I/O devices directly. Unfortunately, the physical device names recognized by PIP have little correspondence to the actual physical devices available on the TRS-80 Model II/12/16. (PIP recognizes only the physical device names of generalized CP/M. Many of the other utilities of P&T CP/M 2 recognize names that have been customized to the Model II/12/16, as discussed in Section 5.7.) Figure 8.88 shows the physical device names that PIP recognizes and their corresponding devices on the Model II/12/16.

name used by PIP	actual physical device
TTY:	serial port A (input and output)
CRT:	system keyboard and video display (input and output)
UC1:	serial port B (input and output)
UR2:	serial port B (input only)
UP2:	serial port B (output only)
UL1:	serial port B (output only)
LPT:	centronics port (output only)
PTR:	user reader device 1 if installed (input only)
UR1:	user reader device 2 if installed (input only)
PTP:	user punch device 1 if installed (output only)
UP1:	user punch device 2 if installed (output only)

Figure 8.88 Physical I/O Device Names Used by PIP

Additional capabilities of the PIP routine are provided by access to special I/O devices. In this way, PIP can perform functions and/or access I/O devices that are not available with the standard system configuration. The names of these special devices are shown in Figure 8.89. More information on the patching of user supplied input and output routines into PIP is presented later in this section.

name	function
NUL:	Sends 40 NUL characters (ASCII code 0) to the destination. It was implemented primarily for use with paper tape punches for generating leader and trailer sections. NUL: may be used as a source only.
EOF:	Sends a CP/M end of file character (<ctrl-Z>) to the destination. Note that PIP automatically sends a <ctrl-Z> to the destination at the end of any ASCII transfer. EOF: may be used as a source only.
PRN:	Sends characters to the system printer (LST: logical device). As the file is transferred, tabs are expanded to every eighth column, all lines are numbered, and form feeds are inserted after every 60 lines. A form feed is sent to the printer before the transfer to position the paper to the top of a new page. The function of PRN: is the same as LST: with the [t8np] options. PRN: may be used as a destination only.
INP:	Accepts characters from a user supplied input routine that is patched into PIP. INP: may be used as a source only.
OUT:	Sends characters to a user supplied output routine that has been patched into PIP. OUT: may be used as a destination only.

Figure 8.89 PIP Special I/O Devices

Figure 8.90 gives several examples of PIP commands with a brief explanation of each.

B:=*.COM	Transfers all files with the extension COM from the current default drive to drive B.
A:=C:INVTRY.*	Transfers all files with the primary name INVTRY from drive C to drive A.
B:=PIP.COM	Transfers the file PIP.COM from the current default drive to drive B.
LST:=FILE1.TXT	Transfers the file FILE1.TXT from the current default drive to the system printer (logical device LST:). Note that no special processing (such as tab expansion) is performed on the file unless options are specified.
TTY:=B:FILE1.TXT	Transfers the file FILE1.TXT from drive B to serial port A.
UC1:=FILE1.TXT,NUL:	Transfers the file FILE1.TXT from the current default drive to serial port B and appends 40 nulls to it.

Figure 8.90 Examples of PIP Commands

The example in Figure 8.91 shows the use of PIP to transfer several files. First (Line 8.91-1) PIP is executed with a command line which includes the PIP command to concatenate (append one to another) two files and place them in a file on disk called TEST.1. PIP is then executed again (Line 8.91-3) but with no PIP commands on the command line. In this case PIP prompts for commands and three separate command lines are entered from the console (Lines 8.91-4 to 8.91-6). Since a wildcard file specification is used on Line 8.91-6, PIP reports the filenames of that match the specification as it copies the files (Lines 8.91-8 to 8.91-11). Finally the <break> key is pressed to return to the operating system.

```
1: A>PIP B:TEST.1=SETUP.COM,SETMISC.COM[V]<enter>
2:
3: A>PIP<enter>
4: *B:=SETUP.COM[V]<enter>
5: *B:TEST.1=SETUP.COM,SETMISC.COM[V]<enter>
6: *B:=*.ASM[V]<enter>
7:
8: COPYING -
9: DUMP.ASM
10: SETTIME.ASM
11: TIME.ASM
12: *<break>
13:
14: A>
```

Figure 8.91 Example of Using PIP

### Processing Options of PIP

PIP provides several processing options that may be used to modify the data being transferred. These are specified by single option characters enclosed within square brackets on the command line (see Figure 8.85). When arguments are needed, they immediately follow the option character. Any numeric arguments are always entered as decimal numbers. Also, several options may be specified within one set of brackets. In this case, the arguments may be separated from each other by spaces if desired, even though spaces are not necessary. Figure 8.92 gives a summary of the options available in PIP.

B	Use block transfer mode.
Dn	Delete characters which extend past column n on each line.
E	Echo all characters transferred to the console.
F	Remove all form feeds from the data as it is transferred.
Gn	Get the file from user number n.
H	Transfer a hex format file while checking for errors.
I	Ignore :00 records in a hex format file (automatically sets H option).
L	Convert upper case characters to lower case during transfer.
N	Prefix each line transferred by its line number. Leading zeroes are suppressed and the number is followed by a colon.
N2	Prefix each line transferred by its line number. Leading zeroes are printed and the number is followed by a tab character.
O	Binary file transfer - Ignore <ctrl-Z> as end of file mark.
Pn	Insert a form feed after every n'th line with an initial form feed.
Qs<ctrl-Z>	Stop the transfer operation when the string "s" is encountered.
R	Read files with the SYS attribute set. Unless this option is used PIP will not find files with the SYS attribute set.
Ss<ctrl-Z>	Begin transferring when the string "s" is found in the source data.
Tn	Expand tab characters to every n'th column with spaces.
U	Convert lower case letters to upper case during transfer.
V	Read back data from the destination to verify that there were no errors in the copy operation. Destination must be a disk file.
W	Do not request permission to delete disk files which are set to R/O and have the same name as the specified destination file.
Z	Clear (set to 0) the high order (parity) bit of each byte transferred.

Figure 8.92 Summary of PIP Options

## B option

This option causes PIP to store data from the source device and save it in a memory buffer until a <ctrl-S> character is received from the source. Upon receiving the <ctrl-S>, PIP will transfer the data to its destination (typically a disk file) and begin to accept characters from the source again. (Note: There is no interaction between PIP and the source device; the source device must wait until PIP has completed the transfer to the destination before beginning to send data again.) The amount of data which can be buffered in memory depends on the size of the system in which PIP is operating.

## Dn option

The Dn option causes PIP to truncate all lines after the "nth" column. All characters of any line that fall beyond the specified column will be ignored. This function is useful in transferring a file with long lines to a device (such as a printer) that cannot handle them.

## E option

This option causes all data bytes transferred to be echoed to the system console. It is useful as a monitor of activity as the transfer takes place. However, care should be taken not to use the E option when transferring binary data, since it may cause strange things to happen on the console display. Also note that using the E option will slow down the transfer somewhat due to the time required for the display.

## F option

The F option causes PIP to remove all form feeds from the data as it is transferred. This function is most useful when transferring files containing form feeds to a printer. The F and P options may be used in combination to remove existing form feeds and insert new ones at different locations within the file. (This action may be needed, for example, to print a file on a different sized paper.)

## G option

Usually, PIP permits only the disk files under the current user number to be involved in data transfer (as source or destination). The G option allows a file to be retrieved from another user number and stored under the current user number. A transfer of this type is the one case where PIP will let you specify the same disk drive for both the source and destination files.

Note that PIP has no option which allows files to be stored under a specific user number. This implies that PIP itself must already be resident under the user number to which the files are to be transferred. Immediately, the question of how to store PIP under a new user number presents itself. There is only one answer: read PIP into memory, switch to the new user number, and then save it back on the disk (under the new number). This procedure requires the USER and SAVE commands of CP/M as well as the DDT program.

In the example shown in Figure 8.93, PIP is transferred to users 3 and 5. In other situations, PIP can be transferred to as many users as desired merely by repeating Lines 8.93-7 and 8.93-8 for each user number. You should keep in mind, however, that a new copy of PIP is stored and more disk space is taken up each time you perform this operation. Clearly, the need to maintain separate copies of commonly used programs can limit the usefulness of user numbers on floppy disk based systems.

Note that the Pickles & Trout Advanced Command Processor (P&T ACP) for CP/M 2 can eliminate many of the problems involved in using the different user areas on a disk. The P&T ACP is available as a separate product; contact your dealer or Pickles & Trout for further details.

```
1: A>USER 0<enter>
2: A>DDT PIP.COM<enter>
3: DDT VERS 2.2
4: NEXT PC
5: 1E00 0100
6: ~<break>
7: A>USER 3<enter>
8: A>SAVE 31 PIP.COM<enter>
9: A>USER 5<enter>
10: A>SAVE 31 PIP.COM<enter>
11: A>USER 0<enter>
12: A>
```

Figure 8.93 Example of Transferring PIP to Other User Numbers

### H option

The H option informs PIP that an Intel hex format file is being transferred. PIP then checks all records within the file for errors and determines whether they have the proper hex format. Any non-essential characters between records are removed during the transfer. If PIP should find an error, it will report it on the system console and wait for operator action.

If an error is reported while a disk file is being transferred, only two option are open to you. By pressing the <enter> key, you will cause PIP to continue with the next record in the source. Or, you can type <ctl-Z> to abort the transfer. If you continue the transfer, the record containing the error will not be transferred to the destination.

In a few cases, it is possible to "rewind" the source device to a location previous to the error and try again. This is a holdover from the days of paper tape (perish the thought) when reading errors were all too common. After the source device is rewound and <enter> is pressed, PIP will begin reading again. If the error was a transitory one, the transfer might be successfully completed.

Error checking is not automatically performed on files whose names have the extension "HEX". If you want the format of a hex file checked, you must use the H option. A simple way to perform this operation is to use the H option while transferring the file to the CON: logical device. Since the console is made the destination of the transfer, another disk file is not created.

### I option

This option causes PIP to assume that the file being transferred is a hex file and to remove any \$0 records (since they contain no meaningful data) from it. In doing so, it automatically turns on the H option (described above).

### L option

The L option causes all upper case letters to be converted to lower case as the transfer is made.

## N and N2 options

Both cause PIP to number the lines of a file as it is transferred. The five-digit line numbers are placed at the beginning of each line starting at one and incrementing by one for each line transferred. These options can be useful when using an editor (such as ED) that uses line numbers in the editing process.

If the N option is specified, the leading zeroes of the line numbers are replaced by spaces and the line number is followed by a colon and a tab character. The characters on the line immediately follow the tab. If the N2 option is specified, the leading zeroes of the line number are printed and a tab character (only) separates the line number and the characters on the line.

## O option

The O option forces PIP to treat the file as a binary (object) file. PIP will not stop transferring data upon finding a <ctl-Z> character, as it would with an ASCII file; it will continue until encountering the physical end of file. All files with the "COM" extension are automatically treated as binary files.

## Pn option

The P option causes PIP to insert a form feed character into the file after every "nth" line as it is being transferred. A form feed character is also inserted as the first character sent to the destination. If n is specified as 1 or is not given, form feeds are inserted after every 60 lines. If the P option is used in combination with the F option, the form feeds are removed from the source data before new ones are inserted by the P option.

## Qs<ctl-z> option

The Q option causes PIP to stop the transfer of data after encountering a specified string (denoted by "s"). A <ctl-Z> character must be entered to terminate the string, and the string will be included in the data transferred. If the Q option is included in a command line that executes PIP (as shown on Line 8.84-1), all of the characters in the string will be converted to upper case. If the Q option is used while PIP is prompting for commands, the string will not be converted to upper case.

## R option

PIP normally does not access files that have been marked with the system attribute (SYS). If such files (which would not be included in directory listings) are to be transferred, the R option allows PIP to access them.

## Ss<ctl-z> option

The S option causes PIP to search the source data for the string specified (denoted by "s"). When the string is found, PIP will begin to transfer data, with the string included, to the destination. A <ctl-Z> character must be entered to terminate the string. If the S option is included in a command line that executes PIP (as shown on Line 8.84-1), all of the characters in the string will be converted to upper case.

If the S option is used while PIP is prompting for commands, the string will not be converted to upper case.

The S and Q options can be used together to extract a section of a file by specifying both the starting and ending strings.

### Tn option

The T option causes PIP to expand tab characters into multiple spaces so that the effective tab stops occur in every "nth" column. This option is particularly useful when transferring a text file to a printer or other device that does not know how to handle tabs.

### U option

The U option causes PIP to convert all lower case letters into upper case as the data is transferred.

### V option

The V option may be used only when the destination is a disk file. It causes PIP to read back the data written to the file and check that it was written correctly. If a discrepancy is found in the re-reading of the data, an error is reported. Transfers with the V option require somewhat more time than those without it due to the extra steps involved.

### W option

When the destination for a transfer is a disk file and there is a file of the same name already on the destination disk, PIP will ordinarily delete the existing file. However, if that file's "read only" attribute has been set, PIP will prompt you for permission before making the deletion so long as the W option is not used. If you do not grant permission, PIP will abort the transfer and then return to the operating system or prompt for another command. If permission is granted, the existing file will be deleted and replaced by the result of the transfer.

The W option causes PIP to ignore the "read only" attribute of files on the destination disk and delete any files whose names have been duplicated by a transferred file. In other words, PIP will not request your permission to make the deletion; it will occur regardless of whether the file is set to "read only" or not. This action can be useful in transferring a number of files without operator interaction, but it can also be risky to use. Files are usually set to "read only" status to prevent their modification. Since the W option removes this protection, it should be used sparingly and carefully.

### Z option

The Z option instructs PIP to clear the high order bit (bit 7; sometimes called the parity bit) on each data byte as it is transferred. This can be useful when transferring a file that might contain non-ASCII characters to a device (such as a printer) that might not understand them.

### Special Notes About PIP

<ctl-Z> is the standard CP/M end of file mark for all files containing ASCII characters. PIP always looks for a <ctl-Z> to mark the termination of ASCII data, even when the source is a non-disk device. For example, if the CON: device is used as a source, any characters typed on the console keyboard will be transferred to the destination device. PIP will continue to wait for more characters from the console until a <ctl-Z> is typed, which PIP recognizes as the end of the data.

When using the RDR: device as a source, it is possible to terminate the reading of data by typing a <ctl-Z> at the system console. Unfortunately, this feature has some limitations. PIP checks the console keyboard for a character between the reading of any two characters from the RDR: device. If a character is available from the keyboard, PIP reads it and checks for <ctl-Z>. If it is a <ctl-Z>, PIP acts as if it had read a <ctl-Z> from the RDR: and stops reading data from it.

However, when PIP requests a character from the RDR: device, the system will not return control to PIP until a character from the RDR: device is available. When no characters are coming in from the RDR: device, PIP will lose control of the computer as soon as it requests a character from the RDR: device. Since PIP loses control of the computer under this circumstance, it will not be able to check the console for characters. As a result, input from the RDR: device can be interrupted from the console only if data is actually being received. If reception of data has stopped, the system will wait forever for the next byte to arrive from the RDR:. To overcome this, you must append a <ctl-Z> to the end of the file being transferred. If the file is coming from another computer, you may append a <ctl-Z> by using the EOF: special I/O device. Figure 8.94 shows a command line that executes PIP to transfer a file to the PUN: device and append a <ctl-Z>.

PIP PUN:=FILE.TXT,EOF:

Figure 8.94 Example of Appending <ctl-Z> to a File

When files are concatenated (more than one source file is specified in the PIP command), PIP ordinarily assumes that they contain ASCII characters and that each file is terminated with a <ctl-Z>. When PIP encounters the first <ctl-Z> in a file, it stops reading. If non-ASCII files are being concatenated, the O option should be used. As noted above, it forces PIP to ignore <ctl-Z>'s and continue to the physical end of file for each file in the list.

When transferring data to a non-disk I/O device with PIP, any character typed at the console keyboard will abort the transfer. Obviously, this action prevents you from entering more PIP commands during the transfer by typing ahead. However, when PIP is transferring to a disk file, it cannot be interrupted until the transfer is completed, so it is possible to type ahead during disk transfers.

When PIP discovers an error and aborts an operation, it will also erase any \$\$\$SUB file on drive A. This has the effect of terminating a SUBMIT file that is in operation.

### User Supplied I/O Routines for PIP

PIP provides space from 103h to 1FFh for special user-supplied I/O routines to be patched in. When PIP wants to read a character from the user-supplied input routine (INP:), it makes a CALL to location 103h. After making this CALL, PIP expects the input data to be in location 109h when it regains control from the

user's routine. Note: the data should have its high order bit set to zero. When PIP sends a character to the user-supplied output routine (OUT), it makes a CALL to location 106h with the character residing in register C.

The user-supplied I/O routines can be inserted into PIP by using the assembly feature of DDT or by writing a small assembly language program and patching it into PIP with DDT. Figure 8.95 shows such an assembly language program for a hypothetical I/O device.

```
; example user I/O routines for PIP

stat equ 40h ;status port for I/O device
data equ stat+1 ;data port for I/O device
statin equ 1 ;mask for data input ready bit
statout equ 2 ;mask for data output ready bit

org 103h
jmp input ;jump to input routine
jmp output ;jump to output routine

retbyt: db 0 ;place to return input data

input: in stat ;read device status
ani statin ;check for data ready
jz input
in data ;read input character
ani 7fh ;clear the high order bit
sta retbyt ;save in return area
ret ;return to PIP

output: in stat ;read device status
ani statout ;check if can send data out
jz output
mov a,c ;get the data byte
out data ;output it
ret ;return to PIP

end
```

Figure 8.95 Example of User Supplied I/O Drivers for PIP

Figure 8.96 shows the steps taken to assemble the I/O routines shown in Figure 8.95 and install them into PIP. This example assumes that the routines shown in Figure 8.95 are stored in a file named PPIO.ASM on drive A. It also assumes that the programs PIP, ASM, and DDT are also available on drive A. Note that the patched version of PIP is saved in a file named PIPP.COM. In this way, the original version of PIP is not erased and will be available if needed.

```
A>ASM PIP10.AAA<enter>
CP/M ASSEMBLER - VER 2.0
0124
000H USE FACTOR
END OF ASSEMBLY

A>DDT PIP.COM<enter>
DDT VERS 2.2
NEXT PC
1E00 0100
-IPI10.HEX<enter>
-R<enter>
NEXT PC
1E00 0100
-<break>
A>SAVE 31 PIPP.COM<enter>
A>
```

Figure 8.96 Example of Installing User I/O Routines in PIP

### Possible Error Messages

#### DISK READ ERROR

An error has occurred during the reading of a disk file. This message is encountered very rarely, and it may indicate a hardware problem.

#### DISK WRITE ERROR

An error has occurred while data was being written to a disk file. It most commonly indicates that available space has run out on the destination disk. When this message is encountered, check the available space with STAT. If it is low or zero, delete unwanted files from the disk or move to another disk and try again.

#### VERIFY ERROR

If the V option is specified for a disk to disk copy, PIP rereads the destination file and checks for errors. This message indicates that PIP has found a discrepancy while performing this check. PIP will abort the transfer if a verify error occurs.

#### NOT A CHARACTER SOURCE

An "output only" logical or physical device (such as PUN: or PTP:) has been specified as a source.

#### INVALID USER NUMBER

In using the G option, a user number has been specified which is not in the range of 0 to 15.

#### RECORD TOO LONG

A record that is too long has been found within a hex file whose contents are being checked by the H option of PIP. (Any single record within a hex file can represent no more than 255 bytes of binary data.)

#### INVALID DIGIT

In processing a file with the H option, PIP has found a character that is not a valid hex digit (0-9 and A-F) in one of its records. Most often, this error occurs because the processed file does not contain hex format data or it contains a

corrupted record (possibly caused by an attempt to patch the file with a text editor).

#### CHECKSUM ERROR

In processing a file with the H option, PIP has found a discrepancy between the checksum of a record and the checksum computed from the record's data.

#### CORRECT ERROR, TYPE RETURN OR CTL-Z

In processing a file with the H option, PIP has found an error of some kind and is waiting for you to specify its next action. If you press the <enter> key on the console, PIP will continue the transfer with the next record from the source; it will not transfer the record in which the error was found. If you type <ctl-Z>, PIP will immediately end the transfer.

In the (unlikely) event that the source device you are using can be rewound to a location prior to the record in which the error was found, you may rewind it and press <enter>. PIP will resume reading data from the source device, ignoring any records it has already correctly received. If the error was of a transitory nature, PIP may read the data record properly the second time.

#### INVALID FORMAT

You have made some error in the command line. It is likely that you used a wild card file name illegally or that you specified an invalid file or drive name. (An invalid drive name is given, for example, if you inadvertently type a semi-colon in place of a colon after a drive designation.)

#### NO DIRECTORY SPACE

In attempting to write to a disk, PIP has found that the directory was full. Typically, this will occur when PIP is trying to open a file, but it may also arise when PIP needs to extend the length of a file (thus causing a need for another directory entry).

#### NO FILE

PIP could not find the file of the specified file name on the drive designated.

#### START NOT FOUND

PIP could not find the string that was specified with the S option in the source data. Check the option specification to be sure the string is correct.

#### QUIT NOT FOUND

PIP could not find the string that was specified with the Q option in the source data. Check the option specification to be sure the string is correct.

#### CANNOT CLOSE DESTINATION FILE

An error has occurred as PIP attempted to close an output disk file. This message is somewhat unusual and may indicate hardware problems.

#### DESTINATION IS R/O, DELETE (Y/N)?

A file already exists on the destination drive with the name that was specified for the destination file, and it is set to "read only" status. If you respond with Y, PIP will delete the file and replace it with the one being transferred. If you respond with N, PIP will not make the requested transfer. The W option of PIP allows you to override the sending of this message, so that you will not be asked for permission to delete "read only" files whose names have been duplicated. (The deletion will then occur automatically.)

**REQUIRES CP/M 2.0 OR NEWER FOR OPERATION**

The version of PIP supplied with P&T CP/M 2 must be used with CP/M 2. You are trying to use it on an older version of the operating system.

**CANNOT WRITE**

The command line you have entered into PIP is invalid because it instructs PIP to write to an "input only" device, such as RDR:

**CANNOT READ**

The command line you have entered into PIP is invalid because it instructs PIP to read from an "output only" device, such as LST:

**8.23 Utility name: SETCCB**

**Purpose:** To set the date and time of the P&T CCB clock calendar board if one is installed in the system.

#### General Description

The function of the SETCCB program relates entirely to the P&T CCB board, which provides a clock, calendar, and audio alarm (bell) when installed in a TRS-80 Model II/12/16. If this board is not installed in your system, the SETCCB utility is of no use to you.

When a CCB board is installed, a number of convenient functions are available. The clock and calendar of the CCB are powered by a lithium pacemaker battery when the computer is turned off, thus allowing accurate date and time to be constantly maintained. P&T CP/M 2 reads this information automatically when the system is loaded, so the user is freed from the necessity to enter it at the console. In addition, P&T CP/M 2 will sound the audio alarm on the board whenever an ASCII BEL character is sent to the system video display, and the board may be accessed directly by programs as needed.

SETCCB allows you to set the date and time of the calendar and clock on the CCB. Although the battery allows the clock to continue operation when the computer is turned off, some periodic correction of the time reading may be necessary. Also, you may want to change the clock setting to reflect the changes between daylight and standard time. The CCB will automatically correct for the extra day in February of a leap year provided that P&T CP/M 2 is loaded at least once between January 1 and February 28 of that same year. If the system is not loaded in this period, you will need to correct the date manually using SETCCB.

#### Using SETCCB

SETCCB is executed by the command line shown in Figure 8.97.

A>SETCCB<enter>

Figure 8.97 Command to Execute SETCCB

SETCCB first clears the console display and displays the current date and time settings of the CCB, as shown below in Figure 8.98 (Lines 8.98-1 to 8.98-8). Note that the time shown on Line 8.98-6 will be running, i.e. it will be updated as the time reading of the CCB clock changes.

```
1: CCB Time and Date Set Utility - ver 1.xxx
2: Copyright (c) 1983 Pickles & Trout
3: all rights reserved
4:
5: CCB date Tuesday January 9, 1908 (1/9/08)
6: CCB time 14:28:21
7:
8: ----- Press BREAK to quit -----
9:
10:
11: Do you want to change the date? Y<enter>
12:
13: Enter date as MM/DD/YY 1/6/82<enter>
14: Enter day of week (0=Sun,1=Mon,2=Tue,3=Wed,4=Thu,5=Fri,6=Sat): 3<enter>
15:
16: The date you have entered is Wednesday January 6, 1982 (1/6/82)
17: Is this correct (Y/N)? Y
```

Figure 8.98 Example of Setting the CCB Date with SETCCB

SETCCB will then ask you if you want to change the date setting of the CCB (Line 8.98-11). If you answer affirmatively (as shown in Figure 8.98), SETCCB will ask you to enter the date (Line 8.98-13). As illustrated in the example, you do not have to enter two digits where one will suffice. In addition, any non-numeric character (such as a space, period, or comma) may be used to separate the numbers of the date. However, only one separating character is allowed. Note: SETCCB does not check the maximum day number for each month. It will, for example, accept February 30 as a valid day.

After the date has been entered, SETCCB will ask you for the day of the week (Line 8.98-14). Enter the number corresponding to the desired day which is given on the display. SETCCB will then report back to you the date you have entered (Line 8.98-16) and ask you to confirm that it is correct (Line 8.98-17). If you respond with an "N" on Line 8.98-17, SETCCB will ignore the date information you have just entered and start over again on Line 8.98-11. If you respond with a "Y", SETCCB will set the CCB calendar to the date you have specified, update the date shown at the top of the screen, and return to Line 8.98-11. In either case, you are asked again if a date change is desired. If your response is negative, SETCCB will ask if you want to change the time setting of the CCB clock.

Figure 8.99 shows how the CCB time is set.

```
1: CCB Time and Date Set Utility - ver 1.xxx
2: Copyright (c) 1983 Pickles & Trout
3: all rights reserved
4:
5: CCB date Wednesday January 6, 1982 (1/6/82)
6: CCB time 13:28:42
7:
8: ----- Press BREAK to quit -----
9:
10:
11: Do you want to change the date? N<enter>
12: Do you want to change the time? Y<enter>
13:
14: Enter current time in 24 hour format (hh:mm)
15: NOTE: seconds are always set to 0
16: 13:32<enter>
17: Hit enter to set clock. <enter>
```

Figure 8.99 Example of Setting the CCB Time with SETCCB

As in this example, the time setting may be changed without altering the date merely by responding negatively to the query on Line 8.99-11. SETCCB then asks you to enter the time in 24 hour format and reminds you that the seconds are always set to 0 when the time is revised (Lines 8.99-14 and 8.99-15). In our example, 1:32 PM is entered as "13:32" (Line 8.99-16). Once again, either the hour or the minute may be entered as a single digit number if the value is in the range of 0 to 9. Also, any non-numeric character may be used to separate the two numbers.

If you are synchronizing the CCB to another time source, you should enter the next minute. For example, if the current time is 10:02 and 37 seconds, you should indicate the time of 10:03. Once your response is entered, SETCCB will prompt you to press the <enter> key again, and the clock will not be set until you do so. Now you can wait until the precise moment to establish the synchronization (press <enter> at precisely 10:03). SETCCB will then update the time shown at the top of the screen and return to Line 8.99-11.

Figure 8.100 shows the exit from the SETCCB program. After all desired settings have been made, simply depress the <break> key when asked again if you wish to change the date (Line 8.100-11). SETCCB will ask you if you want the system time synchronized to the CCB (Line 8.100-13). If you respond with a negative answer, SETCCB will return immediately to the operating system without affecting the system date and time. (The CCB will retain the values you entered, however.) If you respond affirmatively on Line 8.100-13 (as shown in the example), SETCCB will set the system date and time to agree with the current settings of the CCB before returning to the system.

```
1:          CCB Time and Date Set Utility - ver 1.xxx
2:          Copyright (c) 1983 Pickles & Trout
3:          all rights reserved
4:
5:  CCB date  Wednesday January 6, 1982  (1/6/82)
6:  CCB time  13:32:33
7:
8:  ----- Press BREAK to quit -----
9:
10:
11: Do you want to change the date? <break>
12:
13: Do you want the system time synchronized to the CCB? Y<enter>
14: A>
```

Figure 8.100 Example of Returning to P&T CP/M 2 from SETCCB

### Possible Error Messages

**There is no CCB installed in this computer  
or you have not used SETMISC to set the CCB-port address**

This error message indicates that the port number for the CCB board has not been changed from its default value of 254, which informs SETCCB (and the rest of the system) that no CCB board is installed in the system. If one is installed, you need to use the SETMISC utility program (see Section 8.25) or system MENU option CA (see Chapter 4) to set the port number appropriately. Also, you will need to make the change permanent by using system MENU option FR. See the CCB manual to determine which port number to use.

**Day of month is out of range**

The day of the month you have specified is not in the range of 1 to 31. Note: SETCCB does not check the maximum day number for each month. It will, for example, accept February 30 as a valid day.

**Month of year is out of range**

The number you have entered for the month of the year is not in the range of 1 - 12.

**Year is out of range**

SETCCB will accept years in the ranges 0 - 99 and 1900 - 1999. The number you have entered is not in these ranges.

**You have not entered all 3 numbers**

You have not entered all three of the numbers required to specify the date. This message may occur if all three numbers are entered but are separated by more than one non-numeric character.

**Day of week is out of range**

The number you have entered to indicate the day of the week is not in the range of 0 - 6.

**Hours must be less than 24, please re-enter:**

The number you have entered for the hour of the time is not in the range 0 - 23. Note that midnight is "00" hours, not "24" hours. Re-enter the entire time, specifying the hour with a valid number.

**Minutes must be less than 60, please re-enter:**

The number you have entered for the minute in the time setting is not in the range 0 - 59. Re-enter the entire time, specifying the minutes with a valid number.

**Seconds must be less than 60, please re-enter:**

The number you have entered for the seconds in the time setting is not in the range 0 - 59. Note that although SETCCB will ignore any number you enter for seconds since it always sets the seconds to 0, it will allow you to enter a non-zero number for seconds. Re-enter the entire time, specifying the seconds with a valid number.

**You must enter the time, press <break> to quit**

Once you have told SETCCB that you want to enter the time, you must enter it. If you really do not want to enter the time, press <break> to abort the program. You can then run it again and respond negatively when it asks if you want to set the time.

**Y or N only:**

You have responded to a yes/no question with a letter other than "Y", "y", "N", "n". Re-enter your response using one of these letters. (Note: If you enter several characters, only the first one is considered. For example, "ycrud" will be interpreted as an affirmative response.)

**8.24 Utility name: SETDATE**

**Purpose:** To set the system date from the console.

**General Description**

SETDATE allows you to set the system date from the console keyboard. If no P&T CCB board is installed in your computer, the system will load with the date set to 00/00/00. When the correct system date is of importance to you (i.e. used by one of your programs), it must then be set by SETDATE or by a program. (The special system function which performs this task is discussed in Chapter 16 of this manual.)

SETDATE has no effect on a CCB board which is installed in the computer; it may be used to set the system date to an incorrect value for testing or other purposes without modifying the date kept by the CCB. The next time the system is loaded or the utility program SYNCRO is run, the system date and time will again be set to the CCB.

**Using SETDATE - Interactive Mode**

The example in Figure 8.101 illustrates the use of SETDATE.

```
1: A>SETDATE<enter>
2:
3:
4:           Utility to Set System Date - Ver 2.xxx
5:           copyright 1980,83 (c) Pickles & Trout
6:           all rights reserved
7:
8: Enter date: Jan 4, 1983<enter>
9:
10: The date you have entered is Tuesday January 4, 1983      (1/4/83)
11: Is this correct (Y/N)? N<enter>
12:
13: Enter date: 1/2/83<enter>
14:
15: The date you have entered is Sunday January 2, 1983      (1/2/83)
16: Is this correct (Y/N)? Y<enter>
17: A>
```

Figure 8.101 Example of Using SETDATE

SETDATE is executed by the command line given on Lines 8.101-1. After it displays its opening messages (Lines 8.101-4 to 8.101-6) it asks you for the date (Line 8.101-8). You may enter the date in a fairly free form. You may specify the date as all numbers (e.g. 12/5/82) or you may type out the name of the month.

If you enter the date as all numbers, you must enter it in the following order: month, day, year. The year may be entered as two digits or four digits. If you enter it as 4 digits, only years between 1900 and 1999 are accepted. Entering the date with the name of the month gives you have a little more latitude. In this case, you may enter the date with either the month or the day first. Figure 8.102 shows some examples of forms in which the date may be entered.

May 13, 1972	13 May 72
13 May 1972	5 13 72
5/13/1972	5.13.72
5/13/72	5:13:1972

All of these are valid designations  
for May 13, 1972.

Figure 8.102 Examples of Forms in Which Dates can be Entered to SETDATE

On Line 8.101-8 the date Jan 4, 1983 is entered. SETDATE includes a 20th century calendar which allows it to find out that this date falls on a Tuesday. On Line 8.101-10 SETDATE shows you the date you entered and the day of the week it falls on and, on Line 8.101-11, asks you to verify that the date is correct.

If you respond affirmatively, SETDATE will set the system date accordingly and return you to the command level of the system. If you respond negatively, as shown on Line 8.101-11, SETDATE will ask you again for the date. This example shows a second date entered as 1/2/83 (Line 8.101-13) and its being verified as correct (Line 8.101-16).

Since SETDATE has a built in calendar, you may use it to find the day of the week for any day in the 20th century. If you have entered a date just to find out the day of the week and do not want the system date set, just press <break> to return to the system command level with no action.

### Using SETDATE - Command Line Mode

SETDATE also has a command line mode where you specify the date to which the system date is to be set on the command line that executes SETDATE. The form of the command line for SETDATE is shown in Figure 8.103.

SETDATE (date)

Figure 8.103 General Form of Command Line for SETDATE

The date specified on the command line may be in any of the forms that can be entered in the interactive mode. Figure 8.104 shows some examples of command lines that might be used with SETDATE.

SETDATE OCT 12, 1978  
SETDATE 11/18/82  
SETDATE 5 DEC 72

Figure 8.104 Examples of Command Lines for SETDATE

### Possible Error Messages

#### >>> Error: Day of month is out of range.

The day of the month you have specified is not valid for the month specified.  
Remember 30 days hath September....

#### >>> Error: Month of year is out of range.

The number you have entered for the month of the year is not in the range of 1-12.

**>>> Error: Year is out of range.**

SETDATE will accept years in the ranges 0 - 99 and 1900 - 1999. The number you have entered is not in one of these ranges.

**>>> Error: Invalid entry.**

The line you have entered for the date does not make sense for some reason. Perhaps you mis-spelled the month or left out part of the date.

**8.25 Utility name: SETMISC****Purpose:** To set miscellaneous I/O parameters from the console.**General Description**

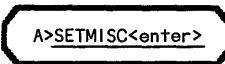
SETMISC provides a means of altering a number of I/O parameters of P&T CP/M 2 directly from the console keyboard. These parameters take effect immediately and will remain in effect until further modification by SETMISC or until the system is reloaded. In order to make the parameters permanently resident on the system diskette, System MENU option FR should be used (see Chapter 4 of this manual).

SETMISC has both an interactive and a command line mode. In the interactive mode, you will be shown the current values of the parameters that SETMISC can alter and allowed to change them at will. The command line mode of SETMISC accepts instructions from the command line that executes SETMISC and changes the I/O parameters accordingly. The command line mode is useful when you know exactly what you want to do and don't want to be bothered with prompts. It also allows SETMISC to be used from a SUBMIT file or a menu system.

**Using SETMISC - Interactive Mode**

When entering responses into SETMISC, you may hit the <break> key at any time to bring an immediate return to the operating system. If you press only the <enter> key as a response, it will be interpreted as a "0".

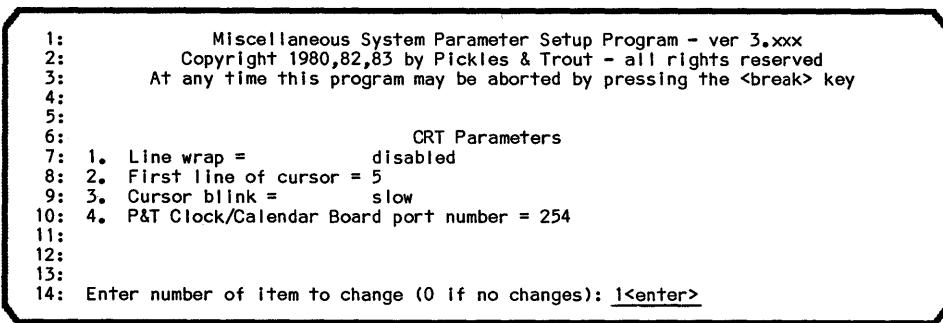
SETMISC is executed by the command line shown in Figure 8.105.



A>SETMISC<enter>

Figure 8.105 Command Line to Execute SETMISC

SETMISC begins by clearing the console display and showing the first set of parameters available for modification, as shown in Figure 8.106. Note that the line shown as Line 8.106-14 will actually appear at the bottom of the console display.



```
1:           Miscellaneous System Parameter Setup Program - ver 3.xxx
2:           Copyright 1980,82,83 by Pickles & Trout - all rights reserved
3:           At any time this program may be aborted by pressing the <break> key
4:
5:
6:           CRT Parameters
7:   1. Line wrap =      disabled
8:   2. First line of cursor = 5
9:   3. Cursor blink =    slow
10:  4. P&T Clock/Calendar Board port number = 254
11:
12:
13:
14:  Enter number of item to change (0 if no changes): 1<enter>
```

Figure 8.106 Display of CRT Parameters in SETMISC

The first five lines shown in Figure 8.106 will remain on the screen at all times while SETMISC is running. The remaining lines on the display will change as various sets of parameters are inspected and/or modified.

As the figure indicates, the CRT parameters are the first to be displayed. The current value of each parameter is shown (Lines 8.106-7 to 8.106-10), and you are prompted for the number of an item that you want to change (Line 8.106-14). If you enter "0" (or just press <enter>) at this point, SETMISC will go on to the next set of parameters.

Entering a "1" on Line 8.106-14 indicates that you wish to modify the setting of the line wrap option. If this option is enabled, the cursor on the video display will automatically go to the beginning of the next line when it reaches the end of a line. Likewise, it will move to the end of the preceding line when it is moved backwards from the beginning of a line. If this option is disabled, the cursor will stop at the ends of a line and not move to another line unless an explicit command (such as line feed) is given.

After entering a "1" on Line 8.106-14, the bottom line of the display will be replaced with the line shown in Figure 8.107. If you enter a zero at this point, no changes will be made. Entering a one will disable line wrap, and entering a "2" will enable it. Once your response is entered, the system line wrap parameter will be set and Line 8.106-7 updated to reflect your choice.

Enter 0->no change, 1->disable, 2->enable 2<enter>

Figure 8.107 Prompt Line for Altering Line Wrap Option

Entering a "2" on Line 8.106-14 indicates your wish to modify the type of cursor that is used on the video display. Each character on the video display is composed of 10 rows of dots (numbered 0 at the top to 9 at the bottom). To form the cursor, one or more of these rows are displayed in reverse video. You may modify the number of rows which are used by the cursor by specifying the line on which the reverse video will begin. The reverse video portion will always begin on the line you specify and end on the bottom line (line 9) of the character cell. You may modify the cursor from a single underscore (starts on line 9) to a complete block (starts on line 0). (Note: The system's default setting begins the cursor on line 5.) The change in the cursor is made immediately so you can see the result of the choice you have made. If you do not like it, you may choose a different cursor size.

After entering "2" on Line 8.106-14, the bottom line of the console display will be replaced with the one shown in Figure 8.108. In response, you should enter the number of the line on which you want the cursor to begin. The cursor size will then be changed, Line 8.106-8 will be updated to reflect the new value, and you will be prompted for a new item number.

Enter character line on which cursor should begin (0-9): 8<enter>

Figure 8.108 Prompt Line for Altering Cursor Size

Entering a "3" on Line 8.106-14 indicates that you wish to modify the cursor blink rate. You may turn the cursor off entirely, cause it not to blink, or set it to blink at a slow or fast rate. (The system defaults to the slow rate, which is about half as fast as the fast rate.)

After entering "3" on Line 8.106-14, the bottom line of the console display will be replaced by the one shown in Figure 8.109. In response, you may enter a zero (or press <enter>) to make no change, a "1" to turn off the cursor (not really very useful), a "2" if you want the cursor not to blink at all, a "3" if you want the slow blink rate, or a "4" if you want the fast blink rate. The cursor will then be

immediately modified, Line 8.106-9 will be updated to reflect your choice, and you will be prompted for a new item.

Enter 0->no change, 1->cursor off, 2->no blink, 3->slow blink, 4->fast blink : 0<enter>

Figure 8.109 Prompt Line for Altering Cursor Blink

Entering a "4" on Line 8.106-14 indicates your desire to modify the port number at which a P&T CCB clock/calendar/bell board is addressed. All CCB's are shipped with their base address set to port 190. You may change the port setting by means of a DIP switch on the CCB board (see the CCB manual for details). If the CCB port number is set to 254 (the system default value), the system assumes that no CCB is present. Any other port number will indicate that a CCB is installed in the computer at the base port number specified. If you have a CCB you must use SETMISC to change this parameter so that the system can access it.

After entering a "4" on Line 8.106-14, the bottom line of the display will be replaced with the one shown in Figure 8.110. You should respond by entering the base port number at which your CCB is addressed. The number will then be installed in the system I/O drivers, Line 8.106-10 will be updated to reflect your entry, and you will be prompted for a new item.

Enter base port number for the CCB board (254->no board present): 190<enter>

Figure 8.110 Prompt Line for Altering CCB Base Port Address

If you enter a "0" (or press <enter>) on Line 8.106-14, SETMISC will display the next set of I/O parameters, as shown in Figure 8.111. Once again, the line shown as 8.111-14 will actually appear at the bottom of the console display.

1: Miscellaneous System Parameter Setup Program - ver 3.xxx  
2: Copyright 1980,82,83 by Pickles & Trout - all rights reserved  
3: At any time this program may be aborted by pressing the <break> key  
4:  
5:  
6: Drive step rates in milliseconds  
7: 1. drive A: 10  
8: 2. drive B: 10  
9: 3. drive C: 10  
10: 4. drive D: 10  
11:  
12:  
13:  
14: Enter number of item to change (0 if no changes): 2<enter>

Figure 8.111 Display of Drive Step Rates in SETMISC

The rate at which the computer attempts to move the read/write head of a diskette drive is referred to as the drive step rate. The floppy disk controller in the computer is capable of step rates of 15, 10, 6, and 3 milliseconds, and SETMISC allows you to choose a step rate individually for each diskette drive on the system. The system defaults to 10 msec step rates for each of the diskette drives (as specified by Tandy). A faster rate results in faster operation, and it frequently reduces the noise as head is moved. However, not all drives are capable of a step rate faster than 10 msec. It is your responsibility to determine if your drive will operate at the rate which you specify. The thinline drives used in the Models 12 and 16 will run at 3 msec, the fastest rate. If you set the step rate too fast, you will typically start getting SEEK errors.

To change the stepping rate of a disk drive, enter the number corresponding to the drive as given on the display (Line 8.111-14). The bottom line of the display will then be replaced by the one shown in Figure 8.112. Now enter a "0" (or press <enter>) to leave the rate unchanged, a "1" for a step rate of 3 msec, a "2" for a step rate of 6 msec, a "3" for a step rate of 10 msec, or a "4" for a step rate of 15 msec. Once your response is entered, it will be installed in the I/O drivers of the operating system, the appropriate line of Figure 8.111 will be updated, and you will be prompted for a new item.

Enter new step rate (0->no change, 1->3ms, 2->6ms, 3->10ms, 4->15ms): 2<enter>

Figure 8.112 Prompt Line for Altering Drive Step Rate

If you enter a "0" (or press <enter>) on Line 8.111-14, you will proceed to the third (and last) set of options, those for the parallel printer port. Note that if a parallel printer module has not been included in the system, this display will be skipped and you will go directly to the prompt shown in Figure 8.119.

These options were included in the system for the owners of Radio Shack printers which were designed to be used with the TRSDOS operating system. The current values of the options will be displayed as in Figure 8.113. As before, the line shown as Line 8.113-14 actually appears at the bottom of the console display.

```
1:           Miscellaneous System Parameter Setup Program - ver 3.xxx
2:           Copyright 1980,82,83 by Pickles & Trout - all rights reserved
3:           At any time this program may be aborted by pressing the <break> key
4:
5:
6:           Centronics Options
7:   1. Suppress extra line feeds: disabled
8:   2. Emulate form feed:      disabled
9:   3. Do auto form feed:    disabled
10:  4. Paper length (in lines): 66
11:  5. Number of lines before auto form feed: 60
12:
13:
14:
15: Enter number of item to change (0 if no changes): 1<enter>
```

Figure 8.113 Display of Parallel Port Options in SETMISC

Entering a "1" on Line 8.113-14 indicates your desire to modify the line feed suppression option of the parallel printer port driver software. When enabled, this option attempts to produce normal printed output on Radio Shack printers by removing unnecessary line feeds. (Radio Shack printers automatically advance the paper when sent a carriage return character.) When disabled, all line feeds are sent to the port.

After entering "1" on Line 8.113-14, the bottom line of the display will be replaced by the one shown in Figure 8.114. You may now enter "0" (or press <enter>) if you do not want to make a change, "1" if you want to disable the option, or "2" if you want to enable the option. The new value (if any) of the option will then be installed in the operating system, Line 8.113-7 will be updated, and you will be prompted for a new item.

Enter 0->no change, 1->disable, 2->enable 2<enter>

Figure 8.114 Prompt Line for Altering Items 1, 2, and 3 of Parallel Port Options

Entering a "2" on Line 8.113-14 indicates your wish to modify the form feed emulation option of the parallel printer port driver software. This option is needed with printers that do not respond to a form feed character (as is the case with some Radio Shack printers). When enabled, this option attempts to keep track of the current paper location. Upon receiving a form feed character, it then sends what it considers to be the exact number of line feed characters to the port that will cause the paper to advance to the next page. Since this option requires the software to keep track of the current location on the page, it is possible for it to be fooled in some cases (like sending an escape sequence containing a line feed character). When the option is disabled, all form feed characters are sent directly to the port.

After entering "2" on Line 8.113-14, the bottom line of the display will be replaced by the one shown in Figure 8.114. You may now enter "0" (or press <enter>) if no change is desired, "1" if you want to disable the option, or "2" if you want to enable the option. The new value (if any) of the option will then be installed in the operating system, Line 8.113-8 will be updated, and you will be prompted for a new item.

Entering a "3" on Line 8.113-14 allows you to modify the auto form feed option of the parallel printer port driver software. When enabled, this option will insert a form feed character into the output data after a specified number of lines have been sent. See option 5 below to change the number of lines printed on the page. (The form feed character will be expanded into line feeds if the form feed emulation option is enabled. The "auto form feed" will function with printers that do not respond to a form feed character only if "emulate form feed" is also enabled.) After your "3" is entered, the bottom line of the display will be replaced by the one shown in Figure 8.114. You may now enter "0" (or press <enter>) for no change, "1" to disable the option, or "2" to enable the option. The new value (if any) of the option will then be installed in the operating system, Line 8.113-9 will be updated, and you will be prompted for a new item.

Entering a "4" on Line 8.113-14 indicates your desire to modify the paper length used by the parallel printer port driver software for form feed emulation. The bottom line of the display will then be replaced by the one shown in Figure 8.115. You may now enter the page length in lines; the default setting is 66 (11 inch paper at 6 lines per inch). After you have entered the number, it will be installed in the operating system, Line 8.113-10 will be updated, and you will be prompted for a new item.

Enter page length in number of lines (0-254): 33<enter>

Figure 8.115 Prompt Line for Altering Parallel Port Page Length

You would enter a "5" on Line 8.113-15 if you wish to modify the number of lines printed before a form feed character is inserted by the auto form feed option (when enabled). The line shown in Figure 8.116 will then appear at the bottom of the display, and you may enter the desired number of lines. (The default setting is 60.) After your number is entered, it will be installed, Line 8.113-11 will be updated, and you will be prompted for a new item.

Enter number of lines to print before auto form feed (0-254): 30<enter>

Figure 8.116 Prompt Line for Changing Number of Printed Lines per Page

Finally, you should enter a "0" (or press <enter>) on Line 8.113-14 when no changes - or no further changes - are desired among the parallel printer port option settings.

The line shown in Figure 8.117 then appears at the bottom of the display. You may now enter a "0" (or press <enter>) to return to the operating system or a "1" to start over at the beginning of SETMISC.

Enter 0 to return to CP/M, 1 to restart SETMISC: 0<enter>

Figure 8.117 Prompt Line for Returning to Command Level of System

### Using SETMISC - Command Line Mode

In the command line mode, all instructions are given to SETMISC on the command line which executes the program. SETMISC accepts a wide variety of instructions in an attempt to provide fairly free form input. In most cases there is a long form of the instruction and one or more short forms. If you use a certain instruction frequently, you will probably want to learn one of the short forms since they usually involve considerably less typing. The long form of the instructions is fairly English-like and should be easy to remember.

You may type as many instructions as you like on a command line (subject to the normal command line limitation of 128 characters).

In the following descriptions of the instructions, the general form of each instruction is shown as well as some specific examples. In describing the general form, these conventions are used:

1. When one of several words must be used, the words will be enclosed in angle brackets. For example <OFF, ON, SLOW, FAST> indicates that one of the words "OFF", "ON", "SLOW", or "FAST" should be used.
2. <num> indicates that a decimal number should be entered.
3. <enabled> is equivalent to <ENABLED, EN, ON, +, 1> meaning that one of the words "ENABLED", "EN", "ON", "+", or "1" should be used.
4. <disabled> is equivalent to <DISABLED, DIS, OFF, -, 0> meaning that one of the words "DISABLED", "DIS", "OFF", "-", or "0" should be used.
5. The space between a word or phrase (e.g. FLC) and the following number is required.

LINE WRAP <enabled, disabled>  
LW <enabled, disabled>

Figure 8.118 Instructions for Setting Console Line Wrap

Some examples are:

**LINE WRAP ENABLED** - enables line wrap.

**LINE WRAP OFF** - disables line wrap.

**LW ON** - enables line wrap.

CURSOR <OFF, ON, SLOW, FAST>  
CB <OFF, ON, SLOW, FAST>

Figure 8.119 Instructions for Setting Console Cursor Blink

Some examples are:

**CURSOR ON** - sets a steady cursor (no blink).

**CURSOR SLOW** - sets the cursor to slow blink.

**CB SLOW** - sets the cursor to slow blink.

FIRST LINE OF CURSOR <num>  
FIRST LINE CURSOR <num>  
FLC <num>  
FL <num>

Figure 8.120 Instructions for Setting First Line of Console Cursor

Some examples are:

**FIRST LINE OF CURSOR 0** - sets first line of cursor to 0 (full block).

**FIRST LINE CURSOR 9** - sets first line of cursor to 9 (underscore).

**FLC 5** - sets first line of cursor to 5 (half block).

**FL 8** - sets first line of cursor to 8 (two line cursor).

CCB PORT <num>  
CCB <num>

Figure 8.121 Instructions for Setting CCB Port Number

Some examples are:

**CCB PORT 190** - sets the CCB port number to 190.

**CCB 16** - sets the CCB port number to 16.

STEP RATE <drive> <3, 6, 10, 15> <drive> <3, 6, 10, 15> ...  
STEPRATE <drive> <3, 6, 10, 15> <drive> <3, 6, 10, 15> ...  
RATE <drive> <3, 6, 10, 15> <drive> <3, 6, 10, 15> ...  
SR <drive> <3, 6, 10, 15> <drive> <3, 6, 10, 15> ...

<drive> should be replaced with a logical drive letter or a physical drive number.

Figure 8.122 Instructions for Setting Drive Step Rates

Some examples are:

**STEP RATES A 3,B 6,C 10** - sets A to 3 msec, B to 6 msec, C to 10 msec.

**STEPRATE A 6,B 6,C 6,D 6** - sets drives A, B, C, and D to 6 msec.

**RATE B 3** - sets drive B to 3 msec.

**SR C 15 D 15** - sets drive C and D to 15 msec.

SUPPRESS EXTRA LINE FEEDS <enabled, disabled>  
SUPPRESS LINE FEEDS <enabled, disabled>  
SUPPRESS XLF <enabled, disabled>  
SXLF <enabled, disabled>  
SELF <enabled, disabled>

Figure 8.123 Instructions for Setting Line Feed Suppression

Some examples are:

**SUPPRESS EXTRA LINE FEEDS ENABLED** - enables line feed suppression.  
**SUPPRESS LINE FEEDS OFF** - disables line feed suppression.  
**SUPPRESS XLF EN** - enables line feed suppression.  
**SXLF DIS** - disables line feed suppression.  
**SELF +** - enables line feed suppression.

EMULATE FORM FEEDS <enabled, disabled>  
EMULATE FORMFEEDS <enabled, disabled>  
EFFS <enabled, disabled>  
EFF <enabled, disabled>

Figure 8.124 Instructions for Setting Form Feed Emulation

Some examples are:

**EMULATE FORM FEEDS ENABLED** - enables form feed emulation.  
**EMULATE FORMFEEDS ON** - enables form feed emulation.  
**EFFS OFF** - disables form feed emulation.  
**EFF ON** - enables form feed emulation.

AUTO FORM FEEDS <enabled, disabled>  
AUTO FORMFEEDS <enabled, disabled>  
AUTO FFS <enabled, disabled>  
AUTO FF <enabled, disabled>  
AFF <enabled, disabled>

Figure 8.125 Instructions for Setting Auto Form Feed

Some examples are:

**AUTO FORM FEEDS DISABLED** - disables auto form feed.  
**AUTO FFS ON** - enables auto form feeds.  
**AFF OFF** - disables auto form feeds.

PAPER LENGTH <num>  
PAGE LENGTH <num>  
PPL <num>  
PL <num>

Figure 8.126 Instructions for Setting Paper Length

Some examples are:

**PAPER LENGTH 66** - sets paper length to 66 lines.  
**PAGE LENGTH 88** - sets paper length to 88 lines.  
**PL 33** - sets paper length to 33 lines.

NUMBER OF LINES PER PAGE <num>  
NUMBER OF LINES/PAGE <num>  
LINES PER PAGE <num>  
LINES/PAGE <num>  
NL <num>

Figure 8.127 Instructions for Setting Printed Lines per Page

Some examples are:

**NUMBER OF LINES PER PAGE 58** - sets auto form feed after 58 lines.  
**LINES/PAGE 60** - sets auto form feed after 60 lines.  
**NL 30** - sets auto form feed after 30 lines.

Some examples of command lines for SETMISC that use several instructions follow:

#### SETMISC FLC 8, CURSOR SLOW, LINE WRAP OFF

Sets the first line of the cursor to line 8 (a double underscore cursor), sets the cursor to slow blink, and disables the line wrap on the console display.

#### SETMISC STEPRATE:A 3,B 3,C 6 ; SUPPRESS LFS ON

Sets the step rate on drives A and B to 3 msec, the step rate on drive C to 6 msec, and enables line feed suppression on the parallel printer port.

### Possible Error Messages

#### Non-modular system, not compatible with this program.

You are trying to run this program with a previous version of P&T CP/M 2. The program makes use of features found only in the modular version of P&T CP/M 2 and hence cannot run on this system.

#### Number was out of range, please re-enter.

This message is displayed in the interactive mode if the number you entered is not in the range of valid responses to the prompt. Re-enter your response using a number from the specified range.

---- ?

In the command line mode, if an instruction on the command line does not make sense to SETMISC, it will display the word it could not understand followed by a question mark. In this case none of the instructions on the command line will take effect. You should correct the mistake and type the line again.

#### >>> Error: No parallel printer module in system.

This message indicates that you have attempted to set parameters associated with the parallel printer module when such a module has not been included in the system. If you want to use the parallel printer port you should run the MODSEL program (Section 6.3) and select one of the available parallel printer modules for inclusion in the system.

**8.26 Utility name: SETTIME**

Purpose: To set the system time-of-day clock from the console.

**General Description**

SETTIME allows you to set the system time of day clock from the console keyboard. The clock is kept by the system and must be set each time the system is loaded (RESET), either by SETTIME or by a program using a special system function (as discussed in Chapter 16 of this manual) if you wish to make use of it. SETTIME has no effect on a P&T CCB clock/calendar board installed in the system, so temporary changes can be made in the system time while the CCB maintains its timekeeping.

If you have a Pickles & Trout CCB clock board installed in your system, P&T CP/M 2 will automatically read the board and set the system time of day when the system is loaded

**Using SETTIME - Interactive Mode**

The use of SETTIME is illustrated in Figure 8.128. The program is executed by the command given on Line 8.128-1. After displaying its opening messages (Lines 8.128-4 to 8.128-6), it prompts you for the new time setting (Line 8.128-8). You should enter the time in hours:minutessseconds format using 24 hour time. The numbers specifying the hours, minutes, and seconds may each consist of one or two digits, and they may be separated by commas, colons, and/or one or more spaces.

After you have entered the time, SETTIME will prompt you to press the <enter> key to set the clock (Line 8.128-9). The clock will not be set until you press <enter>, thus allowing you to synchronize the system clock to an external time source. SETTIME then returns directly to the operating system. (Note: Pressing the <break> key at any time will abort SETTIME without affecting the system time of day clock.)

```
1: A>SETTIME<enter>
2:
3:
4:                               Utility to Set System Time - Ver 2.xxx
5:                               copyright 1980,83 (c) Pickles & Trout
6:                               all rights reserved
7:
8: Enter current time (hh:mm:ss): 17:39:10<enter>
9: Hit enter to set clock.
10: A>
```

Figure 8.128 Example of Using SETTIME

**Using SETTIME - Command Line Mode**

In the command line mode of SETTIME you specify the time to which the system clock is to be set on the command line that executes the program. SETTIME sets the system time to the specified time immediately after it begins execution. Note that, since it may take a few seconds to load and execute SETTIME, close

synchronization with an external time source is more difficult than in the interactive mode.

Figure 8.129 shows the general form of the command line to set the system clock with SETTIME.

SETTIME (time)

Figure 8.129 Form of the Command Line for SETTIME

The following examples show command lines that might be used with SETTIME.

**SETTIME 9:30:00** - sets the system clock to 9:30 AM.

**SETTIME 15:04:25** - sets the system clock to 3:04:25 PM.

### Possible Error Messages

#### >>> Error: Hours out of range (0-23)

Given when the hours of the time you specified are not acceptable. Note that midnight is 0 hours.

#### >>> Error: Minutes out of range (0-59)

Given when the minutes of the time you specified are not acceptable. Re-enter the time using a value for minutes in the range 0 to 59.

#### >>> Error: Seconds out of range (0-59)

Given when the seconds of the time you specified are not acceptable. Re-enter the time using a value for seconds in the range 0 to 59.

**8.27 Utility name: SETUP**

**Purpose:** To alter the serial port configuration and the IOBYTE assignments.

### General Description

The SETUP utility program provides a visual representation of the current state of the serial port configurations and the system IOBYTE assignments. It also allows you to change the serial port configurations (for their use with different peripheral devices) and the logical-to-physical I/O device assignments which are controlled by the IOBYTE. Note that these assignments can also be changed with the ASSIGN utility program.

SETUP includes a command line mode that allows you to change the serial port configurations by typing a single command line. This feature may be useful if you frequently change the serial port configurations or need to change them from a SUBMIT file or a menu system.

### Using SETUP - Interactive Mode

Figure 8.130 shows the command line which will execute the SETUP program.



Figure 8.130 Command Line to Execute SETUP

SETUP first clears the screen and displays the available serial port configurations and logical-to-physical device assignments, as shown in Figure 8.131. Those which are currently in effect are highlighted in reverse video.

When SETUP first begins, the cursor will be located at the upper leftmost option available (110 baud on port A). To change a particular option, merely move the cursor (using the cursor arrow keys right, left, up, or down) to the desired value and press <hold>. The previous value for that option will be returned to normal video and the value you have selected will be changed to reverse video. The cursor can only be positioned to locations on the screen where valid option selections are shown.

When you are finished, simply press <esc> to exit the program. Any changes you have specified will not take effect until this time. If you want to abort the program without making any changes, type <break>.

Each serial port has 6 lines of parameters displayed. The first line (Lines 8.131-3 and 8.131-11) shows the baud rates that may be selected for the port. This should be set to agree with the peripheral which is connected to the port. The second line (Lines 8.131-4 and 8.131-12) shows the number of stop bits that are transmitted at the end of each character. This setting should also agree with the peripheral device connected to the port, but it is always acceptable to specify a greater number of stop bits than the peripheral expects. For example, all serial peripherals should be operable if 2 stop bits are specified. However, data transmission will be slowed by about 10% if an unneeded stop bit is transmitted.

```
1:          10 SETUP PROGRAM v3.xxx Copyright 1980,83 by PICKLES & TROUT
2: SERIAL PORT A
3: baud rate: 110 134.5 150 300 600 1200 2400 4800 9600
4: # stop bits: 1 1.5 2
5: DTR:      high low      RTS: high low
6: word length: 5 6 7 8 bits
7: parity:   on off even odd
8: xmit on char: 17 xmit off char: 19
9:
10: SERIAL PORT B
11: baud rate: 110 134.5 150 300 600 1200 2400 4800 9600
12: # stop bits: 1 1.5 2
13: DTR:      high low      RTS: high low
14: word length: 5 6 7 8 bits
15: parity:   on off even odd
16: xmit on char: 17 xmit off char: 19
17:
18: CON:      S10A    CRT:      CENTRON  S10B
19: LST:      S10A    CRT:      CENTRON  S10B
20: PUN:      S10A    UPUN1   UPUN2   S10B
21: RDR:      S10A    URDR1   URDR2   S10B
22:
23:
24:     Arrows move cursor | <hold> - set field | <esc> - accept | <break> - quit
```

Figure 8.131 SETUP Display for Default Parameters

The third line of parameters for each port (Lines 8.131-5 and 8.131-13) shows the signal levels which will be placed on the Data Terminal Ready (DTR) and Request To Send (RTS) status lines on the RS-232 connector. The available choices for each of these parameters, "high" and "low", refer to the voltage present at the status line. "High" indicates a positive voltage and "low" indicates a negative voltage. Often, the status lines will not be used, so these parameters may be ignored. They must be set appropriately, however, when they provide needed signals to a peripheral device. Using the lines in this way will probably require special cabling and a thorough understanding of the device. Programs may also manipulate the status lines directly to send various instructions to a peripheral device.

The fourth line (Lines 8.131-6 and 8.131-14) of the serial port parameters shows the various word lengths available. Once again, this setting should agree with the peripheral device connected to the port.

The fifth line (Lines 8.131-7 and 8.131-15) shows the various options for parity. A parity bit is used by some devices for the detection of errors. When the parity option is "on", a parity bit is added to the ASCII code for each character sent. This bit is set so that either an even or odd number of bits in the character are "1". If the "on" and "even" options are selected for parity, the parity bit will be "1" if an odd number of bits in the character are "1" (resulting in an even number for the character and parity bit). Likewise, selecting the "on" and "odd" options will cause the parity bit to be "1" if an even number of bits in the character are "1". If the "off" option is selected, no parity bit will be sent and the specification of "even" or "odd" will have no effect. In general, parity should be avoided for the most common uses of the serial ports, such as running a printer, since it slows the transmission rate by about 10%.

The sixth line of parameters for each port (Lines 8.131-8 and 8.131-16) indicates the type of handshaking to be performed by the serial port I/O driver. The selection is made by the value assigned to the "xmit on" character, as shown in Figure 8.132.

xmit on	xmit off	type of handshaking
0	n/a	none
1	n/a	ETX/ACK
2	n/a	RS-232 status line
3-255	0-255	XON/XOFF type

Figure 8.132 Types of Serial Port Handshaking

When **no handshaking** (xmit on = 0) is specified, all characters sent to the serial port are transmitted at once. The serial port is essentially invisible to the program in this case. This option should be selected when a program uses a serial port but perform its own handshaking with the peripheral device.

When the **ETX/ACK** handshaking (xmit on = 1) is selected, the serial port driver will transmit an ASCII ETX character to the peripheral device after every 128 or so characters. After the EXT character is sent, all output to the peripheral device to be suspended until an ASCII ACK character is received from the device. This type of handshaking is performed by some printers (older Diablo's, for example). In general, ETX/ACK is less efficient than either RS-232 status line or XON/XOFF handshaking. Therefore, if you have a choice, avoid using ETX/ACK handshaking.

The **RS-232 Status Line** handshaking (xmit on = 2) causes the serial port to monitor the Clear To Send (CTS) and the Device Carrier Detect (DCD) input status lines of the serial port. Data will not be transmitted to the peripheral device unless the CTS line is high (positive voltage). If the CTS line goes low (negative voltage) during the transmission of a character, further transmission will be halted at the end of that character until the CTS line is high again. The DCD line must be high in order for the serial port to accept incoming characters. If the DCD line is low, no incoming characters on the serial port will be recognized. RS-232 status line handshaking is probably the most efficient of the handshaking methods provided and should be used if the peripheral device supports it. Among others, the TRSDOS system uses this type of handshaking to access the serial ports.

The **XON/XOFF** type of handshaking causes the serial port software to monitor the peripheral device constantly for incoming characters. Any character which is received is read by the software and checked to determine if it is the "xmit on" character or the "xmit off" character. All other characters are discarded. If the "xmit off" character is received, the serial port ceases to transmit data until an "xmit on" character is received from the peripheral.

If the port is set for 8-bit words, all 8 bits of the incoming characters are used when comparing to the "xmit on" and "xmit off" characters. Some peripheral devices always set the eighth bit to a "1", making it necessary (in some cases) to add 128 to the actual value of the "xmit on" and "xmit off" characters in order for the handshaking to work properly. For example, it is very common to have a value of 17 (ASCII XON) for the "xmit on" character and 19 (ASCII XOFF) for the "xmit off" character. For some peripheral devices, then, it may be necessary to set the "xmit on" character to 145 and the "xmit off" character to 147.

To change the value of the "xmit on" or "xmit off" character for a port, move the cursor to the current value of the character on the display and press the <hold> key. The last line of the display will then be replaced with the line shown in Figure 8.133. You may now enter the new value and press the <enter> key.

Enter the numeric value of the character: 2<enter>

Figure 8.133 Prompt Line for Entering "xmit on" and "xmit off" Values

Lines 8.131-18 to 8.131-21 show the current logical to physical I/O device assignments. The four CP/M logical I/O devices are displayed in the left column, and the physical I/O devices that may be assigned to each are shown on the same line. Figure 8.134 explains the symbols which denote the physical I/O devices. The I/O device assignments are changed in the same manner as the serial I/O parameters (ie. use the arrow keys to move the cursor to the desired assignment and press <hold>).

symbol	device
S10A	serial port A
S10B	serial port B
CRT	system video display and keyboard
CENTRON	parallel printer port
UPUN1	user supplied punch routine 1
UPUN2	user supplied punch routine 2
URDR1	user supplied reader routine 1
URDR2	user supplied reader routine 2

Figure 8.134 Physical I/O Devices

As an example, assume that serial port A is to be connected to the standard serial port printer and the printer has the I/O parameters shown in Figure 8.135. (These are actual parameters for a TI-820 dot matrix printer). 8

9600 baud  
1 stop bit  
8 bit data  
no parity  
RS-232 status line handshaking

Figure 8.135 Sample Serial Printer Parameters (TI-820)

After the needed settings have been made, the SETUP display will appear as shown in Figure 8.136.

```
1:          IO SETUP PROGRAM v3.xxx Copyright 1980,83 by PICKLES & TROUT
2: SERIAL PORT A
3: baud rate:   110   134.5 150   300   600   1200  2400  4800  9600
4: # stop bits: 1      1.5    2
5: DTR:        high   low    RTS:  high   low
6: word length: 5      6     7     8   bits
7: parity:     on     off   even odd
8: xmit on char: 2           xmit off char: 19
9:
10: SERIAL PORT B
11: baud rate:   110   134.5 150   300   600   1200  2400  4800  9600
12: # stop bits: 1      1.5    2
13: DTR:        high   low    RTS:  high   low
14: word length: 5      6     7     8   bits
15: parity:     on     off   even odd
16: xmit on char: 17          xmit off char: 19
17:
18: CON:        S10A    CRT    CENTRON  S10B
19: LST:        S10A    CRT    CENTRON  S10B
20: PUN:        S10A    UPUN1   UPUN2    S10B
21: RDR:        S10A    URDR1   URDR2    S10B
22:
23:
24:     Arrows move cursor | <hold> - set field | <esc> - accept | <break> - quit
```

Figure 8.136 SETUP Display After Setting Parameters for Serial Printer

### Using SETUP - Command Line Mode

The command line mode of SETUP allows you to set the serial port parameters by typing a single command line. You may set up either port A or port B or both of them by a single command line. The command line has the form shown in Figure 8.137. If both ports are being configured with a single command line, either port's parameters may appear first on the command line. Only the parameters appearing on the command line will be affected; all others will remain at their previous values.

```
SETUP [port letter] [port options] [port letter] [port options]
[port letter] is replaced by A or B.
[port options] is replaced by one or more instructions for
setting parameters for the port.
```

Figure 8.137 Form of the Command Line for SETUP

When specifying parameters on the command line, they must be separated from each other by one or more delimiting characters. The delimiting characters that are recognized are shown in Figure 8.138. Note that due to the variety of delimiting characters you can type exactly the same commands in a large variety of ways. For example "BAUD RATE=9600" has identical effect as "BAUD RATE 9600".

```
space , : . =
```

Figure 8.138 Delimiter Characters that can be Used on SETUP Command Line

There are several ways of specifying each of the parameters. In the following descriptions, when there are several words or characters to choose among in a particular place, the choices will be shown enclosed in angle brackets, e.g. <5, 6, 7, 8>.

```
WORD LENGTH <5, 6, 7, 8>
WORD <5, 6, 7, 8>
```

Figure 8.139 Instructions for Setting Word Length

Some examples are:

**WORD LENGTH 7** - set the word length to 7 bits.

**WORD 8** - sets the word length to 8 bits.

```
STOP BIT <1, 1.5, 2>
STOP <1, 1.5, 2>
SB <1, 1.5, 2>
```

Figure 8.140 Instructions for Setting Number of Stop Bits

Some examples are:

**STOP BIT 1** - sets one stop bit.

**SB 2** - sets two stop bits.

DTR <HIGH, HI, H, LOW, LO, L>  
RTS <HIGH, HI, H, LOW, LO, L>

Figure 8.141 Instructions for Setting Port Status Line Outputs

Some examples are:

**DTR HIGH** - sets a high voltage on the DTR output.

**RTS LO** - sets a low voltage on the RTS output.

**DTR H** - sets a high voltage on the DTR output.

PARITY <ODD, OD, O, EVEN, EV, E, OFF>  
PAR <ODD, OD, O, EVEN, EV, E, OFF>  
P <ODD, OD, O, EVEN, EV, E, OFF>

Figure 8.142 Instructions for Setting Parity

Some examples are:

**PARITY ODD** - sets odd parity.

**PAR EV** - sets even parity.

**P OFF** - sets no parity (parity off).

XMIT ON <num>  
XON <num>  
XMIT OFF <num>  
XOFF <num>  
XOF <num>

Figure 8.143 Instructions for Setting XMIT ON and OFF Characters

Some examples are:

**XMIT ON 0** - sets xmit on character to 0.

**XON 17** - sets xmit on character to 17.

**XOFF 19** - sets xmit off character to 19.

BAUD RATE <baud rate number>																		
BAUD <baud rate number>																		
BR <baud rate number>																		
where the baud rate number is chosen from the following:																		
<table><thead><tr><th>baud rate</th><th>numbers</th></tr></thead><tbody><tr><td>110</td><td>110, 11</td></tr><tr><td>134.5</td><td>134.5, 13</td></tr><tr><td>300</td><td>300, 30</td></tr><tr><td>600</td><td>600, 60</td></tr><tr><td>1200</td><td>1200, 12</td></tr><tr><td>2400</td><td>2400, 24</td></tr><tr><td>4800</td><td>4800, 48</td></tr><tr><td>9600</td><td>9600, 96</td></tr></tbody></table>	baud rate	numbers	110	110, 11	134.5	134.5, 13	300	300, 30	600	600, 60	1200	1200, 12	2400	2400, 24	4800	4800, 48	9600	9600, 96
baud rate	numbers																	
110	110, 11																	
134.5	134.5, 13																	
300	300, 30																	
600	600, 60																	
1200	1200, 12																	
2400	2400, 24																	
4800	4800, 48																	
9600	9600, 96																	

Figure 8.144 Instructions for Setting Baud Rate

Some examples are:

**BAUD RATE 9600** - sets baud rate to 9600.

**BAUD 1200** - sets baud rate to 1200.

**BR 48** - sets baud rate to 4800.

Some examples of command lines that might be used with SETUP are:

**SETUP A BAUD=9600,SB=1,WORD=8,PARITY OFF,XON=2**

Sets up serial port A for 9600 baud, 1 stop bit, 8 bit data, no parity, and hardware handshaking. The status line outputs are left in their previous state.

**SETUP B BAUD=1200 SB=1 WORD=8 PARITY ODD XON=0**

Sets up serial port B for 1200 baud, 1 stop bit, 8 bit data, odd parity, and no handshaking. The status line outputs are left in their previous state.

**SETUP A BR 1200 ; B BR 9600 XON 17 XOFF 19**

Sets up serial port A for 1200 baud; all other parameters of serial port A are left unchanged. Also sets up serial port B for 9600 baud and XON/XOFF handshaking with the standard ASCII XON and XOFF characters. All other parameters of serial port B are left unchanged.

### Possible Error Messages

**Bad value, please re-enter:**

Displayed in the interactive mode if the value you have entered for either the "xmit on" or "xmit off" character was not in the acceptable range of 0 to 255.

**No serial port driver in the system.**

This message is displayed if you execute SETUP on a system that does not have a serial port driver. If you want to use the serial ports you should use the MODSEL utility program (see Section 6.3) to select a serial port module to be included with the system.

**Command line error with SUBMIT in progress.**

**Do you want to continue with the SUBMIT file? (Y/N) :**

(Command line mode) When SETUP detects an error in its command line it checks for a SUBMIT file in progress. If it finds one, it asks you whether you

want to continue with it. If you respond affirmatively, the SUBMIT file will continue. If you respond negatively, the SUBMIT file will be aborted. If you are using the command line mode of SETUP from a SUBMIT file, any error on the command line may cause the rest of the SUBMIT file to misbehave. This message allows you the option of terminating the SUBMIT file if you wish.

**Port not selected.**

(Command line mode) This message indicates that SETUP could not find a port indicator ("A" or "B") on the command line.

**Invalid Command.**

(Command line mode) This message is displayed if SETUP cannot recognize an instruction given on the command line.

**Bad Stop-Bit value.**

(Command line mode) Displayed if the number of stop bits specified is not valid.

**Bad DTR value.**

(Command line mode) Displayed if the level to set the DTR output status line is not properly specified.

**Bad RTS value.**

(Command line mode) Displayed if the level to set the RTS output status line is not properly specified.

**Bad Word-Length value.**

(Command line mode) Displayed if the number of bits specified for a data word is invalid.

**Bad Parity value.**

(Command line mode) Displayed if the parity is improperly specified.

**Bad Xmit-On value.**

(Command line mode) Displayed if the Xmit-On character is not specified with a number in the range 0 - 255.

**Bad Xmit-Off value.**

(Command line mode) Displayed if the Xmit-Off character is not specified with a number in the range 0 - 255.

**Bad Baud-Rate value.**

(Command line mode) Displayed if the value specified for the baud rate is invalid.

**8.28 Utility name: STAT**

**Purpose:** To display various system statistics on the console.

### General Description

STAT provides a means of displaying various system statistics and parameters at the system console. In addition, STAT can be used to modify the system's logical-to-physical I/O device assignments and the disk file attributes. STAT may be executed from a SUBMIT file but it does not contain any of the output redirection capabilities available in some P&T CP/M 2 utility programs.

### Using STAT

The parameters specified on the command line that executes STAT entirely control the operation of the program. In other words, STAT does not prompt for commands; it performs the functions specified on the command line and then returns to the operating system. It is also important to note that all of STAT's commands (except USR:) pertain only to the current user area. If no parameters are specified when STAT is executed, STAT will report the amount of available space on each of the active drives, as shown in Figure 8.145. In this example, only drive A was active. (Remember: Only those drives which have been accessed since the last warm boot are active.)

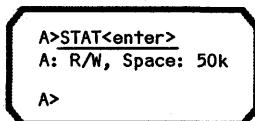


Figure 8.145 Example of Using STAT with no Parameters

In order to show the available space on a particular drive (whether it is active or not), the logical drive letter may be specified in the command line, as shown in Figure 8.146. In this case, STAT will show the available space on the specified drive only.

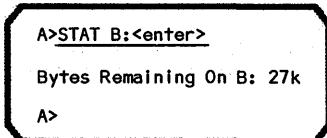


Figure 8.146 Using STAT to Display Available Space on a Particular Drive

It is also possible to have STAT show statistics about a specific disk file. Figure 8.147 shows the use of STAT to display the statistics of PIP.COM.

```
A>STAT PIP.COM<enter>
      Recs   Bytes   Ext Acc
      58     8k      1 R/W A:PIP.COM
Bytes Remaining On A: 50k
A>
```

Figure 8.147 Displaying Statistics for a Particular File Using STAT

When displaying file statistics, STAT uses the form shown in Figure 8.148. The first column of the display, labeled "Recs", shows the number of logical sectors (128 bytes) that have been written to the file. The most accurate estimate of the actual amount of information stored in the file is obtained by multiplying the number of logical records by 128. This number is still an estimate, however, since there is no way to determine how many bytes of actual data are present in the last sector.

```
1:  Recs   Bytes   Ext Acc
2:  58     8k      1 R/W A:PIP.COM
3:
4:  58     8k      1 R/O A:PIP.COM
5:
6:  58     8k      1 R/W (A:PIP.COM)
7:
8:  58     8k      1 R/O (A:PIP.COM)
```

Figure 8.148 Forms of File Statistics Display

The second column, labeled "Bytes", shows the number of bytes of disk space allocated to the disk file. This number is always given in Kbyte (1024 byte) units. Since disk space is allocated in blocks of 1, 2, 4, 8, or 16 Kbytes, the space allocated will frequently be larger than the space required by the number of logical sectors (also called records, 128 bytes each) in the file. As a result, the amount of disk space allocated to a disk file gives only a rough estimate of the actual size of the file. For diskettes, the allocation block size is 1 Kbyte for single density and 2 Kbytes for double density.

The third column, labeled 'Ext', shows the number of directory entries used by the file. As a file grows it uses more disk space than can be represented in one directory entry. In order to represent this additional space, another directory entry is used as an extension of the first entry. These "extensions" are called extents. The process is repeated as the file grows. Thus, a number of extents may be required for a large file. For example, the storage capacity on a single sided double density diskette is 596 Kbytes, and 16 Kbytes can be stored under each directory entry. If a file were to use the entire diskette, it would then require 38 directory entries.

The fourth column, labeled "Acc", shows the access attribute for the file. If the attribute is shown as "R/W", the file is available for both reading and writing. If it is shown as "R/O", the file is available for reading only, and any attempt to write to it or erase it will cause an error. Lines 8.148-2 and 8.148-6 show files tagged as "read/write", while Lines 8.148-4 and 8.148-8 show "read only" files.

The fifth column of the STAT display, which is not labeled, shows the file name and the drive on which the file was found. If the file has the \$DIR attribute set (so that it will be displayed in directory listings), the name is listed normally. If the file has the \$SYS attribute set (so that it will not be displayed in directory

listings), its name will be shown in parentheses. Lines 8.148-2 and 8.148-4 show \$DIR files, and Lines 8.148-6 and 8.148-8 show \$SYS files.

If a wild card file name is used on the command line, STAT can show the statistics on a group of files. Figure 8.149 shows the use of STAT to list all file names from the current default drive with the extension "ASM". The wild card file name "**\*.\*!**" will cause statistics of all files on the current default drive to be listed.

```
A>STAT *.ASM<enter>
    Recs   Bytes  Ext Acc
      33     6k   1 R/W A:(DUMP.ASM)
      27     4k   1 R/W A:(SETTIME.ASM)
      16     2k   1 R/W A:(TIME.ASM)
Bytes Remaining On A: 50k
A>
```

Figure 8.149 Displaying Statistics for all Files with ASM Extension

In order to have STAT change disk file attributes, merely specify the new value of the attribute following the file name on the command line. In Figure 8.150, STAT is used to set PIP.COM to "read only" status. Note that the "\$" on the command line is part of the instruction to set a file to read only and must be included.

```
A>STAT PIP.COM $R/O<enter>
    PIP.COM set to R/O
A>
```

Figure 8.150 Using STAT to set a File to R/O

The attributes of a group of files may be changed with a single command by specifying a wild card file. In response to the command in Figure 8.151, STAT will set the \$SYS attribute of all files having an extension of "ASM".

```
A>STAT *.ASM $SYS<enter>
    DUMP.ASM set to SYS
    SETTIME.ASM set to SYS
    TIME.ASM set to SYS
A>
```

Figure 8.151 Using STAT to set all Files with Extension ASM to SYS

Figure 8.152 shows the various different indicators recognized by STAT for changing file attributes.

\$SYS	sets a file to system status. The name of a SYS file will not be listed in a directory display.
\$DIR	sets a file to non-system status. The names of non-system files are listed in a directory display.
\$R/O	sets a file to read only status. Any attempt to write to such a file will result in an error.
\$R/W	sets a file to read/write status. Both reading from and writing to such a file are allowed.

Figure 8.152 Indicators Used by STAT for Setting File Attributes

In addition to setting file attributes, STAT can set a disk drive to read-only status. Any attempt to write to that drive will then result in an error. However, setting a drive to read-only status has limited usefulness. Since the status is maintained only until a program resets the disk system or a warm boot is performed, it can be dangerous to try to protect data in this way. After all, a warm boot usually occurs at the end of every program, and a program may reset the disk system at times which the user may not be able to predict. It is much safer to protect the data by physically write protecting the diskette by uncovering its write protect notch.

Figure 8.153 illustrates the use of STAT to set drive A to read only status.

```
A>STAT A:=R/O<enter>
A>
```

Figure 8.153 Using STAT to Set Drive A to Read/Only Status

As shown in Figure 8.154, the DSK: command causes STAT to display the configuration of the drives that are currently logged on.

```
1: A>STAT DSK:<enter>
2:
3:      A: Drive Characteristics
4:      4800: 128 Byte Record Capacity
5:      600: Kilobyte Drive Capacity
6:      128: 32 Byte Directory Entries
7:      128: Checked Directory Entries
8:      128: Records/ Extent
9:      16: Records/ Block
10:     64: Sectors/ Track
11:     2: Reserved Tracks
12:
13:      B: Drive Characteristics
14:      1944: 128 Byte Record Capacity
15:      243: Kilobyte Drive Capacity
16:      64: 32 Byte Directory Entries
17:      64: Checked Directory Entries
18:      128: Records/ Extent
19:      8: Records/ Block
20:     26: Sectors/ Track
21:     2: Reserved Tracks
22:
23: A>
```

Figure 8.154 Displaying Drive Parameters with STAT

In this example, only drives A and B are currently logged on. Drive A has a double density diskette mounted on it, while drive B has a single density diskette. The first line of the parameters given for each drive (Lines 8.154-3 and 8.154-14) shows the total number of logical sectors in the data area of the disk drive. (Note: the data area includes the disk directory, so some of these sectors are not available for data storage.) The second line (Lines 8.154-5 and 8.154-15) shows the size of the data area of each diskette in Kbytes. Once again, this number includes the directory area of the disk.

The third line (Lines 8.154-6 and 8.154-16) of the parameters shows the number of directory entries allocated to the disk. Since each directory entry requires 32 bytes of storage, the available storage of the disk can be calculated by the formula shown in Figure 8.155.

available storage = (drive cap in Kbytes) - [(# dir entries) / 32]

Figure 8.155 Formula for Calculating Available Storage

The fourth line (Lines 8.154-7 and 8.154-17) of the parameters shows the number of directory entries that are checked to test for disk changes. When P&T CP/M 2 accesses a disk, it may perform a check to determine whether the disk has been changed since the last access. It checks by comparing the contents of the directory with information previously derived from the directory. If the two do not match, the system assumes that the disk has been changed without a disk system reset. If the system determines that a disk has been changed, it will mark it as "read only". Any subsequent attempt to write to it will then cause a BDOS "disk R/O" error.

The numbers shown on Lines 8.154-7 and 8.154-17 will be non-zero if checking for changed disks is being performed and zero if it is not. When checking is performed, the parameter will be set equal to the number of directory entries. Diskettes are always checked since they can be changed so easily. For hard disk drives you get to specify (with the HDCONFIG program supplied with hard disk systems) whether the checking should take place. In the case of a fixed hard disk, it is conventional not to perform checking since the disk cannot be changed.

The fifth line (Lines 8.154-8 and 8.154-18) of the parameters shows the number of logical sectors that can be represented by a single directory entry. If a disk file contains more than that number of logical sectors, it will require more than one directory entry.

The sixth line (Lines 8.154-9 and 8.154-19) of the parameters shows the number of logical sectors in each allocation block for the disk. An allocation block is the minimum amount of disk storage that can be assigned to a disk file. In other words, a file with only a single byte of data in it will still use an entire allocation block of disk storage.

The seventh line (Lines 8.154-10 and 8.154-20) of the parameters shows the total number of logical sectors per track on the disk drive. This number has very little importance to the average user of the system.

The eighth (and last) (Lines 8.154-11 and 8.154-21) line of the parameters shows the number of tracks that are reserved for system usage. Diskettes will always have two tracks reserved regardless of whether a system resides on it.

In response to the command given in Figure 8.156, STAT will display the current user and all user numbers under which files are stored on the disk.

A>STAT USR:<enter>  
Active User : 0  
Active Files: 0 3 5  
A>

Figure 8.156 Displaying User Information with STAT

It is generally more convenient to use ASSIGN or SETUP rather than STAT to manipulate the assignment of physical I/O devices to logical I/O devices. In some cases, however, the use of STAT may be necessary. Unfortunately, STAT recognizes only the standard physical I/O device names of generalized CP/M, which have little relationship with the actual devices available on the computer. Figure 8.157 shows

the correspondence between the physical I/O device names used by STAT and SETUP or ASSIGN.

STAT	SETUP
TTY:	S10A
CRT:	CRT
BAT:	CENTRON
UC1:	S10B
PTR:	URDR1
UR1:	URDR2
UR2:	S10B
PTP:	UPUN1
UP1:	UPUN2
UP2:	S10B
LPT:	CENTRON
UL1:	S10B

Figure 8.157 Correspondence of Physical Device Names Between STAT and SETUP

The "DEV:" command causes STAT to display the current I/O device assignment as shown in Figure 8.158.

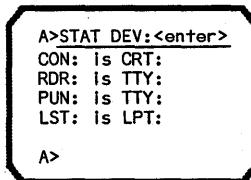


Figure 8.158 Displaying I/O Device Assignments with STAT

Figure 8.159 shows the form of the command which changes the assignments. This particular example assigns serial port B, UR2: to the RDR: logical device.

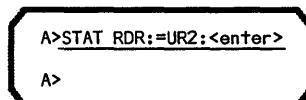


Figure 8.159 Setting Device Assignments with STAT

Finally, Figure 8.160 illustrates how the "VAL:" command causes STAT to display a listing of the various commands it accepts as well as the physical I/O devices that can be assigned to each logical I/O device.

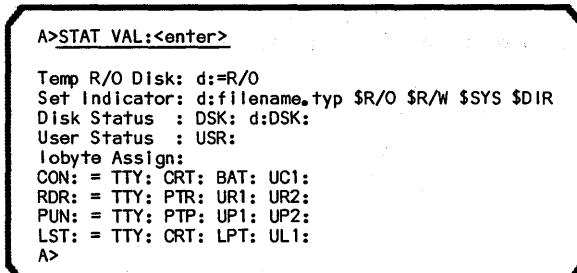


Figure 8.160 Displaying a Summary of STAT Commands and I/O Devices

### Possible Error Messages

#### **\*\* Aborted \*\***

A character has been typed at the console keyboard while STAT was changing file attributes, which causes STAT to abort the command and return immediately to the operating system.

#### **Bad Delimiter**

You have used a bad delimiter between the logical I/O device name and the physical I/O device name when specifying an I/O assignment.

#### **File Not Found**

The file you have specified in the STAT command could not be found on the specified drive.

#### **Invalid Assignment**

While making an I/O device assignment, you have specified an invalid physical I/O device for the logical device. Use the VAL: command to display the valid assignments for each logical device.

#### **Wrong CP/M Version (Requires 2.0)**

This version of STAT will run only with version 2.0 or greater of CP/M. You are attempting to use it with a previous version.

**8.29 Utility name: SUBMIT**

**Purpose:** To batch together a series of commands to be automatically executed by the system.

### General Description

SUBMIT provides a means of executing a sequence of commands (which would normally be entered one at a time) automatically without operator intervention. The user first creates a text file (with the extension "SUB") containing the prototype commands. When SUBMIT is invoked, the commands will then be fed into the Console Command Processor (CCP) in place of keyboard input. In addition, a general SUB file can be used for different purposes by the substitution of parameters.

The commands in a SUBMIT file can perform any operation that you can do by typing a command into the system. In particular, you might use a SUBMIT file and the command line modes of some P&T CP/M 2 utility programs to format, test, and copy data to a new diskette with a single user command.

### Using SUBMIT

In order to use the SUBMIT utility, you first must prepare a SUB file with a text editor. This file contains the command lines that will be fed into the Console Command Processor (CCP). Figure 8.161 shows a sample SUB file.

```
: This is a demonstration of the use of SUBMIT
:
asm $1.aaa
load $1
$1 $2
```

Figure 8.161 Sample SUB File

Note that all characters appearing after a colon (except when used for a drive designation) on a line in the SUB file are ignored. As in our example, this feature can be used to insert comment lines in the file. These lines will be displayed on the console as the SUB commands are executed but will cause no action to be taken by the CCP.

Parameters to be inserted in the prototype commands are specified in the SUB file by a dollar sign followed by a number (e.g. \$1). The number indicates the position on the SUBMIT command line of the parameter to be substituted. In order to specify an actual dollar sign in the SUB file, use two dollar signs ("\$\$"). Notice that two parameters (\$1 and \$2) are used in the SUBMIT file shown in Figure 8.161. This example assembles a file specified by the first parameter on the SUBMIT command line, loads it, and then executes it according to the second parameter on the SUBMIT command line.

Assume that the SUB file shown in Figure 8.161 is stored on drive A with the name TEST.SUB. Figure 8.162 shows the console display which results when this SUB file is used to assemble the DUMP utility and to display its object code.

```
A>SUBMIT TEST DUMP DUMP.COM<enter>
A>: THIS IS A DEMONSTRATION OF THE USE OF SUBMIT
A>:
A>ASM DUMP.AAA
CP/M ASSEMBLER - VER 2.0
0257
001H USE FACTOR
END OF ASSEMBLY

A>LOAD DUMP

FIRST ADDRESS 0100
LAST ADDRESS 0212
BYTES READ 0113
RECORDS WRITTEN 03

A>DUMP DUMP.COM

0000 21 00 00 39 22 15 02 31 57 02 CD C1 01 FE FF C2
0010 1B 01 11 F3 01 CD 9C 01 C3 51 01 3E 80 32 13 02
      :       :       :
0160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

A>
```

Figure 8.162 Example of Submitting the File Shown in Figure 8.161

In processing the SUB file, the SUBMIT program generates a temporary file named \$\$\$.SUB on logical drive A. \$\$\$.SUB contains the processed commands ready to be read by the CCP. As the CCP processes the commands, it modifies the \$\$\$.SUB file to remove the processed commands. Therefore, drive A cannot be write protected while a SUB file is being executed.

The execution of a submitted file may be aborted by typing any character at the console while the next command is being read from the temporary SUB file.

Note: SUBMIT will ignore all lines preceding a blank line in a SUB file, so no lines should be left blank. The most common problem in using the SUBMIT utility occurs because one or more blank lines are left at the end of the SUB file. This causes the entire SUB file to be ignored, giving the appearance that SUBMIT is not working.

Note that SUBMIT sends commands to the CCP only; it does not allow you to specify input which is to be provided to a program while it is running. This capability is available, but it requires the use of XSUB in conjunction with SUBMIT. See the discussion of XSUB in Section 8.33 for details.

A SUB file may invoke the SUBMIT program to begin the execution of another SUB file, but this causes the first SUB file to disappear from memory. The second SUB file cannot "return" to the point in the first where it was invoked. As a result, a SUB file may even invoke execution of itself. A useful application of this technique is the generation of "circular" SUB files, which cause a series of commands to be endlessly repeated until the user intervenes. For example, it is often very convenient to have the system switch automatically between a text editor and the assembler when you are writing an assembly language program. Figure 8.163 shows a SUB file, named ASMBL.SUB, which performs this task.

```
ed $1.asm
asm $1.aaa
submit asmb1 $1
```

Figure 8.163 Example of a Circular SUB File

When ASMBL.SUB is submitted, it first invokes the text editor, allowing the user to write and/or modify the new program. An exit from the editor causes execution of the SUB file to resume, which immediately invokes the assembler. As soon as the assembler returns to the system, the SUB file invokes its own re-execution (by SUBMIT), thus bringing back the text editor. This "tail-chasing" continues until you abort the SUB file by typing a character at the keyboard between execution of its commands.

### Possible Error Messages

#### Error On Line nnn,

This message is used as a lead-in to most of the other error messages. It indicates which line ("nnn") of the SUB file the SUBMIT program was processing when the error occurred.

#### No 'SUB' File Present

SUBMIT could not find a file with the specified primary name and an extension of "SUB" on the specified drive.

#### Disk Write Error

A error has occurred while SUBMIT was writing its output file to the disk. This message most often indicates that the disk is full.

#### Command Buffer Overflow

The internal buffer that SUBMIT uses in processing the SUB file has been filled to capacity. This indicates that the SUB file is too long (has too many commands) to be processed by SUBMIT. The solution is to break the SUB file into two or more parts. Each part is then stored as a separate file, and all but the last should end with a command to execute SUBMIT on the succeeding file.

#### Command Too Long

The maximum length of a command line after all parameter substitution is 127 characters. The indicated line has exceeded this length.

#### Parameter Error

A dollar sign which is not followed by a number or another dollar sign appears in the SUB file and cannot be interpreted. If you want a dollar sign in the SUB file, you should represent it by two dollar signs (ie. " \$\$").

#### Invalid Control Character

Control characters cannot be imbedded in the temporary SUB file regardless of what the standard CP/M documentation states.

#### Directory Full

SUBMIT has found that the directory on the current default drive was already full when it attempted to create its output file.

**8.30 Utility name: SYNCRO**

**Purpose:** To synchronize the system date and time to a P&T CCB board.

**General Description**

SYNCRO reads the date and time from a P&T CCB clock/calendar board (if one is installed in the computer) and sets the system date and time to agree with it. It may be used to restore the correct date and time when they have been altered by a program or by the SETTIME and/or SETDATE utility programs. Also, SYNCRO may be used periodically to correct any inaccuracy in timekeeping during extended system operation. Normally, the system clock loses about 5 seconds per hour of operation.

**Using SYNCRO**

SYNCRO produces no output on the system console in execution, so it may be used within SUBMIT files or with other program chaining techniques. The example in Figure 8.164 shows SYNCRO being run from the console.

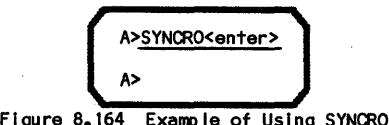


Figure 8.164 Example of Using SYNCRO

**Possible Error Messages**

SYNCRO produces no error messages.

**8.31 Utility name: TRS2CPM**

**Purpose: To transfer disk files from a TRSDOS diskette to a CP/M disk.**

### General Description

The TRS2CPM utility routine allows the user to convert files on a TRSDOS diskette to files on a CP/M diskette. It was designed primarily to transfer programs in source code created by, for instance, TRSDOS Basic. Although it can also transfer most object code files, they will be of limited usefulness under CP/M, since they are designed to be executed in a TRSDOS environment.

### Data Files

TRSDOS data files contain text, program source code, and information stored on the diskette by programs. TRS2CPM transfers a data file from a TRSDOS diskette to a CP/M diskette on a byte-for-byte basis. It does not translate or interpret the data in any way. For text files, however, TRS2CPM does provide an option for inserting a line feed character after each carriage return in the TRSDOS data. Most CP/M programs expect both a carriage return and a line feed at the end of a line, while lines of a text file under TRSDOS end with a carriage return only. Also, the CP/M file will be terminated with a <ctl-Z>, the standard CP/M "end of file" mark for a text file. These actions will make a TRSDOS text file usable by most CP/M programs.

Some programs (notably RSCOBOL) under TRSDOS store textual material in the TRSDOS variable record length format. These files do not have a carriage return at the end of each line, but rather have a character count at the beginning of each line. Transferring such a file to CP/M and attempting to use it (say edit it with a text editor) will cause problems. TRS2CPM provides an option for modifying variable record length files as they are transferred so that they will appear as a normal text file to CP/M programs. This is accomplished by removing the record length count from the beginning of each record and inserting a carriage return and line feed at the end of each record. Of course, you may still transfer the files unmodified (byte-for-byte) if you are writing programs under CP/M that can deal with the TRSDOS variable record length directly.

Once a TRSDOS data file has been moved to CP/M, the different file structures used by the two systems may require that changes be made before the file can be used. In general, the file structure of CP/M is less involved. A CP/M program always writes 128 bytes (a logical sector) of data, and it must keep track of the beginnings and endings of records within the file. A TRSDOS program may write out any number of bytes to a data file, and the file itself maintains information indicating where each record begins and ends.

One result of this difference can be found in the last logical sector of the CP/M file generated from TRSDOS by TRS2CPM, which may contain up to 127 bytes of garbage. (CP/M can detect an "end of file" condition only when the last logical sector is read.) However, the <ctl-Z> that TRS2CPM inserts at the end of the file will always be placed immediately following the last valid byte of data as defined by the TRSDOS "end of file" pointer. In most cases, this will allow an editor or

other program to detect the "end of file" properly, assuming that the transferred file does contain text.

Non-text data files generated by programs under TRSDOS may also be transferred by TRS2CPM, but they may not be directly useful under CP/M. A CP/M program could successfully access the data only by reading it in the same manner as the TRSDOS program which originally wrote the file. Writing such a CP/M program obviously requires a thorough understanding of the original storage mechanisms and considerable programming ability. Moreover, files created by different TRSDOS programs may require different CP/M programs to read them correctly. In some cases, it may be necessary to write a program that converts the data file from its original format to one which can be read by the CP/M program.

When TRS2CPM is used to transfer a random access data file, it may not transfer the entire file. The transfer is performed sequentially from the beginning of the file and will terminate when the first empty record is encountered.

### Load Files

After the source code of a program (in TRSDOS Basic, for instance) is transferred to a CP/M disk it can often be modified somewhat and used under CP/M. This, of course, requires that a similar language be available for use with CP/M. A TRSDOS program in executable machine code may be transferred to a CP/M disk but it will most likely not be usable. This is especially true of Radio Shack programs such as Scripsit and Profile II.

A TRSDOS load file contains program code to be loaded into memory and possibly executed. Each load file contains a load address indicating where it is to be placed in memory. It may also contain an execution address indicating where execution of the program contained in the file is to begin. As a load file is transferred from the TRSDOS diskette to the CP/M diskette, it is converted to an Intel Hex file. (This is the only common type of file under CP/M that can contain both types of addresses.) The load addresses in the hex file are set to the load address specified within the TRSDOS file. Loading of a hex file under CP/M is most commonly performed by the DDT or LOAD utilities.

Since the hex format file represents each byte of the file by two ASCII characters and adds load addresses, checksums, and other overhead, it will be about 2.5 times larger than the file from which it was generated. For long files, the conversion time can be appreciable. The hex file is terminated by a <ctl-Z>, as is true for most ASCII files under CP/M.

Some load files cannot be transferred by TRS2CPM because they specify multiple load addresses that are meaningless under CP/M. Since these files are uniquely associated with the TRSDOS operating system, their transfer should be of no interest. All program load files (generated by a compiler or assembler) should be transferable by TRS2CPM.

### Using TRS2CPM

It is very difficult to give detailed and exhaustive written instructions for the TRS2CPM utility. Only a few of the program's many options are involved in a given transfer, and it would be practically impossible to cover every combination. However, every effort has been made to make the program self-prompting and easy to use. By reading the information in this section carefully, you should be able to

learn enough about TRS2CPM to make use of it, even if your situation varies from the ones in the examples.

To execute the program, type in the line shown in Figure 8.165 while in the command mode of CP/M.

1: A>TRS2CPM<enter>

Figure 8.165 Command Line to Execute TRS2CPM

If your computer system has more than one floppy disk drive, the initial dialog with TRS2CPM be similar to that shown in Figure 8.166.

```
1:          TRS to CP/M Transfer Utility
2:          Version 2.xxx
3:          Copyright 1980,81,83 by Pickles & Trout
4:
5:          TRS is a trademark of Tandy Corp.
6:          CP/M is a trademark of Digital Research
7:
8: 8: Enter source drive (B through C) :C<enter>
9:
10: 10: Enter destination drive (A through C) :B<enter>
11:
12: 12: Mount TRSDOS source disk on drive C
13: 13: and the CP/M destination disk on drive B
14:
15: 15: Press ENTER when disks are mounted <enter>
```

Figure 8.166 Opening Dialog for TRS2CPM on a Multiple Drive System

On Line 8.166-8, TRS2CPM asks for the source drive. This is the floppy disk drive on which the TRSDOS diskette will be mounted. In this example, drive C is specified. Because of restrictions of CP/M, drive A (the built-in drive) cannot be the source drive in a multiple-drive system. For a single drive system, TRS2CPM prompts you to mount diskettes in an order that avoids the problem.

On Line 8.166-10, TRS2CPM shows you the CP/M logical drives that may be used as the destination drive. The transfer of TRSDOS files will be made to the CP/M diskette mounted on the drive which you specify. In this case, the built-in floppy disk drive is available, although the example shows the selection of drive B. It is also possible to specify the same floppy disk drive for both the source and destination (not drive A, of course), as is illustrated below in Figures 8.168 and 8.169.

Once your selections are entered, TRS2CPM will prompt you to mount the TRSDOS diskette and the CP/M diskette on the drives you have specified and then press <enter> (Lines 8.166-12 to 8.166-15). If the destination drive is the built-in floppy disk drive, it is permissible to remove the system diskette and mount another CP/M diskette at this time. TRS2CPM will prompt you to mount a system diskette before attempting to return to the operating system, so there is no risk of an attempted warm boot from a non-system diskette. The destination diskette may be either single or double density, regardless of the drive on which it is mounted.

After you have pressed <enter> (Line 8.166-15), TRS2CPM will flash the message "Drive not Ready" at the bottom of the screen (as shown in Line 8.167-19) if this condition is detected at the source drive. The message will continue to flash until you press the <enter> or <break> key. Either action will cause TRS2CPM to return to the initial drive specification dialog. If the destination drive is not ready, the standard BIOS error message ("Drive X not ready - please check it") will be flashed

in the center of the display. When it is made ready, the program will continue automatically.

```
1:          TRS to CP/M Transfer Utility
2:          Version 2.xxx
3:          Copyright 1980,81,83 by Pickles & Trout
4:
5:          TRS is a trademark of Tandy Corp.
6:          CP/M is a trademark of Digital Research
7:
8:  Enter source drive (B through C) :C<enter>
9:
10: Enter destination drive (A through C) :B<enter>
11:
12: Mount TRSDOS source disk on drive C
13: and the CP/M destination disk on drive B
14:
15: Press ENTER when disks are mounted <enter>
16:
17:
18:
19: Drive not Ready
```

Figure 8.167 Example of Message Given When Source Drive is not Ready

If you specify the same physical floppy disk drive for both the source and destination diskettes, TRS2CPM will prompt you to exchange diskettes at the appropriate times as it switches between reading and writing. For example, it will flash the message "Mount CP/M disk then press ENTER" at the bottom left of the screen when access to that disk is needed. (See Figure 8.168.) Similarly, the message "Mount TRSDOS disk then press ENTER" will be flashed at the bottom right of the screen when TRS2CPM is ready to access that disk. (See Figure 8.169.) Obviously, the time and human interaction involved in this procedure weigh heavily against its use on a multiple-drive system, and it is hard to conceive of a situation when it might be needed.

```
1:          TRS to CP/M Transfer Utility
2:          Version 2.xxx
3:          Copyright 1980,81,83 by Pickles & Trout
4:
5:          TRS is a trademark of Tandy Corp.
6:          CP/M is a trademark of Digital Research
7:
8:  Enter source drive (B through C) :C<enter>
9:
10: Enter destination drive (A through C) :C<enter>
11:
12:
13:
14:
15:
16:
17:
18:
19: Mount CP/M disk then press ENTER
```

Figure 8.168 TRS2CPM Prompt for Mounting the CP/M Disk

```
1: TRS to CP/M Transfer Utility
2: Version 2.xxx
3: Copyright 1980,81,83 by Pickles & Trout
4:
5: TRS is a trademark of Tandy Corp.
6: CP/M is a trademark of Digital Research
7:
8: Enter source drive (B through C) :C<enter>
9:
10: Enter destination drive (A through C) :C<enter>
11:
12:
13:
14:
15:
16:
17:
18:
19:
20: Mount TRSDOS disk then press ENTER
```

Figure 8.169 TRS2CPM Prompt for Mounting the TRSDOS Diskette

If your computer system has a single floppy drive, TRS2CPM will automatically initiate the disk-swapping procedure described above. Since there is no choice of the source or destination drives, TRS2CPM will omit the prompts for your selections and will proceed immediately to the first mount request, as illustrated in Figure 8.170.

```
1: TRS to CP/M Transfer Utility
2: Version 2.xxx
3: Copyright 1980,81,83 by Pickles & Trout
4:
5: TRS is a trademark of Tandy Corp.
6: CP/M is a trademark of Digital Research
7:
8:
9:
10:
11:
12:
13:
14:
15:
16:
17:
18:
19: Mount CP/M disk then press ENTER
```

Figure 8.170 TRS2CPM Opening Dialog for a Single Drive System

After the drive specifications (if any) are made and the diskettes are mounted, TRS2CPM will read the directory from the TRSDOS diskette and display it as shown in Figure 8.171.

1:	SYSTEM .SYS	SYSRES .SYS	BACKUP .	PATCH .	
2:	XFERSYS .	FORMAT .	SYSTEM64.	TERMINAL.	
3:	MEMTEST .	BASIC .	HERZ50 .	LP11 .	
4:	PRTBKSP .	SYSTEM32.	DOC0M32 .	BASCOM32.	
5:	COMSUB32.	EXDATM32.	DATM32 .	DOC0M64 .	
6:	BASCOM64.	COMSUB64.	EXDATM64.	DATM64 .	
7:	.	.	.	.	TRS to CP/M
8:	.	.	.	.	Utility
9:	.	.	.	.	Version 2.xxx
10:	.	.	.	.	(c) 1980,81,83
11:	.	.	.	.	by
12:	.	.	.	.	Pickles & Trout
13:	.	.	.	.	all rights
14:	.	.	.	.	reserved
15:	.	.	.	.	
16:	.	.	.	.	
17:	.	.	.	.	
18:	.	.	.	.	
19:	.	.	.	.	
20:	.	.	.	.	
21:	.	.	.	.	
22:	.	.	.	.	
23:	.	.	.	.	
24:	ESC for new TRSDOS disk	T to transfer a file		BREAK to quit	

Figure 8.171 Display of TRSDOS Diskette Directory

The cursor will be positioned at the beginning of the first file name displayed on the screen, and a summary of the actions you may take will appear on the bottom line of the display. If you want to mount a different TRSDOS diskette, press the <esc> key on the console keyboard. This will cause TRS2CPM to start over from the beginning. Pressing the <break> key will cause TRS2CPM to return to CP/M.

To transfer a file, move the cursor to the beginning of the file name using the cursor arrow keys on the keyboard. The cursor will move one complete file name for each press of an arrow key. (If it cannot move in the direction specified, the cursor will not move at all.) When the cursor is positioned at the name of the desired file, press the "T" key on the console keyboard. TRS2CPM will read the TRSDOS directory to find out what type of file is being transferred, and will display its findings on the right side of the screen (Figure 8.172).

1:	SYSTEM .SYS	SYSRES .SYS	BACKUP .	PATCH .	
2:	XFERSYS .	FORMAT .	SYSTEM64.	TERMINAL.	
3:	MEMTEST .	BASIC .	HERZ50 .	LP11 .	
4:	PRTBKSP .	SYSTEM32.	DOC0M32 .	BASCOM32.	
5:	COMSUB32.	EXDATM32.	DATM32 .	DOC0M64 .	
6:	BASCOM64.	COMSUB64.	EXDATM64.	DATM64 .	
7:	.	.	.	.	TRS to CP/M
8:	.	.	.	.	Utility
9:	.	.	.	.	Version 2.xxx
10:	.	.	.	.	(c) 1980,81,83
11:	.	.	.	.	by
12:	.	.	.	.	Pickles & Trout
13:	.	.	.	.	all rights
14:	.	.	.	.	reserved
15:	.	.	.	.	
16:	.	.	.	.	
17:	.	.	.	.	
18:	.	.	.	.	
19:	.	.	.	.	
20:	.	.	.	.	
21:	.	.	.	.	
22:	.	.	.	.	
23:	.	.	.	.	
24:	Do you want a LINE FEED appended after each CARRIAGE RETURN? (Y/N):N<enter>				PRTBKSP. SIZE:1  Data File

Figure 8.172 Selecting a Data File for Transfer

When a data file is transferred, TRS2CPM will give you one of two options depending on whether the file has a fixed or variable record length. If the file has fixed record length, you have the option of having a line feed inserted after every carriage return in the file. The prompt for your response will appear as shown on Line 8.172-24. In most cases, you should specify "Y" if you are transferring a text file and "N" for some other type of file. In the example shown in Figure 8.172, the user elected not to have line feeds inserted. After your response is entered, TRS2CPM will clear Line 8.172-24 and proceed to transfer the file.

If the file has variable length records, TRS2CPM will give you the option of having each record end with a carriage return and line feed. The console display will appear as shown in Figure 8.172 except that the bottom line of the display will be replaced by the line shown in Figure 8.173. If you respond affirmatively to this question, TRS2CPM will remove the record length count from the beginning of each record and add a carriage return and line feed to the end of each record as it is transferred. You will probably want to respond affirmatively if you are transferring files containing text. If you respond negatively, the file will be transferred on a byte-for-byte basis with no changes being made.

Do you want each variable length record to end with CRLF? (Y/N)

Figure 8.173 Option Prompt for Variable Length Record Data Files

If you have selected a load file for transfer, the load address and execute address (if defined) will be displayed as illustrated in Figure 8.174. Note that the file being transferred in this example has no defined execute address. If one were specified, it would appear immediately below the load address on the display. Since load files are always converted to hex format files on the CP/M disk, the transfer will take considerably longer than for a data file of the same size. In addition, the file generated on the CP/M diskette will be longer than the original file. The last line of the display will be cleared when TRS2CPM begins to transfer the file.

1:	SYSTEM	.SYS	SYSRES	.SYS	BACKUP	.	PATCH	.	TRS to CP/M
2:	XFERSYS	.	FORMAT	.	SYSTEM64	.	TERMINAL	.	Utility
3:	MEMTEST	.	BASIC	.	HERZ50	.	LPII	.	Version 2.0xx
4:	PRTBKSP	.	SYSTEM32	.	DOCOM32	.	BASCOM32	.	(c) 1980,81,83
5:	COMSUB32	.	EXDATM32	.	DATM32	.	DOCOM64	.	by
6:	BASCOM64	.	COMSUB64	.	EXDATM64	.	DATM64	.	Pickles & Trout
7:	.	.	.	.	.	.	.	.	all rights
8:	.	.	.	.	.	.	.	.	reserved
9:	.	.	.	.	.	.	.	.	
10:	.	.	.	.	.	.	.	.	
11:	.	.	.	.	.	.	.	.	
12:	.	.	.	.	.	.	.	.	
13:	.	.	.	.	.	.	.	.	
14:	.	.	.	.	.	.	.	.	
15:	.	.	.	.	.	.	.	.	
16:	.	.	.	.	.	.	.	.	DATM64.
17:	.	.	.	.	.	.	.	.	SIZE:4
18:	.	.	.	.	.	.	.	.	LOAD:EC60
19:	.	.	.	.	.	.	.	.	
20:	.	.	.	.	.	.	.	.	
21:	.	.	.	.	.	.	.	.	
22:	.	.	.	.	.	.	.	.	
23:	.	.	.	.	.	.	.	.	
24:	ESC for new TRSDOS disk	.	T to transfer a file	.	BREAK to quit	.			

Figure 8.174 Selecting a Load File for Transfer

### Possible Error Messages

TRS2CPM can display several different error messages during operation. The general form of these messages is a line at the bottom of the display screen with the message flashing between normal and reverse video. If your computer has a Pickles & Trout Clock/Calendar/Bell board (CCB) installed, the bell will beep as the message is flashed. The message will continue to flash until you press the <enter> or <break> key on the keyboard. TRS2CPM will then return to its opening dialog in most cases, since an error could cause it to malfunction.

#### **Drive not Ready**

The source drive is not ready. Neglecting to mount the TRSDOS diskette when requested is the usual cause of this error.

#### **READ ADDR ERROR**

A problem has been encountered in reading the TRSDOS diskette. This message usually indicates a bad spot on the diskette.

#### **SEEK ERROR**

An error has occurred during the attempt to position the read/write head at a specific location on the TRSDOS diskette.

#### **READ ERROR**

An error was encountered while reading information from the TRSDOS diskette. It usually indicates a bad spot on the diskette.

#### **TRANSFER ABORTED: THIS SPECIAL SYSTEM FILE CANNOT BE TRANSFERRED**

You have attempted to transfer a load file containing load addresses that cannot be meaningfully represented in an Intel hex format file.

#### **Not a valid TRSDOS disk**

The diskette mounted on the source drive is not a TRSDOS diskette.

The operating system may also display error messages if an error occurs while the destination diskette is being accessed. A system message will appear on the bottom line of the display, and it may be incomplete. Since the first 23 lines of the display are locked into place during the execution of TRS2CPM, the display can show only the last line of a multiple-line message issued by the system.

**8.32 Utility name: VERIFY**

**Purpose:** To verify that a copy of a P&T CP/M 2 utility program has not been altered.

**General Description**

Most of the utility programs supplied with P&T CP/M 2 have internal codes to allow you to determine if the copy you are using is valid. The VERIFY program checks these codes and reports to you whether the copy is correct or not. VERIFY can also report the patch numbers of the patches that have been installed on a program using the PATCH utility program.

You can run VERIFY on a program file as many times as you wish; it will not damage or alter the file in any way. If a program file does not verify, it indicates that something has been altered since it was copied from the master diskette. If a program has been altered, it may not perform properly. Changes made by the PATCH program will not affect VERIFY.

**Note:** VERIFY will only work on utility programs supplied with P&T CP/M 2. In particular, it is not a file comparison program that allows you to compare a working copy of any file with another copy (say on a master disk).

**Using VERIFY**

Figure 8.175 shows the general form of the command line that is used to verify a program file. You may specify either a unique file name (as shown) or a wild card file name. If you specify a wildcard file name, all files matching the name will be verified.

VERIFY (file name) [P]  
(file name) is the name of the file to be verified.  
[P] is optional and, if present, causes VERIFY to list the patch numbers of all patches that have been installed

Figure 8.175 General Form of VERIFY Command Line

Figure 8.176 shows an example of using VERIFY on the PATCH.COM program file and Figure 8.177 shows an example of using VERIFY with the P option. If patches had been installed in the program file, their numbers would have been reported on Line 8.177-5.

1: A>VERIFY PATCH.COM<enter>  
2:  
3: PATCH.COM verified.  
4:  
5: A>

Figure 8.176 Example of Using Verify

```
1: A>VERIFY PATCH.COM P<enter>
2:
3: PATCH.COM verified.
4:
5: Patches installed: None.
6:
7: A>
```

Figure 8.177 Example of Using Verify with the P Option

### Possible Error Messages

#### No files found.

VERIFY could not find any files on the disk that match the file name (either unique or wild card) that you gave on the command line. This can be caused by omitting the file name altogether.

#### filename.ext not found.

"filename.ext" will be replaced by the name of the program file you specified on the command line. This message is given if the file cannot be found on the specified disk. Check to see that the program is actually on the disk and run VERIFY again. This error can be caused by mis-typing the file name.

#### filename.ext is not setup for use with VERIFY.

"filename.ext" will be replaced by the name of the program file you specified on the command line. This message indicates that you have tried to run VERIFY on a program file with which it cannot work. Most programs that are distributed with P&T CP/M 2 have been prepared for VERIFY. VERIFY will not work with programs from other sources

#### filename.ext has internal error(s), doesn't verify.

"filename.ext" will be replaced by the name of the program file you specified on the command line. This message indicates that VERIFY has found one or more errors in the program file. You should make another copy of the file from your master diskette.

#### filename.ext read error.

A disk error occurred while reading the file named "filename.ext" to perform the verification.

#### Not enough memory for all files specified.

This message is given if insufficient memory is available to hold information about all the files being verified. It is very unlikely that you will encounter this error. If you do, check the amount of reserved memory above the operating system. Reducing this amount may allow VERIFY to run. Alternately you can specify a more restrictive wild card file name so that VERIFY will check fewer files.

**8.33 Utility name: XSUB**

**Purpose:** To allow SUBMIT files to provide console input to some programs.

### General Description

The standard SUBMIT utility of P&T CP/M 2 can provide the automatic execution of a series of programs. However, it does not provide a means of input to a program. Once a program is running, any console input it requires must be typed in from the console unless SUBMIT is run in conjunction with XSUB.

XSUB allows a SUBMIT file to supply input to programs that use the CP/M buffered console input function (BDOS function 10). It causes the next line from the SUB file to be returned any time that a program calls BDOS function 10.

Although XSUB greatly enhances the capabilities of SUBMIT, a thorough understanding of its limitations is necessary to make effective use of it. In particular, you may not have a control character in a command line nor an empty command line (the result of pressing <enter> by itself). Unfortunately, these restrictions prevent the complete use of some programs. For example, some programs return to the system only after a blank command line or a <ctl-C> (<break>) is entered. Since neither of these can be present in a SUB file, a SUB file which executes such a program and then feeds it commands cannot provide a return to the operating system.

The next most severe limitation of XSUB is that it can feed input into programs only through the buffered console input function. With more and more interactive programs being written, the use of buffered console input is declining. Some programs use both buffered and direct console input. If XSUB is used with these programs, the buffered input will come from the SUB file, but the direct input will have to come from the console.

### Using XSUB

To use XSUB, a SUB file should be prepared exactly as it was for the standard SUBMIT program, except that the first program executed should be XSUB. The rest of the SUB file may include commands which execute programs interspersed with separate lines of data to be fed into them. Once executed, XSUB relocates itself just under the CCP in memory and makes appropriate patches to the system to avoid being over-written by other programs. As a result, there is somewhat less memory available to transient programs while XSUB resides there.

Figure 8.178 gives an example of a SUB file named "TEST.SUB" which makes use of XSUB. The SUB file creates a disk file containing a form feed character followed by three <ctrl-Z> characters. Such a "form feed file" is useful when transferring files to the printer using PIP. It can be sent between the other files to advance the paper to the top of the next page.

```
1: : This is a demonstration of the use of SUBMIT with XSUB
2: :
3: xsub
4: ddt
5: s100
6: 0c
7: 1a
8: 1a
9: 1a
10: .
11: g0
12: save 1 ff
```

Figure 8.178 Example of a SUB File that uses XSUB

To create the disk file, the SUB file inserts the characters into the beginning of the transient program area (TPA) by means of the DDT program. It then saves the first page (256 bytes) of the TPA on disk as a file named FF.

Lines 8.178-1 and 8.178-2 are merely comment lines, since they start with a colon. Line 8.178-3 executes the XSUB program, which allows the SUB file to provide input into the DDT program. Line 8.178-4 executes DDT. Lines 8.178-5 through 8.178-10 use the "S" command of DDT to insert the characters into memory. Line 8.178-11 uses the "G" command of DDT to cause a warm boot and hence a return to the operating system. Finally, Line 8.178-12 uses the CCP SAVE command to save the first page of the TPA as a disk file named FF. Note that Lines 8.178-1 through 8.178-4 and Line 8.178-12 are all command lines to the CCP. Lines 8.178-5 through 8.178-11 are command lines to the DDT program.

Figure 8.179 shows the console display that results when "TEST.SUB" is submitted. Notice that XSUB displays the message "(xsub active)" whenever a warm boot occurs as long as it is in memory. XSUB is automatically de-activated after the last command of a SUB file is processed.

```
A>SUBMIT TEST<enter>
A> THIS IS A DEMONSTRATION OF THE USE OF SUBMIT WITH XSUB
A>
A>XSUB

A>DDT
DDT VERS 2.2
-S100

0100 01 0C
0101 BC 1A
0102 0F 1A
0103 C3 1A
0104 3D .
-GO

(xsub active)
A>SAVE 1 FF
A>
```

Figure 8.179 Console Display from Running the SUB File of Figure 8.178

### Possible Error Messages

#### Xsub Already Present

This message indicates that XSUB was already resident in memory before the command for its execution. No action is taken and the SUB file continues with the next command.

#### Requires CP/M Version 2.0 or Later

You are trying to use XSUB with an older version of CP/M or a non-CP/M operating system.

### 8.34 Examples of Using the Command Line Mode

The command line mode of the various utility programs is a very useful and powerful tool. This is especially true when you are setting up a system of programs for a non-technical user. It is possible to insulate the user from the details of the various utility programs (like FORMAT and DISKCHK) by making them appear as a normal part of the system. Of course the command line mode is also quite useful if you frequently perform certain operations and want to use a single command line rather than being prompted for information.

The following examples show how the command line mode of the utility programs might be used to automate some functions. Since they are intended to be illustrative, only the parts dealing with the utility programs is shown in detail. There are many other ways that the command line mode can be used.

#### Example 1 - Using a SUBMIT File in Software Production

If you have written software that you are distributing, you need some method of duplicating it. The following example assumes that there are three operations to perform to generate a distribution diskette: 1) format the diskette, 2) copy the programs to the diskette, and 3) serialize the programs.

You could start with a stack of blank diskettes and format them all. Then you could copy the programs to each diskette and finally perform the serialization. This technique means that you must make three passes through all of the diskettes. This involves a lot of handling and the associated chance of mistakes. Figure 8.180 shows a submit (named PRODUCE.SUB) file that can be used to perform all three operations on a diskette before moving on to the next diskette.

```
1: : Submit file to FORMAT -> COPY -> SERALIZE
2: :
3: format dr=d sd v pmount
4: clone e>d v t
5: serializ
6: submit produce
```

Figure 8.180 Example Production SUB File that Uses Utility Programs

Lines 8.180-1 and 8.180-2 are merely comment lines that will be displayed on the console as the submit file begins operation. This submit file assumes that the master diskette from which the distribution diskettes are to be made is mounted on drive C and that the diskettes to be produced will be mounted on drive B.

Line 8.180-3 executes the FORMAT program to format the new diskette at single density. Single density should always be used for software distributed on 8 inch diskettes since there is only one single density format used with CP/M but there is a multitude of double density formats. The "V" instruction is used so that FORMAT will make a verification pass on the diskette after it is formatted. The "PMOUNT" instruction is given to FORMAT so that it will prompt for a diskette to be mounted before the format operation begins. This eliminates the need to explicitly prompt the user to mount a diskette.

Line 8.180-4 executes the CLONE program to copy from the master diskette on drive C to the newly formatted diskette on drive B. The "V" instruction causes CLONE to verify each track after it is copied and the "T" instruction is given so that CLONE will stop when it encounters the first empty track on the source diskette.

Line 8.180-5 executes a program named serialize that installs the serial number in the newly copied diskette on drive B. It is assumed here that you would write this program to perform whatever serialization operations are needed.

Finally, on Line 8.180-6, the submit file resubmits itself. The production process will continue until you abort it. To abort the process, you need merely press the <break> key while the initial messages from the submit file are being displayed on the console.

Figure 8.181 shows the console display that results from running the PRODUCE submit file. Note that the only user input is the initial execution of the submit file (Line 8.181-1) and pressing <enter> on Line 8.181-7 after the new diskette is mounted. All other lines of the display come from either the programs that are executed or the submit file itself. Note that on Line 8.181-30, PRODUCE re-invokes itself which will start the process over again at line 8.181-2.

8

```
1: A>SUBMIT PRODUCE<enter>
2:
3: A>: SUBMIT FILE TO FORMAT -> COPY -> SERIALIZE
4: A>:
5: A>FORMAT DR=B SD V PMOUNT
6:
7: Mount disk to format on drive B: and press <ENTER> when ready to start : <enter>
8:
9: ** FORMAT **
10: Formatting disk in drive B: at Single-density.
11: .....
12: Format complete.
13:
14: Checking disk on drive B:
15: .....
16: Checking complete.
17:
18: A>CLONE C>B V T
19:
20:
21:
22: ** CLONE **
23: Copying from drive C: to drive B: at Single Density.
24: ++++++-----+
25: Found empty track -- Copy terminated after track 30
26:
27: A>SERIALIZ
28: Serial number 00005 - successful
29:
30: A>SUBMIT PRODUCE
```

Figure 8.181 Console Display when Using PRODUCE.SUB

**Example 2 - Periodic Checking of a Diskette**

In some cases it is desirable to use DISKCHK on a diskette from time to time to check for errors. If soft errors begin to show up, you have a chance to replace the diskette before any permanent errors occur and data is lost.

The following example uses the submit file (named JOBLSUB) shown in Figure 8.182 to execute two programs and then use DISKCHK to check the diskette for errors. Lines 8.182-1 and 8.182-2 execute two programs named PROG1 and PROG2, respectively. These might be text editing, data base, accounting, or some other kind of programs. The submit file then displays a comment on the console (Line 8.182-4) to indicate that the data diskette is about to be checked.

```
1: : prog1
2: : prog2
3: :
4: : Error checking the data diskette
5: :
6: diskchk dr=d /a:dchk,err
7: mbasic dchk
8: submit job1
```

Figure 8.182 Example of Submit File that Uses DISKCHK

On Line 8.182-6, the submit file executes the DISKCHK utility program to check the diskette mounted on drive D. All console output is directed to a disk file named DCHK.ERR on drive A; no messages appear on the console while the program is running. The user is not bothered by the messages coming from DISKCHK.

On Line 8.182-7, the submit file executes a Basic program that checks the output file created by DISKCHK for error messages. If any error messages are detected, the user is informed and is given a suggested course of action. Finally, the submit executes itself again (Line 8.182-8) and the whole process starts over. Figure 8.183 shows the source code of the Basic program.

```
10 ! This program reads the file A:DCHK.ERR which is created by
20 ! running DISKCHK. It scans the file for errors and, if it
30 ! finds any, it will print an appropriate message on the
40 ! console advising the user what action to take
50 !
55 FLAG=0
60 OPEN "I",1,"A:DCHK.ERR"
70 ON ERROR GOTO 1000           'go to 1000 when get to end of file
80 LINE INPUT #1,TEMP$          'read line from file
90 IF INSTR(TEMP$,">>>")=0 THEN 80
100 IF INSTR(TEMP$,"Soft") THEN 500
105 PRINT
110 PRINT "Your data diskette has developed a permanent bad spot."
120 PRINT "It is imperative that you replace it at once."
130 PRINT "Use the system menu to format a new diskette and"
140 PRINT "copy your working files to it."
150 KILL "A:$$.SUB"
160 SYSTEM
200 !
505 FLAG=1 : PRINT
505 PRINT "Your data diskette has developed a soft error."
510 PRINT "It is recommended that you replace it as soon as possible."
520 PRINT "Do you want to stop now to replace it (y/n)?"
530 RESP$=INPUT$(1)
540 IF RESP$<>"y" AND RESP$<>"Y" THEN 80 ELSE 150
600 !
1000 IF FLAG<>0 THEN SYSTEM
1010 PRINT : PRINT "Your data diskette checks out OK."
1020 SYSTEM
```

Figure 8.183 Basic Program to Check for Errors from DISKCHK

Lines 10-50 of the program are comments. On Line 55 the program sets a flag variable to 0. This flag is used to keep track of any soft errors that are detected. On Line 60 the program opens the file it is to read to check for errors. Line 70 directs any errors to line 1000 so that when all of the file is read, control will pass to the exit code in lines 1000-1020. Lines 80 and 90 form a loop that reads lines from the file until the end of the file occurs or a line containing ">>>" (denoting an error) is encountered.

If an error line is found, Line 100 checks it for the word "Soft". If the word "Soft" is found in the line, Lines 500-540 are executed to notify the user of the soft error and to find out what he wants to do about it. If "Soft" is not found on the error line, it is assumed to be a permanent error. In this case the user is not given a choice. The program automatically aborts the submit file that is in progress (Line 150) and returns to the system command level.

Assuming that the program is not terminated by an error, control finally passes to Line 1000 when the end of the input file is reached. If no soft errors had been encountered, the program displays a message indicating that the diskette is OK and returns to the system level.

Obviously much more involved error checking could be performed and another language might be preferable but this example does illustrate the kinds of checking that can be done. Figure 8.184 shows the console output from running the submit file shown in Figure 8.182.

```
1: A>SUBMIT JOB1<enter>
2:
3: -----> PROG1 is executed here
4:
5: -----> PROG2 is executed here
6:
7: A>:
8: A>: ERROR CHECKING THE DATA DISKETTE
9: A>:
10: A>DISKCHK DR=D /A:DCHK.ERR
11:
12: A>MBASIC DCHK
13: BASIC-80 Rev. 5.21
14: [CP/M Version]
15: Copyright 1977-1981 (C) by Microsoft
16: Created: 28-Jul-81
17: 27704 Bytes free
18:
19: Your data diskette checks out OK.
20:
21: A>SUBMIT JOB1
```

Figure 8.184 Example of Console Display when Using JOB1 Submit File

**Example 3 - Using Utility Programs from Dbase II**

Dbase II (and other languages) allows you to execute a series of programs from a single command line within the Dbase command file. Figure 8.185 shows a skeletal Dbase command file (named MAIN.CMD) that has an option to format and test a new diskette.

```
1: * Example of using P&T utility programs from a Dbase II program
2: store '0' to funct
3:
4: do while val(funct)<>99
5:   erase
6:   ? "Functions:"
7:   ? "1. Enter new data"
8:   ? "2. Print data"
9:   .
10:  .
11:  .
12:  ? "85. Format and test a new diskette"
13:  ? "99. Exit to system"
14:
15: accept "Enter function number " to funct
16:
17: do case
18:   case val(funct)=1
19:     do entdat
20:
21:   case val(funct)=2
22:     do prndat
23:     .
24:     .
25:     .
26:   case val(funct)=85
27:     ?
28:     ?? "Mount a new diskette on drive D and press <enter>"
29:     wait to temp
30:     quit to "format dr=d,dd","disktest dr=d","dbase main"
31:
32: endcase
33:
34: enddo
35: quit
```

Figure 8.185 Example CMD file for Using Utility Programs from Dbase II

Lines 8.185-4 to 8.185-15 display a menu of functions on the console and ask the user to enter the one to be performed. This menu would be expanded to include all the various functions that you have included in the program. Lines 8.185-17 to 8.185-32 execute one set of instructions according to the function number entered in response to the menu. In particular, Lines 8.185-26 to 8.185-30 handle function number 85, "Format and test a new diskette".

Line 8.185-28 prompts the user to mount a new diskette on drive D and press the <enter> key. Line 8.185-29 waits for the <enter>. Line 8.185-30 exits Dbase II and starts a sequence of three command lines. The first one "format dr=d,dd" executes the FORMAT program to format the diskette mounted on drive D at double density.

The second command line, "disktest dr=d" executes the utility program DISKTEST to make a thorough test of the diskette just formatted. The third command line "dbase main" re-executes the command file shown in Figure 8.185 so that the user finds himself back at the menu presented by the command file. All messages from the utility programs are displayed on the console so that the user can take appropriate action should a error occur.

Figure 8.186 shows the console output when the user selects function 85. Note that after Dbase is executed again on Line 8.186-39, it will clear the display and start over with the menu as shown on Lines 8.186-1 to 8.186-6.

```
1: Functions:
2: 1. Enter new data
3: 2. Print data
4: 85. Format and test a new diskette
5: 99. Exit to system
6: Enter function number :85
7: Mount a new diskette on drive D and press <enter>
8: WAITING
9: *** END RUN      dBASE II      ***
10:
11:
12:
13: A>FORMAT DR=D,DD
14:
15:
16: ** FORMAT **
17: Formatting Single-sided disk in drive D: at Double-density.
18: ****
19: Format complete.
20:
21: A>DISKTEST DR=D
22:
23:
24: ** DISKTEST **
25: Testing Double-density disk in drive D:
26:
27:
28: Beginning pass with pattern = 00
29: ****
30:
31: Beginning pass with pattern = FF
32: ****
33:
34: Beginning pass with pattern = DBB6
35: ****
36:
37: Testing complete.
38:
39: A>DBASE MAIN
```

Figure 8.186 Example of Using CMD File of Figure yy98

## 9.1 Introduction

P&T CP/M 2 will issue error messages when various system errors occur. The most commonly encountered messages have to do with disk related errors. Other messages deal with problems related to the parallel printer port. These messages depend on the parallel printer port module (if any) that is included in the system when it is loaded. The messages given by the standard parallel printer port module are described in Chapter 13 on the parallel printer port.

There are two types of disk error messages given by P&T CP/M 2: BDOS messages and BIOS messages. BDOS error messages are generated by the portion of P&T CP/M 2 that was written by Digital Research. They are usually very general in nature, and they do not indicate the cause of hardware errors. In order to provide more detailed information on hardware errors, P&T CP/M 2 has implemented several BIOS error messages which include an error code that supplies additional information. A BIOS and a BDOS error message will usually be displayed together when a disk error occurs.

If you have a large number of disk errors and contact Pickles & Trout for assistance, please have the exact error messages available so that we can help you. If you do not have these messages, we usually cannot do more than give very general suggestions. You might consider keeping a log containing the error message and other information such as how long the machine had been on and what you were doing when the error occurred.

There is one disk related message that, while not strictly resulting from an error, will be discussed here. This message results from the system attempting to access a floppy diskette drive which is not ready. When P&T CP/M 2 attempts to access a diskette drive, a check is made to see that a diskette is mounted on the drive, the door is closed, and the diskette is up to speed. If one of these conditions is not met the drive is said to be "not ready". When a drive is found to be not ready, P&T CP/M 2 flashes the message shown in Figure 9.1 in the center of the console display.

Drive X is not ready - please check it.

Figure 9.1 Message Flashed When a Drive is Not Ready

The "x" in Figure 9.1 will be replaced with the letter of the logical drive which is not ready (e.g. A, or B, or C, ...). When you see this message flashing, you have three options available to you.

1. You may make the drive ready (this typically means mounting the diskette or closing the drive door). As soon as the drive is ready, the flashing message will be removed and the screen will be restored to its appearance before the message was displayed. Operation will then resume automatically.
2. You can press the <break> key. This will abort any program that is running and return to the command level of the system by performing a warm boot. When you are returned to the command level, you will return to the same drive that was the current default drive the last time you were at the command level.

3. You can press the <F1> key. This will abort any program that is running and return to the command level of the system by performing a warm boot. Unlike pressing <break>, pressing <F1> will always return you to logical drive A as the current default drive regardless of what the current default drive was when you were last at the command level of the system. Due to the nature of CP/M 2, it is possible for the system to get "stuck" in a situation where it always gives a "not ready" error after a warm boot. Although this is not common, you may encounter it occasionally or you may want to return to the command level with drive A as the current default drive. In these cases, press the <F1> when the "drive not ready" message is flashing.

## 9.2 BDOS Error Messages

The four generalized error messages shown in Figure 9.2 are given by the BDOS portion of the system.

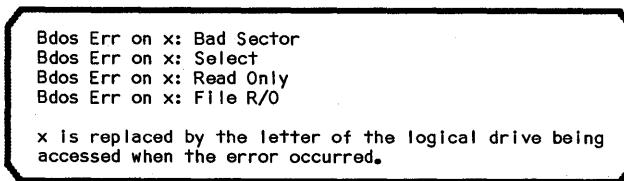


Figure 9.2 BDOS Error Messages

The "Bad Sector" BDOS message is given whenever any error is detected during the reading or writing of a sector on the disk. After the message is given, the system waits for console input. If <enter> is pressed, the bad sector is ignored and operations continue. If <break> or <ctl-C> is pressed, the operation (and the program) is aborted, a warm boot is done, and the system returns to the command level. NOTE: In some cases, pressing <enter> can jeopardize the integrity of data on the diskette being accessed.

The "Select" BDOS message is given when an attempt is made to select a drive which is not valid on the system. After the message is given, any console input will result in a warm boot and a return to command level.

The "Read Only" BDOS message is given when an attempt is made to write on a disk that the system has set to "read only" status. As noted earlier, the system keeps a check on each disk for changes which may have occurred since it was last accessed. Actually, the check is based on information which the system keeps on the disk's directory. When this information does not agree with the directory itself, the system assumes that disks have been exchanged without an intervening warm boot or disk system reset. Consequently, the system sets the disk to "read only" status, thus protecting your files from possible damage. A disk can also be set to "read only" status with the STAT utility program or a standard system call. After a "Read Only" error message is given, any console input will cause a warm boot.  
(NOTE: After the warm boot, the disk will be reset to "read/write" status.)

The "File R/O" BDOS message is given when an attempt is made to write to, rename, or erase a file that is marked as "read only". Once the message is given, the system waits for input from the console. If the next character entered is <enter>, the system will ignore the attempted write and continue. If <break> or <ctl-C> is entered, the system will abort the current operation, perform a warm boot, and return the user to the system command level. (NOTE: A warm boot has no effect on a file that has been tagged as "read only".)

### 9.3 BIOS Error Messages

The BIOS portion of P&T CP/M 2 displays several additional error messages to describe the nature of disk errors more specifically. The general form of these messages is shown in Figure 9.3.

xxxxxx error, code = cccc

Figure 9.3 General Form of BIOS Error Messages

In the BIOS error messages, "xxxxxx" is a word or phrase indicating the type of error, and "cccc" is an error code which gives more details of the error being reported. Figure 9.4 shows the various words and phrases that can appear in the place of the "xxxxxx".

Read	Track
Write	Lock-out
Select	Hard disk read
Locate	Hard disk write
Boot	Hard disk not ready
Seek	Hard disk write protect

Figure 9.4 Words and Phrases Used in BIOS Error Messages

The error code consists of from three to five characters. Since there are a very large number of possible error codes, it is impossible to try to describe them all in a manual like this. For this reason, the ERROR utility program has been supplied with the system. If you want a further explanation of a particular error code, you need merely execute the ERROR program using a command line like that shown in Figure 9.5. Further information on the ERROR program is given in Section 8.15 of this manual.

ERROR code

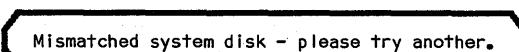
"code" is replaced by the error code for which  
you wish an explanation.

Figure 9.5 General Form of Command Line to Execute ERROR

### 9.4 Warm Boot Error Message

Another type of error message that may be given by the BIOS occurs when a warm boot operation is attempted from an improperly set up system diskette. During the warm boot operation on some systems, a portion of the operating system (the CCP and BDOS) must be read in from the system diskette. During the system loading operation, the copy of these parts of the system that are kept on the diskette were set up to run at a particular location in memory. The location is determined by the combination of modules you selected for inclusion in the system.

If, when a warm boot is attempted, the system finds that these portions of the system are not on the diskette or they have not been properly set up to run with the system in memory, the error message shown in Figure 9.6 is displayed on the console. If this error should occur, you have two possible actions. You can mount another diskette with (hopefully) a correct copy of the CCP and BDOS and press any key to attempt the warm boot from it. Alternatively, you can RESET the computer to reload the system from the currently mounted system diskette.



Mismatched system disk - please try another.

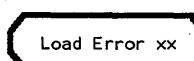
Figure 9.6 Error Message Flashed for Warm Boot Error

This type of error may occur if you change diskettes on the system drive and then attempt to perform a warm boot. You may change diskettes on the system drive as long as the diskette mounted when a warm boot is performed has the CCP and BDOS properly modified. You can insure that they have been properly modified in two ways:

1. You can use CLONE to copy the system tracks from the diskette used to initially load the system to the other diskettes that you will be using on the system drive.
2. You can create identical working system diskettes with exactly the same BIOSPARM.PNT and BIOSMODS.PNT files and then load the system at least once from each of them. The process of loading the system from each system diskette will insure that the copies of CCP and BDOS on each of them are properly modified. Since the same modules are included in the system loaded from each of the diskettes, the CCP and BDOS will be identically modified on each one.

## 9.5 System Load Error Messages

If an error occurs while the system is being loaded from disk, an error message of the general form shown in Figure 9.7 is displayed on the console. After this message is displayed, the machine will stop. In order to continue, you must remedy the problem and RESET the computer to load the system again. The error message includes a numeric code (shown by "xx") which indicates what caused the error.



Load Error xx

Figure 9.7 Form of Loader Error Message

The error codes that might appear in the loader error message are as follows:

- 1 An error occurred while attempting to restore the system boot drive. This can be caused by a hardware problem or by setting the drive step rate to an improper value.
- 2 An error occurred while attempting to move the disk head to a specific track on the disk. This is usually caused by setting the drive step rate to an improper value but also can be caused by a bad spot on the disk.
- 3 An error occurred while reading from the disk. This is usually indicative of a bad spot on the disk.
- 4 An error occurred while writing to the disk. This happens when the loaded tries to write the set up copy of CCP and BDOS back to the system disk. It is usually caused by a write protected system diskette.
- 10 The BIOSPARMLPNT file cannot be found on the disk. Check that it is a system disk.

- 11-39 The BIOSPARM.PNT file is incorrect or incomplete. It may have been accidentally modified by some program or damaged in some other way. Getting a new copy of this file from another disk will usually solve the problem.
- 40 The BIOSMODS.PNT file cannot be found on the disk. Check that it is a system disk.
- 41-49 The BIOSMODS.PNT file is incorrect or incomplete. It may have been accidentally modified by some program or damaged in some other way. Getting a new copy of this file from another disk will usually solve the problem.

## 9.6 Module Error Messages

If you or a program attempt to access a system function that requires a module that was not included in the system when it was loaded, the error message shown in Figure 9.8 is flashed on the screen.

Module Absent

Figure 9.8 Error Message if a Module is not Present When Accessed

When this message is displayed, you have two options:

1. Press the <enter> key to continue as if nothing happened. Note that if another attempt is made to perform the same function, the error message will be displayed again.
2. Press the <break> key to return to the command level of the system by performing a warm boot. This will abort any program that is running at the time.

## NOTES

## 10.1 General Comments

The TRS-80 Model II/12/16 has a built-in video display capable of displaying 24 rows of 80 characters. It can display in normal and reverse video and has a limited set of graphics characters. Through the video display handler of P&T CP/M 2, programs have access to these capabilities and to many powerful display control functions. The display may be accessed by either the CON: or LST: logical devices. The physical device name used by P&T CP/M 2 for the built-in display is "CRT". If the CRT physical device is assigned to either CON: or LST:, all output to either of these logical device will be sent to the video display.

The standard video display handler does not emulate any one video terminal. When P&T CP/M 2 was first introduced, there were no common terminals that could be emulated and still retain access to all the features of the video display. Although terminals have become available which have very similar features, the methods of accessing the various functions with our standard handler have not been changed in order to remain compatible with previous versions of P&T CP/M 2. Some features have been added but they do not affect the functions available in previous versions.

In the current version of P&T CP/M 2, a module is available that will emulate an ADM-3A terminal (see Section 7.5). When this module is included in the system, the console and keyboard will appear (as much as possible) like an ADM-3A terminal. If you are using software that must be customized to the terminal and it does not provide a customization for P&T CP/M 2, you can customize it for an ADM-3A terminal if you include the ADM3A module in the system. Note: If possible, it is usually better to customize the software (using its customization procedure) to use the standard P&T CP/M 2 console display functions. This will result in more efficient use of memory and slightly faster operation.

## 10.2 Display Control Codes

Most of the special functions of the video display are executed by sending control codes to the display driver. (A few of them are accessed by the special system functions, and one is accessed by an escape sequence.) Since control codes are ASCII characters, most programs that can output to the console can perform display control functions. Figure 10.1 lists all the special functions that can be executed by a single control character.

numeric code (dec)	numeric code (hex)	control key	function
1	01	<ctl-A>	clear from cursor to end of line
2	02	<ctl-B>	clear from cursor to end of screen
4	04	<ctl-D>	insert line
6	06	<ctl-F>	home cursor without clearing screen
7	07	<ctl-G>	ring bell (with P&T CCB or Model 12)
8	08	<ctl-H>	non-destructive backspace (move cursor left)
9	09	<ctl-I>	tab cursor
10	0A	<ctl-J>	line feed (move cursor down)
11	0B	<ctl-K>	delete line
12	0C	<ctl-L>	clear screen and home cursor
13	0D	<ctl-M>	carriage return (move cursor to left end of line)
14	0E	<ctl-N>	begin reverse video
15	0F	<ctl-O>	end reverse video
17	11	<ctl-Q>	begin graphics
20	14	<ctl-T>	end graphics
22	16	<ctl-V>	set display to wrap mode
23	17	<ctl-W>	set display to non-wrap mode
25	19	<ctl-Y>	turn off cursor
26	1A	<ctl-Z>	turn on cursor
28	1C		move cursor left
29	1D		move cursor right
30	1E		move cursor up
31	1F		move cursor down
127	7F		destructive backspace

Figure 10.1 Control Character Functions of the Video Display

Further description of these functions is given below:

**<ctl-A> - clear to end of line**

The line containing the cursor is filled with blanks from the current cursor position to the end of the line. The cursor position does not change.

**<ctl-B> - clear to end of screen**

The entire display is filled with blanks from the present cursor position to the bottom rightmost character location. The cursor position does not change.

**<ctl-D> - insert line**

The line containing the cursor's position and all lines below it on the display move down by one line, a new line filled with blanks is inserted to contain the unmoved cursor, and the cursor moves to the left of that line. If the cursor is on the last line of the display, all lines of the display move up one line, the last line is filled with blanks, and the cursor moves to the left end of it.

**<ctl-F> - home cursor**

The cursor moves to the upper leftmost position on the display. Only the cursor position is changed; no characters on the display are altered.

**<ctl-G> - bell**

If the P&T CCB clock/calendar/bell board is installed in the computer, the bell is rung for about 0.3 seconds. If the board has not been installed, no action is taken. If the system is used with a Model 12 computer, the internal bell of the Model 12 will be sounded for about 0.3 seconds.

**<ctl-H> - back space**

The cursor moves one position to the left of its current position. If the cursor is in column 0 (the leftmost column) and the display is in wrap mode, the cursor moves to the rightmost position of the previous line. If the display is in non-wrap mode, no action is taken. In no case will action be taken if the cursor is at the home position (upper leftmost position). Only the cursor position is changed; no characters on the display are altered.

**<ctl-D> - tab cursor**

The cursor moves to the right of its current position to the next multiple of 8 columns from the left edge of the display. If that location is beyond the right edge of the display and the display is in wrap mode, the cursor moves to the first position of the next line. If the display is not in wrap mode in this situation, the cursor moves to or remains in the rightmost column of the current line. Only the cursor position is changed; no characters on the display are altered.

**<ctl-J> - line feed**

The cursor moves down one line on the display while remaining in the same column. If the cursor is on the last line of the display, all lines of the display move up by one line, the last line is filled with blanks, and the cursor is not moved.

**<ctl-K> - delete line**

The line containing the cursor's current position is deleted, all lower lines move up one line, and the last line of the display is cleared. The cursor moves to the beginning of the line which replaced the deleted line. If the cursor is on the last line, the last line is cleared (filled with blanks) and the cursor moves to its beginning.

**<ctl-L> - clear screen and home cursor**

The entire display is cleared (filled with blanks) and the cursor moves to the upper leftmost character position.

**<ctl-M> - carriage return**

The cursor moves to the leftmost character position of the line on which it is positioned. Only the cursor position is changed; no characters on the display are altered.

**<ctl-N> - begin reverse video**

All characters sent to the display following this character are displayed in reverse video (black characters on a white background) until a <ctl-O> is sent.

**<ctl-O> - end reverse video**

All characters following this one return to normal video (white characters on a black background).

**<ctl-Q> - begin graphics**

All characters sent to the display following this character will be displayed as graphics characters until a <ctl-T> is sent. See below for further information on using graphics characters.

**<ctl-T> - end graphics**

All succeeding characters are displayed as normal characters.

**<ctl-V> - set display to wrap mode**

Wrap mode is enabled. In wrap mode, the cursor automatically advances to the beginning of the next line when it reaches the end of a display line. For reverse cursor movement, the cursor will automatically move to the end of the previous line when it reaches the beginning of a line.

**<ctl-W> - set display to non-wrap mode**

Wrap mode is disabled. In non-wrap mode, cursor movement control characters will not cause the cursor to move past the end of a line for forward movement or the beginning of a line for reverse movement.

**<ctl-Y> - turn off the cursor**

Upon receipt of <ctl-Y> the cursor is made invisible. While invisible, the cursor can still be positioned and it does move normally as characters are sent to the display; it is just not displayed.

**<ctl-Z> - turn on the cursor**

Upon receipt of <ctl-Z> the cursor is made visible again. If the cursor was already visible, <ctl-Z> has no effect.

**(1Ch) - move cursor left**

The same action as that of the backspace, <ctl-H>.

**(1Dh) - move cursor right**

The cursor moves one position to the right of its current position. If the cursor is in column 79 (the rightmost column) and the display is in the wrap mode, the cursor moves to the leftmost position of the next line. If the cursor is in column 79 of the last line and the display is in wrap mode, all lines on the display move up one line, a blank line is inserted at the bottom of the display, and the cursor moves to the leftmost position of the last line. If the cursor is in column 79 of any line and the display is in the non-wrap mode, no action is taken.

**(1Eh) - move cursor up**

The cursor moves up one line while maintaining the same column position. If the cursor is on the top line of the display, no action is taken. Only the cursor position is changed; no characters on the display are altered.

**(1Fh) - move cursor down**

Same action as that of the line feed, <ctl-J>.

**(7Fh) - destructive backspace**

The cursor moves one position to the left and any character at that location will be replaced with a blank. If the cursor is in the upper leftmost position of the display, no action is taken. If the cursor is in the leftmost position of any other line and the display is in the wrap mode, the rightmost position of the previous line will be replaced by a blank and the cursor will move there. If the display is in non-wrap mode and the cursor is in the leftmost position of a line, no action is taken.

## 10.3 Cursor Addressing

An escape sequence of 4 characters allows programs to move the cursor to any specific location on the display. Reference to the location is made by giving a row and column number. The row numbers start from 0 at the top of the display and progress to 23 at the bottom. The column numbers start with 0 at the leftmost end

of a line and progress to 79 at the rightmost end of a line. Figure 10.2 diagrams this scheme.

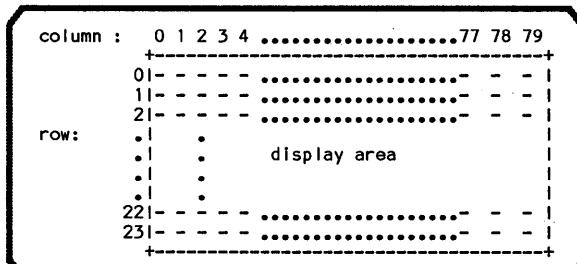


Figure 10.2 Row and Column Positions on the Video Display

The escape sequence for cursor positioning is shown in Figure 10.3.

```
ESC Y rowaddr coladdr
row address code = (row number) + 32
column address code = (column number) + 32
```

Figure 10.3 Cursor Positioning Escape Sequence

The first member of the escape sequence is the ASCII escape character (27 decimal, 1B hex), which is generated by the <esc> key. The second is upper case Y. Completing the sequence are, respectively, the row address code and the column address code. To determine the correct code for a particular location, you would first determine the desired row and column numbers and add 32 to each. For each resulting number, you would send out the ASCII character which has that number as its binary code. These codes are given in Appendix A of this manual, but it may not be necessary to refer to them. There are functions in many languages (such as the CHR\$ function in Basic-80) which directly send out the binary code (character) corresponding to a given decimal number.

As an example, suppose that your program needs to position the cursor in row 10, column 38. You would then need to send out the character codes for the numbers 42 and 70, which happen to be the asterisk (\*) and upper case F. The complete escape sequence would consist of the characters: <esc>Y\*F. As noted above, the program may be able to utilize functions that will send out a character code directly. For example, this program line in Basic-80 could be used to position the cursor at row 10, column 38: PRINT CHR\$(27); "Y", CHR\$(42), CHR\$(70).

Adding 32 to the row and column numbers insures that the needed codes are printing characters and not control characters. Control characters are avoided because they can be modified by certain languages as they are sent to the console. Several languages, for example, expand any tab character with spaces. Obviously, a problem would arise with such a language if a tab character were part of an escape sequence.

The escape sequence described above only affects the position of the cursor. No other changes will be made to the display or the operating mode.

## 10.4 Graphics Mode

When the display is in the graphics mode, all ASCII characters starting with space (20h) and continuing through tilde (7Eh) are interpreted as graphics characters. These characters are fully described the TRS-80 Model II/12/16 documentation.

The conversion used in the display handler of P&T CP/M 2 causes several regular ASCII characters to be mapped to each graphics character. Figure 10.4 gives a complete table of this correspondence. Internal limitations prevent some programs from sending out certain characters. The multiple choices for each graphics character allow such programs ample alternatives so that the full range of graphics is available.

To send out a graphics character, merely shift to graphics mode by sending <ctl-Q>, send any of the characters corresponding to the desired graphics character (using Figure 10.4), and then send <ctl-T> to return to normal. All of the graphics characters may be displayed in reverse or normal video by means of the video mode functions described above.

While in the graphics mode, all other control characters (such as <ctl-A> to erase to the end of the line) and the cursor positioning escape sequence are fully operational and may be used at any time. The example given later in this chapter shows the use of cursor positioning while in the graphics mode.

To get this graphics char as shown in the TRS-80 documentation	Send one of these characters while in the graphics display mode	To get this graphics char as shown in the TRS-80 documentation	Send one of these characters while in the graphics display mode
00	space, @, accent grave	10	0, P, p
01	!, A, a	11	1, Q, q
02	", B, b	12	2, R, r
03	#, C, c	13	3, S, s
04	\$, D, d	14	4, T, t
05	%, E, e	15	5, U, u
06	&, F, f	16	6, V, v
07	', G, g	17	7, W, w
08	(, H, h	18	8, X, x
09	), I, i	19	9, Y, y
0A	*, J, j	1A	: , Z, z
0B	+, K, k	1B	; , { , l
0C	comma, L, l	1C	<, \, ,
0D	-, M, m	1D	=, }, }
0E	period, N, n	1E	>, ^, ~
0F	/, O, o	1F	?,-

Figure 10.4 Graphics Mode Character Correspondence

## 10.5 Languages with Auto New Line

The video control functions described in this section are usually incorporated into programs when complete control over the console display is desired. As an example, a program may use the cursor addressing and graphics characters to draw a form on the display and then move to each location of the form to request information. In cases like this, the program is taking complete responsibility for the console display and does not want interference from any source.

Several languages send a carriage return and line feed to the console display after a certain number of characters have been sent to it. This can result in utter chaos when a program begins to manipulate the display, since the language may be doing

things the program does not expect. Fortunately, most of the languages that incorporate this feature also include a method for defeating it. In particular, there is a command called WIDTH in some versions of Basic which sets the number of characters which are sent to the console before an automatic carriage return and line feed are sent. Frequently, there is one value of WIDTH that will entirely disable the automatic carriage return and line feed. If you should need to defeat auto new line, you should see the documentation for the language you are using.

## 10.6 Languages with Buffered Output

Buffered output to the console can also cause trouble with the video display functions. A language that performs buffered output to the console saves up all the characters that are being sent to the console until the end of a line is reached. Then, the entire line is sent to the console at once. Typically, the end of a line is not equated to the end of a program source code line; it is usually signalled by the sending of a carriage return and line feed to the console.

It is common for a program to move the cursor to a particular location on the display and wait for user input. To move the cursor, the program must send an escape sequence to the console. With languages that use buffered output, the escape sequence will go into a buffer, and it will not reach the console until the program sends out a carriage return and line feed. But the carriage return and line feed defeat the purpose of the escape sequence. The result is that the cursor is moved to the desired spot on the display and immediately returned to the beginning of the next line!

The problem persists even if the language in question has a mechanism for outputting characters without a carriage return and line feed (e.g. ending a PRINT statement with a semicolon in Basic). The characters are still held by the language until the end of a line, which generally consists of a carriage return and a line feed. Usually, there is no way to defeat this form of operation in a language. The only sure solutions involve going outside the language (for example, to a special assembly language routine), changing to another language without buffered output, or (hopefully) finding a newer version of the language that does not incur the problem.

## 10.7 Converting TRS Basic Programs

The utility program TRS2CPM allows programs that have been written under TRSDOS to be transferred over to P&T CP/M 2. (See Section 8.31 for full details on TRS2CPM.) Under CP/M, Microsoft Basic-80 is the most similar of all languages to TRSDOS Basic, but there are still a few conversions that must be made.

The most common conversion is the "PRINT@" capability of TRSDOS Basic. Standard Basic-80 does not support this function, but it may be emulated quite easily by a user-defined function. The technique is to define a function which has two arguments (one for row number and the other for column number) and which returns the appropriate 4-character escape sequence for P&T CP/M 2's console display. With this function defined, "PRINT@ (x,y)" becomes "PRINT FNAT\$(x,y)". Figure 10.5 shows a small Basic-80 program that defines the function and uses it. Note the use of the WIDTH command, mentioned above, to defeat the automatic carriage return and line feed. Also notice that row and column numbers are entered directly into the FNAT\$(x,y) function; the function itself adds 32 to each number and returns the resulting character codes.

```

100 WIDTH 255           'first inhibit auto CRLF on console output
110 '
120 ' now define the function to generate cursor addressing strings
130 '
140 DEF FNAT$(ROW,COL)=CHR$(27)+"Y"+CHR$(ROW+32)+CHR$(COL+32)
150 '
160 ' now make use of the function to print at specific locations
170 '
180 PRINT FNAT$(2,5);"This string begins at row 2, col 5."
190 PRINT FNAT$(20,20);"This string begins at row 20, col 20."
200 END

```

Figure 10.5 Definition and Sample Usage of FNAT\$ Function In Basic-80

## 10.8 Using Console Functions

Figure 10.7 offers a more complex utilization of the console display functions. This program, written in Basic-80, clears the display, draws a box using the graphics characters, and alternates between normal and reverse video as it repeatedly writes a message in the box. In Basic-80, control characters are sent to the console by printing the result of a CHR\$ function. Figure 10.6 shows the control characters that are used in Figure 10.7.

CHR\$(12)	is a clear screen and home cursor command (<ctrl-L>)
CHR\$(17)	is a begin graphics character (<ctrl-Q>)
CHR\$(27)	is the escape character
CHR\$(8)	is a backspace command (<ctrl-H>)
CHR\$(30)	is a move cursor up command
CHR\$(10)	is a move cursor down command (<ctrl-A>)
CHR\$(20)	is an end graphics command (<ctrl-T>)
CHR\$(14)	is a begin reverse video command (<ctrl-N>)
CHR\$(15)	is an end reverse video command (<ctrl-O>)
CHR\$(6)	is a home cursor command (<ctrl-F>)

Figure 10.6 Using CHR\$ In Basic-80 to Generate Control Characters

```
100 ! This program illustrates some of the display control functions
110 ! first it draws a box on the screen using the graphics characters
120 ! then it repeatedly writes a message in the box alternating
130 ! between normal and reverse video
140 !
150 WIDTH 255           'inhibit auto CRLF on console output
160 !
170 ! define the cursor addressing function
180 DEF FNAT$(ROW,COL)=CHR$(27)+"Y"+CHR$(ROW+32)+CHR$(COL+32)
190 !
200 PRINT CHR$(12);      'clear screen
210 !
220 ! put in the box
230 PRINT CHR$(17);      'switch to graphics
240 !
250 ! message will appear at row 10, column 30 so start drawing the box
260 ! at row 10, column 29 First move the cursor there
270 PRINT FNAT$(10,29);
280 !
290 ! the next statement puts in a vertical line, backspaces the cursor and
300 ! moves it up one line, then puts in the upper left corner of the box.
310 PRINT "4";CHR$(8);CHR$(30);"; "
320 !
330 FOR I=1 TO 19
340 PRINT "6";          'put in top line
350 NEXT
360 !
370 ! The next statement puts in the upper right corner, backspaces the
380 ! cursor, moves it down a line, puts in a vertical line, backspaces the
390 ! cursor, moves it down a line, then puts in the lower right corner
400 PRINT "!";CHR$(8);CHR$(10);"4";CHR$(8);CHR$(10);"B";
410 !
420 PRINT FNAT$(11,29); 'move to beginning of bottom line of box
430 !
440 PRINT "C";          'put in lower left corner of box
450 FOR I=1 TO 19
460 PRINT "6";          'put in bottom line
470 NEXT
480 ! box is finished now.
490 !
500 PRINT CHR$(20);      'go back to non-graphics
510 RV=15                'initialize to normal video
520 !
530 ! the next line moves to the beginning of the message, prints the
540 ! character given by RV to set for reverse/normal, prints the message,
550 ! returns to normal mode, and moves the cursor to the home position
560 PRINT FNAT$(10,30);CHR$(RV);;"This is the message";CHR$(15);CHR$(6);
570 !
580 IF RV=15 THEN RV=14 ELSE RV=15      'swap video mode
590 FOR I=1 TO 500:NEXT      'delay a bit so blink isn't too fast
600 GOTO 560
```

Figure 10.7 Example of Using the Display Control Functions

## 10.9 Special System Functions

P&T CP/M 2 uses several special system functions in order to provide additional video display features. (See Chapter 16 of this manual for full details.) These functions are accessible from most high level languages and supply the features listed in Figure 10.8.

function number	function performed
17	read XY position of cursor
18	read character at current cursor location
19	set size and blink of cursor
20	set cursor blink and on/off
21	enable direct access to screen memory
22	disable direct access to screen memory
23	set split screen mode
31	enable or disable terminal emulation
32	set up message to be flashed on screen
33	flash message if enough time has elapsed
34	restore screen to state before flashing message

Figure 10.8 Special System Functions for the Video Display

The "split screen" feature of the video display software allows you to set a number of lines at the top of the video display to be non-scrolling. In effect, this feature allows you to partition the display into two areas, a fixed area at the top of the display and a scrolling area at the bottom of the display. As information is displayed, the bottom portion will fill up and begin to scroll as usual, but the upper portion will remain in place. Although the information in the non-scrolling area remains on the display, it is not protected. You may address the cursor into that area and change the information in a normal manner. In addition, the clear screen code will clear both the non-scrolling and scrolling portions. Special System Function 23 invokes the "split screen" feature. See Chapter 16 of this manual for further details.

Special System Functions 32 through 34 may be used by a program to flash a message in the center of the screen and then restore the screen to its previous condition when the message is no longer needed. This can be particularly useful for reporting errors or alerting the user to various important conditions. Since the console display is fully restored after the message is removed, the program need not be concerned about the display being modified when such a message is given. Chapter 16 of this manual gives further information about the use of these functions. Figure 10.9 shows an example of the use of these functions in a Basic program.

```

10 'Example of using the Flashing Message Routines From Basic-80
20 '
30 SYSENT=&H43
40 ON ERROR GOTO 10000
.
.
.
10000 MSG$="Error # "+STR$(ERR)+" occurred on line "+STR$(ERL)+"
" - <esc> to quit, <enter> to continue"
10010 POKE &H47, 32
10025 I$=VARPTR(MSG$)
10027 POKE &H46, PEEK(I$)
10030 POKE &H4A, PEEK(I$+1)
10040 POKE &H4B, PEEK(I$+2)
10050 CALL SYSENT
10060 POKE &H47, 33 : CALL SYSENT : C$=INKEY$
10065 IF LEN(C$)=0 THEN 10060
10070 IF (ASC(C$)<>27) AND (ASC(C$)<>13) THEN 10060
10080 POKE &H47, 34 : CALL SYSENT
10090 IF ASC(C$)=27 THEN SYSTEM ELSE RESUME

```

Figure 10.9 Sample Basic-80 Program Using a Flashing Message

Line 30 of the program sets up a variable named SYSENT to be used when calling the special system functions. Line 40 tells Basic to transfer control to Line 10000 if

an error should occur. The routine starting on Line 10000 flashes the message "Error # n occurred on line nnn - <esc> to quit, <enter> to continue" in the center of the screen and waits for input from the console.

First, the entire message string is assembled in the string variable MSG\$ (Line 10000). Pseudo register B is loaded with the number of the special system function (32) which is used to set up the message to be flashed on Line 10010. Next the address of MSG\$ is stored in I% (Line 10020). Due to the way Basic-80 works, I% actually contains the address of a three byte "string descriptor" rather than the string itself. This "string descriptor" consists of one byte containing the number of characters in the string followed by two bytes containing the address of the string.

Line 10025 transfers the length of the string into pseudo register C. Lines 10030 and 10040 transfer the low and high bytes of the address of the string into pseudo registers L and H, respectively. Finally on Line 10050, the special system function is called to set up the message to be flashed. On Line 10060, Special System Function 33 is called to allow the message to be flashed and the console is checked for a character. This is repeated until a character arrives from the console. Line 10070 causes any characters except <esc> and <enter> to be ignored. If <esc> or <enter> is received, Line 10080 calls Special System Function 34 to restore the console display and line 10090 takes the appropriate action based on the character.

## 10.10 Line Wrap

Some programs expect that the cursor of the console display will advance automatically to the next line after a character is displayed in the last column. Other programs assume that the cursor will remain at the last position on the line. The video display software in P&T CP/M 2 allows you to specify either of these actions. If a program you are using is producing unexpected results at the ends or beginnings of lines, it may expect a different behavior than the system is currently providing. For programs that expect the cursor to move to the next line, the wrap mode of the video display should be enabled. For programs that expect the cursor to remain at the end of the line, the wrap mode should be disabled. The wrap mode can be set by the control codes <ctl-V> and <ctl-W>, the system MENU option CP, or by the SETMISC utility program. (See Section 8.25 for full details on the SETMISC utility.)

It might not be totally obvious from what appears on the console display that a problem is due to the display being set to non-wrap mode. When some programs want to fill the entire display, they position the cursor to the upper left-most corner of the display and then send 1919 characters to the console. Such programs assume that the console will wrap to the next line after filling any given line. If you run such a program when the display driver is in the non-wrap mode, the first 79 characters will be displayed properly on the first line of the display and the next 1840 will be displayed in the last column of the first line!! If you run a program and it seems to place information only on the first line of the console display when you expect it to fill the entire screen, it is very likely that you will need to place the display in the wrap mode.

It is very poor design for a program to assume that a console display device is always in the wrap (or non-wrap) mode. If you are writing a program, a much better programming technique is to allow for an initialization string that can be sent to the console to enable or disable wrapping as necessary. It is even better to avoid the entire problem by positioning the cursor to the beginning of each line before sending out the characters to that line. Note that on the last line of the

display you should send out only 79 characters (one less than the maximum) so that the display will not scroll up one line if the display is in the wrap mode.

### 10.11 Direct Display Access

The video display in the TRS-80 Model II/12/16 is accessed as if it were memory. In other words, all display characters simply reside for the time in a special area of memory. In order to provide maximum memory space for programs, the computer provides a means for alternating the video display memory with regular memory. The video display is made available only when it is accessed; at other times, normal memory takes its place. This action is commonly called "bank switching," because the video display memory bank is switched in and out of the accessible memory space.

P&T CP/M 2 takes full advantage of the video memory bank switching. Video memory, when it is enabled, resides in the 2 Kbyte block of memory beginning at location F800h. When it is disabled, that region of memory is used by a portion of the operating system. Since some programs need to access the video display memory directly, P&T CP/M 2 provides Special System Functions 22 and 23 for enabling and disabling access to the video memory. It is very important to use these special system functions when directly accessing the video display.

The operating system must keep track of the current state of the video memory so that it may take appropriate steps if it needs the memory that is alternated with the video memory. Special System Functions 22 and 23 allow the system to keep track of the state of the video memory even when another program is accessing it.

There are some cases where the system cannot take corrective action even if it knows the current state of the video memory. For this reason, **it is essential that no operating system functions be used while the video memory is enabled for direct access**. A program should enable the video memory, make any necessary access to it, and then disable access to the video memory before attempting to use any system function (such as console input or printer output).

Figure 10.10 shows the use of direct video memory access in a Basic-80 program. This program reads the entire video display into a large array and then sends it out to the system printer. Because this program is written for the Basic-80 interpreter, all access to the display must take place entirely within one line of the program. The Basic-80 interpreter checks the console status at the end of each line, which involves an operating system function, so the access to the display must be disabled by the end of each line.

```
10 ! Program to illustrate the use of direct access to the video memory
20 !
30 ! First set a constant to the location in memory to call to invoke the
40 ! special system functions - see the section on the special system
50 ! functions for full details
60 SYSENT$ = &H43
70 SPF$ = &H47      'location to poke special system function number
80 !
90 ! Dimension an array to receive the contents of the video memory
100 DIM SCR$(1919)
110 !
120 ! Now read the screen into the array - note this is all done on one line
130 ! of the program
140 POKE SPF$,21 : CALL SYSENT$ :
FOR I%=0 TO 1919 : SCR%(I%)=PEEK(&HF800+I%) : NEXT :
POKE SPF$,22 : CALL SYSENT$
150 !
160 ! Now print the contents of the array out to the printer
170 J% = 0
180 FOR I%=1 TO 24          'print 24 lines
190   FOR K%=1 TO 80         'print 80 characters on each line
200     LPRINT CHR$(SCR%(J%)) : J% = J% + 1
210   NEXT
220 LPRINT                  'follow by CR,LF
230 NEXT
```

Figure 10.10 Example of Using Direct Video Memory Access

## 10.12 SCRNDUMP Module

If you include the SCRNDUMP utility module in the system, it allows you to print the current contents of the console display by pressing <ctl->. All characters from the display will be printed on the system printer (the LST: device) except for graphics characters. Any graphics characters will be replaced by spaces so that the spacing on the printed copy will be the same as on the screen. Reverse video is ignored during the print process so characters that are in reverse video will be printed the same as normal characters.

**NOTES**

## 11.1 General Comments

The keyboard is the principle means of user input to the computer. It has keys for generating all the normal typing characters, some special keys, and a numeric key pad for easy entry of numbers. When the power is turned on, the keyboard is initialized to the normal lower case state. To produce upper case letters, or the upper case symbols shown on the key legends, the action is like that of a typewriter: either <shift> key is depressed before the key is struck.

There are two forms of shift lock on the keyboard. The <caps> key turns on and off a lock that causes all alphabetic letters to be shifted to upper case. All other keys are not shifted. When the red light in the <caps> key is on, this function is enabled. The caps shift is turned off by pressing the <caps> key again. The other shift lock has the same function as the one on most typewriters, as it causes all keys to be shifted. This lock is enabled by pressing the <lock> key on the keyboard; pressing either of the <shift> keys disables it. The <lock> key also has a red light which comes on when the function is enabled.

Any character on the keyboard can be repeated about 3 times per second by holding down its key and pressing the <repeat> key.

In addition to the normal alphabetic, numeric, and special characters, the full set of ASCII control characters can be generated. To generate a control character, use the <ctrl> key as you would the <shift> key: hold down the <ctrl> key and press the key corresponding to the desired control code. For example, to send a <ctrl-D> (numeric code = 4), hold down the <ctrl> and press the D key. A full table of the ASCII codes is given in Appendix A.

There are a few printing characters that are not shown on the keyboard but can be typed by using the control key. These are shown in Figure 11.1. Note that the accent grave cannot be generated from the TRS-80 Model II/12/16 keyboard. You can, however, use the KEYXLATE utility module described later in this chapter to translate one of the keys on the keyboard into accent grave.

<u>to get this</u>	<u>type this</u>
~ (tilde)	control 6
\ (back slash)	control 9
(logical or)	control 0
(rub out)	control -

Figure 11.1 Hidden Keyboard Characters

Most of the special keys on the keyboard generate special characters. Figure 11.2 summarizes the codes generated by these keys.

key	code (dec)	code (hex)	character
ESC	27	1B	ESC
TAB	9	09	HT, horizontal tab, <ctrl-I>
BACK SPACE	8	08	back space, <ctrl-H>
BREAK	3	03	ETX, <ctrl-C>
HOLD	19	13	DC3, X-OFF, <ctrl-S>
ENTER	13	0D	CR, carriage return, <ctrl-M>
left arrow	28	1C	FS
right arrow	29	1D	GS
up arrow	30	1E	RS
down arrow	31	1F	US
F1	1	01	SOH, <ctrl-A>
F2	2	02	STX, <ctrl-B>
F3 (Mod 12 only)	4	04	EOT, <ctrl-D>
F4 (Mod 12 only)	12	0C	FF, <ctrl-L>
F5 (Mod 12 only)	21	15	NAK, <ctrl-U>
F6 (Mod 12 only)	16	10	DLE, <ctrl-P>
F7 (Mod 12 only)	14	0E	SO, <ctrl-N>
F8 (Mod 12 only)	19	13	DC3, X-OFF <ctrl-S>

Figure 11.2 Codes Generated by Special Keys (black keys)

## 11.2 Direct Keyboard Input

Two types of console input are available with P&T CP/M 2. When a program uses the buffered input capability of the system, an entire line of console input is collected from the keyboard before it is returned to the program. The input is held in a buffer until a carriage return (<enter>) character is received, which causes the line to be returned. This type of input (used, for example, by the CCP in gathering command lines) allows some limited editing capabilities, as discussed in Section 5.17. Since some characters are used for the editing functions, they cannot be entered to a program while buffered input is being performed.

Direct console input is the second type. When a program uses direct console input, it receives all characters typed from the console, one at a time. In this case, the system merely acts as a conduit for the characters, so a program has much greater control over the actions taken in response to characters entered from the console.

Direct console input can be advantageous or necessary in many situations where you want to get characters from the keyboard without affecting the display. This allows your program to decide what changes are to be made on the display in response to each character. For example, suppose that a program presents a menu on the console and allows the user to move the cursor from one point to another with the cursor arrows. (One such program is the SETUP utility routine.) To perform this function it must gather one character at a time from the keyboard, checking each one for the codes which cursor arrows generate.

There are two types of direct console input available. The first type (BDOS Function 1) accepts a character from the keyboard and, if it is not a control character, a or is a carriage return, line feed, or backspace, will echo it back to the console output (a tab character will move the cursor to the next tab stop on the line). In other words, all characters except control characters are automatically displayed on the console as they are typed.

The second type of direct console input (BDOS Function 6) just returns the next character from the keyboard or a null character (00h) if there is no character available from the keyboard. This function does not echo anything back to the console output under any circumstances.

(If you need more details about using the BDOS functions from your programs, see Chapter 5 of the CP/M Operating System Manual.)

Direct console I/O is available in many languages (e.g. the INKEY\$ function in Microsoft Basic-80 and the CONCHAR% function in CBASIC). Some of them use BDOS Function 1 and some use BDOS Function 6. If you are getting unwanted echoing of the input characters on the console display, chances are that the language you are using uses BDOS Function 1 for direct console input. You should check the language documentation to see if there is a way of using BDOS Function 6 for the direct console input. In some cases, the only solution is to write an assembly language routine to call the BDOS console input functions (BDOS Function 6) directly.

### 11.3 Type-ahead

P&T CP/M 2 provides a 64-character type-ahead buffer for keyboard input. Characters can be accepted from the keyboard at any time (even during disk accesses), and they are held in the buffer until the program calls for them. This feature insures that no characters will be lost when a fast typist is entering information.

Typing a <ctl-X> on the keyboard has one of two functions. If there are characters waiting in the type-ahead buffer, they will be removed from the buffer. This action allows you to get rid of characters in the buffer that are no longer wanted as console input. If there are no characters waiting in the buffer, the <ctl-X> will be placed into the buffer and will be supplied as the next keyboard input character.

The type-ahead buffer also allows several commands to be entered at one time. (NOTE: A special command structure may be needed; see following paragraphs.) The computer then executes each command when it is ready. This function can be useful when the computer's response will be delayed. For example, an editor can be slow in returning to the system command level if a large file is being edited. While the editor finishes its "housekeeping", you can type in the command to execute another program, such as the assembler. As soon as the editor is finished, the CCP will read the command from the type-ahead buffer and act upon it.

There are certain problems relating to the type-ahead function in a CP/M system. Most common is the occasional loss of a character as normal console I/O is being performed. For example, you may have discovered that part of a command can be lost when you enter several commands at once. (This is a **very** common problem!) In this case, the culprit is BDOS.

BDOS periodically monitors console input and gets a character if one is available. It then determines whether or not the character is a <ctl-S> (generated by the <hold> key). If it is a <ctl-S>, BDOS stops console output until another character is received (including another <ctl-S>). If the character is not a <ctl-S>, BDOS saves the character until console input is requested. Until that time, BDOS will not check for another character from the console, since it can save only one. As a result, BDOS will not respond to any subsequent characters from the console as long as a character is being saved.

You may have discovered one effect of this situation while attempting a pause in console output from a program. Ordinarily, the <hold> key will effect a pause in program output, thus "holding" the display at any desired point. But if you should accidentally press any key other than <hold> or <ctl-S> before the pause is made,

there will be no further response to the <hold> key until the program performs some console input (to get the character being saved).

Any character stored in BDOS will usually go on to its rightful destination at the appropriate time. However, it will be lost whenever a warm boot is performed. Since a warm boot reloads the CCP and BDOS from the system diskette, all of the previous BDOS disappears, including any character it might have saved.

The net result is that if you type ahead at the end of a program (as suggested above), there is a good chance that the first character you type will be lost during the warm boot. Unfortunately, it is difficult to predict when this will occur. (It depends on the action of the program after you type the character.) But all is not lost. An inelegant but effective solution is to insert extra carriage returns (press the <enter> key) before each command to CP/M. In other words, press <enter> both before and after each command line when stacking several commands in the buffer. CP/M understands carriage returns only as markers for the ends of commands, and it treats any extra ones as ends of empty commands, duly ignoring them. Carriage returns, then, are ideal sacrificial characters. It makes no difference whether or not they are actually lost during a warm boot.

Additional considerations concerning the type-ahead buffer are listed below:

1. The type-ahead buffer cannot be used with certain languages (such as Microsoft BASIC-80). These languages check the console for a character on a periodic basis. Any character found is read and checked to determine if it is the abort character (often <ctl-C>). If it is the abort character, appropriate action is taken. If it is not, the language will calmly throw it away and read the next waiting character. In other words, the language does not keep the character, as BDOS does. One by one, all the characters waiting in the buffer are read and consequently lost. Since the programmer usually has very little control over this action, the type-ahead function (in this case) is virtually useless.
2. Some programs will gobble up one or two characters from the type-ahead buffer and do nothing with them. The characters, which were meant for another program or another part of the same program, are then lost from the buffer. Unfortunately, there is no system-level solution to this problem.
3. Some programs can take a considerable amount of time to act upon each character received. For example, many screen oriented text editors take a considerable amount of time when scrolling the console display backwards. If the scroll and repeat keys are used together (to repeat the scroll function), it is possible to have up to 64 characters piled up in the type-ahead buffer, each one ready to tell the editor to scroll backwards. Obviously, this backlog of characters can cause scrolling to continue past the desired point. If you find yourself in this situation, you can simply type <ctl-X> to clear the buffer and prevent unwanted action.

## 11.4 The <break> Trap

Several programming languages use <ctl-C> (the <break> key) to abort program execution. On the TRS-80 Model II/12/16 keyboard, it is easy to hit <break> when <back space> is intended, because the two keys are adjacent. At the very least, it is a nuisance for the program to be aborted by such a common slip of the finger. To avoid this problem, P&T CP/M 2 provides a <ctl-C> trap in the keyboard handler. When the trap is enabled, any <ctl-C> or <break> typed at the console will be replaced by a NULL.

The <ctl-C> trap is available regardless of the physical device which is assigned to the CON: logical device. It will perform just as well, then, if a terminal rather than the regular system keyboard is connected to a serial port. The <ctl-C> trap is enabled and disabled by Special System Function 26 (see Chapter 16). The trap is not affected by warm boots, so any program which enables it must disable it before returning to the operating system. Otherwise, it will not be possible to warm boot from the system command level.

## 11.5 AUTOKEY Module

If you include the AUTOKEY utility module in the system, it provides you with five programmable function keys. Each of the keys can be programmed to generate a string of up to 79 characters with a maximum of 128 characters for all strings. Any characters (including control characters) may be included in a programmed string. The keys may be programmed at any time directly from the console keyboard or by a program using an escape sequence. In addition, the string programmed into any of the keys can be viewed at any time without affecting the console display.

The five function keys are accessed by pressing <ctl-1>, <ctl-2>, <ctl-3>, <ctl-4>, and <ctl-5>. This allows the function keys to be supported without conflicting with any other control keys. For example, the <f1> key on the keyboard is identical to <ctl-A>. If <f1> were made into a programmable key, <ctl-A> would no longer be available (it would generate the same string as <f1>).

Pressing <ctl-7> followed by one of <ctl-1> through <ctl-5> followed by a string terminated with <enter> programs the string to be returned by a function key. Pressing <ctl-8> followed by one of <ctl-1> through <ctl-5> displays the current string programmed into that key on the bottom line of the console display. After displaying a string, the next key pressed will restore the console display to its previous condition.

See Section 7.2 of this manual for complete information about using the AUTOKEY module.

## 11.6 KEYXLATE Module

If you include the KEYXLATE utility module in the system, it allows you to translate up to 16 keys on the console keyboard to other characters. This lets you reassign keys for use with various programs. For example, some text editors do not allow you to use the cursor arrow keys on the keyboard. They insist, instead, on using control characters. With KEYXLATE you can translate the arrow keys to the control characters the program expects for cursor movements. The arrow keys will then function just like you would expect them to.

The translations performed by the KEYXLATE module may be defined by the KXEDIT utility program. They may also be defined from any program by sending an escape sequence to the console. See Section 7.3 of this manual for information on the KEYXLATE module and Section 8.18 for information on the KXEDIT program.

## NOTES

## 12.1 General Comments

In order to have access to the serial ports, a serial port module must be included with the system when it is loaded. For information about how to select the modules to be loaded with the system, see Section 6.3. There may be more than one serial port module available for you to choose from. The information presented here is for the standard serial port module, SIO. If there are other modules available, you may use the MODSEL program to display the descriptions of what they do. Appendix C also lists the modules available with a brief description of their functions.

P&T CP/M 2 provides very complete drivers for the RS-232 serial ports of the TRS-80 Model II/12/16. Previous sections have described the assignment of serial ports to logical I/O devices. This section will describe the capabilities of the built-in drivers in P&T CP/M 2, how to set them up, and how to access the serial ports at a more direct level in order to write special I/O routines.

Serial ports are most commonly used to connect printers to the computer and to provide communication with other computers. For your convenience, P&T CP/M 2 supplies the three most common types of serial printer protocol as standard features. The system does not include communications protocols because they are much more varied. However, all of the features needed to implement communications protocols are available via the special system functions so that programs can perform the necessary protocol operations on their own.

## 12.2 SIO Protocols

Printers that are connected to the serial ports generally utilize some sort of protocol to inform the computer when characters can and cannot be received. Many printers have internal buffers that hold incoming characters, but the buffers can fill up if characters arrive at a faster rate than they can be printed. When its buffer is full (or nearly full), a printer will inform the computer in some way that it should stop sending characters. If more characters were to arrive when the buffer is full, they would be lost. When the printer is ready to accept more characters, it will tell the computer to resume sending.

In P&T CP/M 2, the operating system itself responds to printer protocol. If a printer has indicated that it cannot accept characters and a program tries to send one, the system will wait until the printer is ready. In some cases (such as a printer malfunction) the printer may be unable to continue, thus causing the system to lock up. In order for the program to regain control, the printer must be made ready to accept characters. The value of the "xmit-on" character determines the type of protocol used for a serial port (see Section 8.27 on the SETUP program for instructions on changing the "xmit-on" character).

There are four types of protocols available for the serial ports with P&T CP/M 2. **The first is no protocol at all.** (xmit-on character = 0) In this case, the serial port software merely sends and receives characters without any interaction with or modification of the data being transferred. The serial ports should be set to no protocol when they are used with programs that take care of protocol considerations on their own.

**The second type is ETX/ACK protocol.** (xmit-on character = 1) This protocol is often used with older printers (particularly Diablos). It is the least desirable from an efficiency standpoint, so you should select another type if your printer allows you an option.

**The third type is the RS-232 status line protocol** (xmit-on character = 2) which is sometimes called reverse channel, hardware handshake, or clear-to-send protocol. When this type of protocol is used, the serial port will examine the Clear To Send (CTS) and Device Carrier Detect (DCD) lines coming in from the printer. Data will be sent out only when the CTS line is high, and it will be received only when the DCD line is high. If the CTS line goes low during transmission, the transmission will cease at the end of the current character and will resume only when the CTS line goes high again. If the DCD line is not high, the serial port will not recognize any characters coming in to the serial port. This is the same serial port protocol that is used by TRSDOS and is the preferred method if the printer will support it.

**The fourth type is XON/XOFF protocol.** (xmit-on character anything other than 0, 1, or 2) It is preferred over ETX/ACK protocol but is less efficient than the RS-232 status line protocol. With XON/XOFF, the printer sends a special character to the computer when it cannot accept more characters, and it sends another special character when it is ready again. In this manual, the special character that is sent to stop transmission is called the **Xmit Off** character; the one to restart transmission is called the **Xmit On** character. The most commonly used **Xmit Off** character is the ASCII control character XOFF (<ctl-S>, ASCII character 17). The ASCII control character XON (<ctl-Q>, ASCII character 19) most commonly serves as the **Xmit On** character. The default setting of the serial ports in P&T CP/M 2 provides the XON/XOFF protocol with the **Xmit On** character set to <ctl-Q> and the **Xmit Off** character set to <ctl-S>.

P&T CP/M 2 allows the serial ports to be set to recognize any 5-bit to 8-bit characters except NULL, <ctl-A>, and <ctl-B> as the **Xmit On** and **Xmit Off** characters. However, a problem can arise if all 8 bits are used. Any ASCII character is represented by only 7 bits, but many printers arbitrarily set the eighth bit to a zero or a one before sending it to the computer. If the eighth bit is set to a one, the numeric code sent will be greater by 128 than the correct code for the character. For example, the numeric code for <ctl-S> is 17 and for <ctl-Q> is 19. If the printer sets the high order bit of each character to one, it will actually send a code of 145 for <ctl-S> and 147 for <ctl-Q>. Unless these changes are made, the computer will not recognize the handshaking characters coming back from the printer and the printer's internal buffer may overflow.

The problem is avoided if successful communication can be established with the serial ports set to recognize characters of only 7 data bits. However, some printers require 8-bit characters. If you have such a printer and characters are being lost (while using XON/XOFF protocol), try adding 128 to the numeric value of each of the **Xmit On** and **Xmit Off** characters.

### 12.3 Serial Port Parameters

In addition to the type of protocol, you may specify the following parameters for each serial port: baud rate, parity, number of stop bits, word (character) length in bits, data terminal ready (DTR) and request to send (RTS) status outputs. Figure 12.1 shows the various values that may be selected for each of these parameters.

baud rate:	110, 134.4, 150, 300, 600, 1200, 4800, 9600
parity:	on, off, even, odd
stop bits:	1, 1.5, 2
word length:	5, 6, 7, 8 (bits)
DTR output:	high, low
RTS output:	high, low

Figure 12.1 Serial Port Parameter Options

The serial port parameters may be selected in a variety of ways. One is the SETUP utility, which provides a user interactive, screen oriented method for setting the serial port parameters and the system IOBYTE. See Section 8.27 of this manual for full details on using SETUP.

P&T CP/M 2 provides Special System Function 0 to set the serial port parameters under program control. Full details on this function are found in Chapter 16 of this manual.

## 12.4 Accessing the Ports

The serial ports are accessible through the standard CP/M I/O facilities. A serial port can be assigned to any of the CP/M logical devices (CON:, LST:, RDR:, PUN:) by changing the value of the IOBYTE (see Chapter 14). The IOBYTE may be changed directly by a program or from the console by running the SETUP or ASSIGN utility program. See Chapter 8 for information on the utility programs. If you assign a serial port to the CON: logical device, whatever is attached to the serial port immediately becomes the system console and has control of the system. This assignment would be made, for example, if you wanted to use a serial video terminal as a remote console to the computer.

You would assign a serial port to the LST: logical device in order to use a serial printer as the system printer. Typically, the port would be assigned to the RDR: or the PUN: if the attached device is not to be treated as the system console or printer. It might be assigned to the RDR: to link two computers together for file transfer via the PIP routine. Or, the assignment to RDR: or PUN: could be used to connect an alternate printer or other device. Both of these logical devices are accessible from some high level languages.

P&T CP/M 2 provides several special system functions which are not available through the normal CP/M I/O facilities. Some of these functions allow programs direct access to the serial ports, as shown in Figure 12.2.

function number	function performed
1	read a character from serial port A
2	read a character from serial port B
3	output a character to serial port A
4	output a character to serial port B
5	read serial port A status
6	read serial port B status

Figure 12.2 Special System Functions for Accessing the Serial Ports

Section 16.3 of this manual gives an example of using these functions from Basic to implement a simple terminal program.

## 12.5 Connecting to the Ports

Figure 12.3 shows the location of the pins on the serial port connectors on the back of the computer.

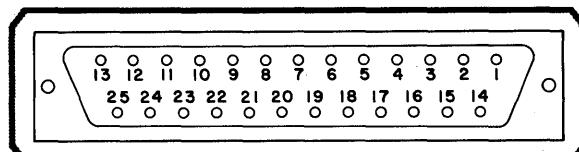


Figure 12.3 Pin Locations on the Serial Port Connectors

Figure 12.4 shows the functions of the serial port output pins used by P&T CP/M 2.

<u>pin</u>	<u>function</u>	
1	chassis ground	
2	transmitted data	(data output from computer)
3	received data	(data input to computer)
4	request to send	(status line output from computer)
5	clear to send	(status line input to computer)
7	signal ground	
8	carrier detect	(status line input to computer)
20	data terminal ready	(status line output from computer)

Figure 12.4 Definition of Pins on Serial Port Connectors

There is often some confusion about how to connect two RS-232 devices together. The RS-232 standard is defined to connect modems to terminals, so it is useful to think of any device as resembling either a modem or a terminal. The TRS-80 Model II/12/16 is set up as if it were a terminal. Connecting it to a modem, then, requires only a straight-through cable (one that connects each pin at the computer to the same pin at the modem). Connecting another device which is like a terminal (most printers are set up as terminal-type devices) requires a cable that crosses over certain lines.

Figures 12.5 and 12.6 show how a cable might be constructed to connect modem-like and terminal-like devices. These descriptions are very general and should be used only as examples. When connecting a serial device to the computer, you should consult its documentation to determine what sort of cable is actually required.

<u>Pin # at device</u>	<u>Pin # at computer</u>	<u>comments</u>
1	1	establish common ground between chassis
2	2	connect transmitted data from computer to received data of modem
3	3	connect transmitted data from modem to received data of computer
4	4	connect request to send output from computer to request to send input to modem (since some modems require this signal to operate)
7	7	establish a common signal ground
20	20	connect data terminal ready output from computer to data terminal ready input of modem (since some modems require this signal to operate)

Figure 12.5 Typical Cable for Connecting a Modem-like Device

<u>Pin # at device</u>	<u>Pin # at computer</u>	<u>comments</u>
1 <-----> 1		establish a common chassis ground
2 <-----> 3		connect transmitted data from terminal to received data of computer
3 <-----> 2		connect transmitted data from computer to received data of terminal
5 <-----> 4		connect clear to send input to terminal to request to send output of computer (since some terminals require a high level on clear to send to operate)
6 <-----> 4		connect data set ready input to terminal to request to send output of computer (since some terminals require a high level on data set ready to operate)
7 <-----> 7		establish a common signal ground
8 <-----> 20		connect carrier detect input of terminal to data terminal ready output of computer (since some terminals require a high level on carrier detect to operate)

Figure 12.6 Typical Cable for Connecting a Terminal-like Device

## NOTES

### 13.1 General Comments

Note that in order to have access to the parallel printer port, a parallel printer port module must be included with the system when it is loaded. For information about how to select the modules to be loaded with the system, see Section 6.3. There may be more than one parallel printer port module available for you to choose from. The information presented here is for the standard parallel port module, PPSTD. If there are other modules available, you may use the MODSEL program to display the descriptions of what they do. Appendix C also lists the modules available with a brief description of their functions.

P&T CP/M 2 provides access to the parallel printer port of the TRS-80 Model II/12/16 for printers that use a Centronics style parallel interface. To connect this type of printer, the ASSIGN or SETUP utility must connect the CENTRON physical device to either the CON: or LST: logical device. As already noted, CENTRON is the physical device name for the parallel printer port.

Refer to the Radio Shack documentation for the pin assignments on the parallel port connector. NOTE: In at least some of the Radio Shack manuals, the diagram of the connector is incorrect. As viewed from the back of the computer, the row of odd-numbered pins on the connector is the bottom row.

The parallel printer port output routine monitors the BUSY status line which determines whether a character can be sent out. If this line is high when a character is received, the routine waits until the line is low (signifying not busy) before sending the character over the data lines. When the character is sent, the routine also issues a strobe pulse over the strobe line.

### 13.2 Parallel Port Options

The printers that are most often connected to the parallel printer port are made by Radio Shack. Most of these printers are internally set to advance the paper automatically whenever a carriage return is received. However, almost all CP/M programs expect a printer that does not perform this function. These programs send out both a carriage return and a line feed at the end of each line. The result, unfortunately, is double spacing, because the printer advances once for the carriage return and once for the line feed.

The best solution for this problem requires a printer that does not generate automatic line feeds. Some of the newer Radios Shack printers have a switch or software option that allows you to defeat the automatic line feeds. See the documentation that came with your printer to determine if this option is available. If it is available, you should make use of it. This will result in the best performance.

If you do not have an option for defeating the automatic line feed or if you do not wish to use it, there are features incorporated into the P&T CP/M 2 software which can possibly provide an alternative. However, it is very difficult to provide general purpose software that can be adapted for all varieties of Radio Shack printers. In some cases, it may not be possible to make a printer perform exactly as desired through the software options alone.

The available options for the parallel printer port software are shown in Figure 13.1.

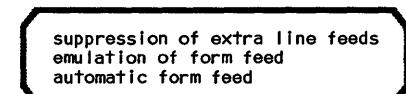


Figure 13.1 Options for the Centronics Parallel Port

If the suppression of extra line feeds option is enabled, the parallel port software attempts to remove the line feeds that cause double spacing. **SPECIAL NOTE:** Line feeds that do not follow a carriage return will not be removed.

If the "emulation of form feeds" option is enabled, the parallel port software will keep track of the current position of the paper. When it receives a form feed character, it will then send the appropriate number of line feeds to advance the paper to the top of the next page. This option is useful for printers that do not recognize a form feed character.

The location of the top of the page can be set in a variety of ways. When the system is first loaded (RESET), it assumes that the current paper position is at the top of the page. Secondly, if a <ctrl-T> is sent to the printer, the parallel port software interprets it as a command to set the current paper position to be the top of page. A third way to make this setting is through Special System Function 12, as described in Chapter 16 of this manual.

**NOTE:** If (and only if) the form feed emulation option is enabled, a <ctrl-T> character will be used by the parallel printer port software and not be sent out to the printer. If you need to send a <ctrl-T> (ASCII character 20) to a printer under these circumstances, send it with the high order bit set (i.e., send the ASCII character 148). Then it will not be recognized by the parallel port software and will be passed on to the printer.

If the "auto form feed" option is enabled, the parallel port software will insert a form feed into the output after a specified number of lines have been printed. This option can be useful in producing paginated listings from programs incapable of pagination.

Remember that all of the options available for the parallel printer port are options; all can be disabled. When they are disabled (ie. for a "normal" printer), all characters will simply be sent directly to the printer.

The parallel printer port options can be set by the utility routine SETMISC, system MENU option PP, or by programs which use the special system functions. For details on the SETMISC utility program, see Section 8.25 of this manual. The system MENU is described in Chapter 4. The special system functions are discussed in Chapter 8. Besides the ones which set the port options, there is a function which checks the parallel port status and one which sends characters directly to the port. All of the special system functions that deal with the parallel port are shown in Figure 13.2.

function number	function performed
7	read Centronics port status
8	send character to Centronics port
9	set Centronics port options
10	set page length
11	set number of lines per page
12	set top of page to current position

Figure 13.2 Special System Functions for the Centronics Port

Any changes in the parallel port parameters will remain in effect only until the next time the system is loaded (RESET) unless they are made permanent by system MENU option FR (see Chapter 4).

### 13.3 Special Notes

Some Radio Shack printers can perform a reverse line feed to move the paper backwards. This reverse motion is typically invoked by the two-character sequence ESCAPE, LINE FEED. A minor problem arises if the form feed emulation in the parallel port driver is enabled, as it will interpret the second part of the sequence as a normal line feed. Thus, if you are using reverse line feeds, the form feed emulation logic will be ineffective. In this case, you should disable form feed emulation and have your program should keep track of the current position on the page and issue multiple line feeds when necessary for page advances.

### 13.4 Error Messages

The parallel printer port software monitors the status signals sent by the printer and, if an error condition is detected, it will display an appropriate message on the console display. These messages will flash in the center of the display screen. After you take action to correct or bypass the error, the display will be restored to its former condition so that there is no effect on a program that is running.

Parallel printer port errors are detected only when a program attempts to send a character to the port. When you receive a parallel printer error message, you have three options available to you.

1. You can press the <break> key. Doing this will cause the system to immediately return to the command level by performing a warm boot. In this case any program that is executing will be terminated.
2. You can correct the error condition and press the <enter> key. When you press <enter> the system will again attempt to send the character out to the parallel printer.
3. You can press the <esc> key followed by the <F1> key. This action will cause the parallel printer port to be disabled until the next system warm boot (ie. when the system next returns to the command level). This option is useful when an error condition occurs that cannot be corrected immediately (for example the printer is not connected to the computer). If such an error occurs while a program is running and you want the program to run to completion and you are willing to sacrifice the printed output, you can press <esc><F1>. You will lose the printed output but all the other functions of the program will

perform as normal. When the system returns to the command level, the parallel printer port will be re-enabled.

The parallel printer port will be re-enabled only if the program terminates by a warm boot. Some programs (like STAT) return directly to the CCP without actually performing a warm boot. In addition, the built-in functions of the CCP (DIR, ERA, etc.) do not end with a warm boot either. In these cases, the printer port will not be re-enabled unless you force a warm boot by pressing the <break> key. In addition some menu systems do not ever allow control to pass to the warm boot routine in BIOS. With some of these systems, the parallel printer port may not be re-enabled after you press <esc><F1> until you exit the menu system.

### Possible Error Messages

#### **Parallel Printer Not Selected**

This message indicates that the select line of the printer is not active. This is usually caused by the printer being in the off-line or local mode. Check the printer and, after correcting the condition, press <enter>. This message may also be caused by the printer not being connected to the computer. If you cannot correct the problem, you may press <esc><F1> to suspend printing yet allow the program to continue.

#### **Parallel Printer Out of Paper**

This message indicates that the printer is reporting an out of paper condition. Check the printer and, after correcting the condition, press <enter>. If you cannot correct the problem, you may press <esc><F1> to suspend printing yet allow the program to continue.

#### **Parallel Printer Fault**

This message indicates that the printer is reporting a fault condition. The causes of this vary from printer to printer. Check the documentation for your printer to see if the condition can be remedied. If you can remove the problem, do so and press <enter> to continue printing. If you cannot correct the problem, you may press <esc><F1> to suspend printing yet allow the program to continue.

## 14.1 IOBYTE

As already noted, P&T CP/M 2 provides the ability to assign physical I/O devices to logical I/O devices. Assignments for all the logical I/O devices in the system are controlled by the byte stored in location 3 of memory, which is known as the IOBYTE.

The IOBYTE is divided into four fields, each having two bits. The value of each two-bit field determines the physical device assignment for one logical device. Bits 0,1 determine the console assignment, 2,3 the reader assignment, 4,5 the punch assignment, and 6,7 the list device assignment. Figure 14.1 gives a schematic view of this layout.

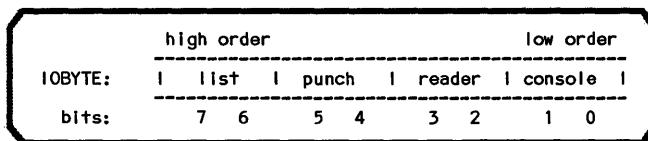


Figure 14.1 IOBYTE Bit Assignments

The correlation between the value of each bit and the physical device assigned is shown in Figure 14.2.

CONSOLE (bits 0 and 1)			
binary	decimal	assignment	
00	0	input and output is done to serial port A	
01	1	input comes from the system keyboard, output goes to the CRT	
10	2	input comes from the system keyboard output goes to the parallel printer port	
11	3	input and output is done to serial port B	

READER (bits 2 and 3)			
binary	decimal	assignment	
00	0	input is from serial port A	
01	1	input is from user defined device	
10	2	input is from user defined device	
11	3	input is from serial port B	

PUNCH (bits 4 and 5)			
binary	decimal	assignment	
00	0	output is to serial port A	
01	1	output is to user defined device	
10	2	output is to user defined device	
11	3	output is to serial port B	

LIST (bits 6 and 7)			
binary	decimal	assignment	
00	0	output is to serial port A	
01	1	output is to CRT	
10	2	output is to parallel printer port	
11	3	output is to serial port B	

Figure 14.2 IOBYTE Physical Device Assignments

The IOBYTE is commonly used to setup the I/O assignments before executing a program. In trying out a program, for example, it may be desirable to have its LST: output sent to the video display instead of a printer or other hardcopy device. In this case, you would simply set bits 7 and 6 of the IOBYTE to 01b. When the program is run, all output that is sent to the LST: logical device will go to the

video display. When you are ready for hardcopy on your printer, you would modify the same bits to the appropriate value. Assuming that your printer is connected to the parallel printer port, you would change bits 7 and 6 to 10b.

There are three ways to modify the IOBYTE from the system console. The easiest, by far, makes use of the ASSIGN and SETUP special utility programs which were written especially for P&T CP/M 2. These programs allow you to set the IOBYTE interactively from the console. In addition, ASSIGN allows you to set the various I/O devices on the system. The alternative is to use the STAT utility routine. Modification of I/O assignments through STAT is less straightforward, since the routine was written for generalized CP/M. ASSIGN, SETUP, and STAT are covered in more detail in Sections 8.5, 8.27, and 8.28, respectively, of this manual.

NOTE: The interactive mode of ASSIGN and SETUP involve many of the P&T CP/M 2 video display functions, so it can be used only when the console (CON: logical device) is assigned to the CRT. If the CON: is assigned to some other device (e.g. a terminal connected to a serial port), you should make changes in the I/O assignment using the command line mode of ASSIGN or STAT only.

It is also possible for programs to reassign the IOBYTE "on the fly." There are two ways to do this. One is through special system calls that return the IOBYTE and allow it to be changed. (See Chapter 5 of the CP/M Operating System Manual for details.) This method is recommended if the language in use allows access to these calls. Figure 14.3 shows an assembly language program that accesses the IOBYTE through BDOS functions.

```
; Program to illustrate manipulation of the IOBYTE using
; the standard BDOS calls

0100      org    100h

0100 112B01  lxi   d,conmsg ;send a message to the console
0103 0E09    mvi   c,9   ;BDOS function for string print
0105 CD0500  call   5    ;call BDOS entry point

0108 0E07    mvi   c,7   ;BDOS function for getting IOBYTE
010A CD0500  call   5    ;call BDOS entry point
010D 322A01  sta    old   ;save old value of iobyte
0110 F603    ori    3    ;set low 2 bits to 1
0112 5F      mov    e,a   ;prepare to set IOBYTE
0113 0E08    mvi   c,8   ;BDOS function for setting IOBYTE
0115 CD0500  call   5    ;call BDOS entry point

0118 115701  lxi   d,prnmsg ;send a message to the printer
011B 0E09    mvi   c,9   ;note: using BDOS function to send
                           ; a string to the CON: device
011D CD0500  call   5    ;call BDOS entry point

0120 3A2A01  lda    old   ;restore original IOBYTE
0123 5F      mov    e,a   ;BDOS function for setting IOBYTE
0124 0E08    mvi   c,8   ;call BDOS entry point
0126 CD0500  call   5    ;call BDOS entry point

0129 C9      ret    ;return to system command level

012A 00      old:   db    0    ;storage for original IOBYTE
012B 5468697320conmsg: db    'This will be printed on the video display',13,10,'$'
0157 5468697320prnmsg: db    'This will be printed on the printer',13,10,'$'

017D      end
```

Figure 14.3 Using BDOS Functions to Access the IOBYTE

When a language does not allow direct access to system (BDOS) calls, programs may manipulate the IOBYTE directly. When this is done, it is important for the IOBYTE to be left in a reasonable state before a return to the system. For example, suppose that a program leaves the the console linked to serial port B, but the device on serial port B is an output only device. Since there is no way to enter commands to the CCP, the system will lock up. The example given in Figure 14.4 shows a BASIC program which uses both the CRT and serial port B (where it assumes a printer is connected) while doing only console output.

```
100 ' Program to demonstrate switching the IOBYTE under program control
110 '
120 ' First get the existing value of the IOBYTE and save it
130   A = PEEK(3)
140 '
150 ' Assume the IOBYTE starts with the console linked to the CRT
160   PRINT "THIS MESSAGE WILL APPEAR AT THE CONSOLE"
170 '
180 ' Now link the console logical device to serial port B
190 '
200   B = (A AND &HFC) OR 3
210   POKE 3,B           ' setup new value of IOBYTE
220 '
230 ' Now send a message to the printer
240   PRINT "THIS MESSAGE WILL APPEAR AT THE PRINTER"
250   PRINT
260 '
270 ' Note: The second PRINT is necessary with some printers to force the
280   previous line to be printed
290 '
300 ' Now set the IOBYTE back to normal and stop
310   POKE 3,A
320   STOP
```

Figure 14.4 Accessing the IOBYTE from Basic-80

## NOTES

## 15.1 General Comments

This section of the manual is supplied for advanced programmers who need to use the system I/O routines directly. This information is unimportant to many users, who may skip it without hampering their understanding of how to use the system.

The section of CP/M which communicates directly with the hardware of the system is called the Basic Input Output Subsystem (BIOS). Under normal circumstances, you will not need to know the details of its operation. P&T CP/M 2 is delivered with a BIOS that fully utilizes the standard I/O capabilities of the TRS-80 Model II/12/16. However, there are some programs which access the BIOS directly (without benefit of BDOS), so this section will describe its entry points and a few other relevant considerations.

Chapter 6 of the CP/M Operating System Manual helps the systems programmer adapt CP/M 2 to some specific hardware. Most of this chapter is irrelevant to P&T CP/M 2, since it has already been adapted to the TRS-80 Model II/12/16. For example, the programs SYSGEN and MOVCPM are not necessary for moving P&T CP/M 2. However, the following sections of that manual do contain useful information about the system:

- Sec. 6.6 "The BIOS Entry Points" (p.137); contains additional information about the CP/M 2 standard entry points to BIOS.
- Sec. 6.9 "Reserved Locations in Page Zero" (p.144); describes the usage of the system parameter area in the first page (0 - 100h) of memory.
- Sec. 6.10 "Disk Parameter Tables" (p.145); covers the details of the parameter tables which determine the characteristics of the disk drives in the system. There is a standard system call which returns the address of a disk parameter table, and this material may help you in interpreting the table. However, the system call is not commonly used, and this information can certainly be ignored by a beginning user.

## 15.2 Register Preservation

All of the BIOS routines supplied as part of P&T CP/M 2 preserve the Z-80 alternate register set and the Z-80 index registers. Neither the CCP nor BDOS use these registers, so a program may assume that they will be unaltered during any system function. But there is no such guarantee about the primary register set. In general, a program should assume that all of the primary registers will be modified by any system function, whether in BDOS or BIOS, and should take steps to preserve any registers that must not be modified.

The traditional way to preserve registers is by pushing them onto the stack before calling a system routine and popping them off of the stack afterwards. In P&T CP/M 2, an alternate approach may be used. If a program needs to preserve the contents of any of registers B through L, it can merely exchange the register set before and after the system call. Figure 15.1 gives an example of this technique for calling a system function which returns no results in registers B through L.

```
• ;normal program steps
EXX ;swap in alternate registers
MVI B,13 ;set up for special system function 13
CALL 40h ;call it
EXX ;restore original registers
• ;normal program resumes here
```

Figure 15.1 Using EXX to Preserve Registers During A System Call

### 15.3 BIOS Jump Table

The BIOS is accessed by means of a jump table at its very beginning. The form of this table is shown in Figure 15.2. The numbers in the first column of the figure give the offset from the beginning of BIOS in bytes as a hex number.

00	JMP	BOOT	; cold boot entry point to the system - perform ; any needed initialization
03	JMP	WBOOT	; ->jump at location 0 points to the next instruction ; warm boot entry point, read back in CCP & BDOS ; from the system area on the disk on drive A
06	JMP	CONST	; return the console (CON:) input status, returns OFFH ; in register A if a character is ready at the ; the console and 00H if no character is ready.
09	JMP	CONIN	; read the next character from the console (CON:) and ; return it in register A
0C	JMP	CONOUT	; send the character in register C to the console (CON:)
0F	JMP	LIST	; send the character in register C to the list device (LST:)
12	JMP	PUNCH	; send the byte in register C to the punch device (PUN:)
15	JMP	READER	; read the next byte from the reader (RDR:) and return it ; in register A
18	JMP	HOME	; set the next track to access on the disk = track 0
1B	JMP	SELDSK	; select the disk drive given by register C for ; further operations, determine the density of the ; disk on the selected drive, and return the ; address of disk parameter header for the drive.
1E	JMP	SETTRK	; set the next track number to access to the value in BC
21	JMP	SETSEC	; set the next sector to access to be the number in BC
24	JMP	SETDMA	; set the location in memory to get data from or put data ; to on the next disk access to the address given in BC
27	JMP	READ	; read the selected sector from the selected ; track on the selected drive to the memory ; address specified by the last call to SETDMA
2A	JMP	WRITE	; write one sector from the memory address specified ; by the last call to SETDMA to the selected sector ; of the selected track on the selected drive
2D	JMP	LISTST	; returns the status of the list device, returns OFFh in ; A if the list device is ready to accept a character ; and 00H if not ready to accept a character
30	JMP	SECTRAN	; performs logical to physical sector number mapping - ; used at single density to improve system response

Figure 15.2 P&amp;T CP/M 2 BIOS Jump Table

Each jump points to a specific BIOS routine. The very first instruction of BIOS is a jump to the entry point for a system load (cold boot). This entry is used only when the system is turned on or RESET is pressed. The second instruction is a jump to the warm boot routine. There is a jump to this instruction at location 0 of memory, which provides a way to find the beginning of BIOS. In particular, location 1 of memory contains the low byte of the address for the warm start entry to BIOS; location 2 contains the high byte. These bytes may be inspected to determine the BIOS address when it is needed. (Note: BIOS always starts on a page boundary, so the address of its very beginning will have a low byte of 00h. Consequently, the address for the warm start entry to BIOS will have a low byte value of 03h, since there are 3 bytes to each entry in the jump table.)

Under normal system operation, you should never need to access BIOS directly. To do so is to play with fire, and it becomes your responsibility to avoid being burned. You should be sure that you know what you are doing before trying to access the BIOS directly.

## 15.4 System Parameter Area

The system parameter area, commonly called the base page of memory, occupies the first 256 bytes of memory (0-FFh). It is reserved for system usage, and it contains buffers and system entry points. Figure 15.3 shows a map of this area under P&T CP/M 2.

address	usage
00h-02h	Contains a jump to the warm boot entry point of the BIOS.
03h	Storage for the IOBYTE which controls the logical to physical I/O device assignments.
04h	Storage for the current user number and the current default drive number. User number is stored in the high 4 bits and the drive number is stored in the low 4 bits. (A>0, B>1, ..., P>15)
05h-07h	Contains a jump to standard BDOS entry point. Programs should make a call to location 5 when using standard BDOS functions. Also, the address contained in locations 6h and 7h is the beginning of the resident portion of P&T CP/M 2. Programs may use all memory from 100h up to but not including this address.
08h-2Fh	Not currently used.
30h-37h	Not currently used, reserved.
38h-3Ah	Used by DDT and SID for breakpoint processing. Otherwise not used by P&T CP/M 2.
3Bh-3Fh	Not currently used, reserved.
40h-42h	Standard entry point for P&T CP/M 2 special system functions.
43h-45h	Entry point for P&T CP/M 2 special system functions with register setup.
46h-4Ch	Used by P&T CP/M 2 special system functions for register setup and parameter passing.
4Dh-4Fh	Not currently used.
50h-5Bh	Not currently used, reserved.
5Ch-7Fh	Space for default file control block.
80h-FFh	Default buffer for disk I/O. Also used to pass command lines from the CCP to transient programs.

Figure 15.3 Memory Map of the System Parameter Area (0h - FFh)

## 15.5 Interrupts

P&T CP/M 2 includes a system real time clock which operates on an interrupt basis. This means that once every 10 msec whatever is happening is interrupted for a brief period to take care of timekeeping functions. Several system functions such as keyboard input, diskette drive deselection, and the system time of day are tied to the real time clock. No program should ever disable interrupts for more than a few milliseconds since doing so will interfere with these system functions.

In addition, a program should not try to set up its own interrupt handlers. Doing so will almost certainly cause the system to crash. If you are writing a program that requires interrupt operation of an I/O device, contact Pickles & Trout for information about how to add an interrupt driver.

## 16.1 Introduction

P&T CP/M 2 provides numerous facilities that are not available as standard CP/M functions. In order to provide program access to these facilities in a convenient manner, P&T CP/M 2 implements **special system functions**. These functions are used in much the same manner as the standard BDOS functions as described in Chapter 5 of the CP/M Operating System Manual.

These special system functions can easily be accessed from assembly language programs and from most high level languages. In the explanation that follows, examples are given of using the special system functions from assembler and several high level languages.

To use the special system functions, a program must load the various machine registers with values that pertain to the function and then perform a special entry point to the system. Although this may sound complicated, it is really quite straight forward from most languages since, unlike the standard BDOS functions, Pickles & Trout provides a mechanism to make it easy to call the special system functions from a high level language.

Technical note for assembly language programmers: the special system functions preserve the contents of the Z-80 alternate register set and index registers (just like the P&T CP/M 2 BIOS does). The primary register set will not in general be preserved by the special system functions; hence programs using the special system functions are responsible for saving the contents of any of the primary registers that may be needed later before calling the special system functions. You usually need not worry about this when using special system functions from high level languages since most languages automatically preserve the contents of the needed registers.

## 16.2 Using Special System Functions from Assembly Language

As described above, all an assembly language program need do to use a special system function is load the Z-80 registers and perform a CALL to a special system entry point. This technique can be used by any language that can load the Z-80 registers and perform a CALL. The steps that the program must take are shown in Figure 16.1.

1. Load register B with the desired function number.
2. Load the other registers as required by the selected function.
3. Perform a call to location 40h.

Figure 16.1 Steps for Calling a Special System Function from Assembler

Note that the special system function entry point is at address 40h. When control returns to the calling program, the specified function will have been performed and any returned information will be in the registers as indicated in the descriptions of the individual functions.

Figure 16.2 shows the PRN listing of an example assembly language program using the special system functions. The program is a very simple-minded terminal program. It assumes that serial communication is going to be performed through

serial port B.

First the program sets up serial port B for 300 baud, no parity, 8 bit characters, 1 stop bit, and RTS and DTR both high. It then goes into a loop where it continually checks the console keyboard and serial port B status. If a character comes in from the serial port, it is displayed on the console screen and any characters that are typed on the console keyboard are sent out serial port B. This continues until <ctrl-C> or <break> is pressed on the console keyboard at which time the program returns to the operating system. For further information on using the BDOS functions (#6 is used here to perform console I/O) refer to Chapter 5 of the CP/M Operating System Manual.

```
;      Program to demonstrate use of Special System Functions to
;      set up and access the serial ports

0000 =      setup    equ     0      ;function for port setup
0006 =      status   equ     6      ;function for read port B status
0002 =      input    equ     2      ;function for read character from port B
0004 =      output   equ     4      ;function for write character to port B
0005 =      bdos    equ     5      ;entry point to system for BDOS functions
0040 =      ssf     equ     40h   ;entry point for special system functions

0100          org     100h

0100 0600      mvi    b,setup    ;function number for setup
0102 0E80      mvi    c,80h    ;setup serial port B for no parity
0104 1103E6      lxi    d,0e603h ;DTR=high, 8 bit words, 1 stop bit
0107 210000      lxi    h,0      ;RTS=high, 300 baud
010A CD4000      call   ssf     ;disable all status checking
                                ;call the special system function

010D 0606      loop:   mvi    b,status   ;read port B status
010F CD4000      call   ssf
0112 E601      ani    1      ;check for input character ready
0114 CA2201      jz     none    ;skip the following if no character

0117 0602      mvi    b,input   ;read the character waiting at port B
0119 CD4000      call   ssf
011C 5F      mov    e,a      ;move the character to E for output to con
011D 0E06      mvi    c,6      ;output using BDOS function 6
011F CD0500      call   bdos

0122 0E06      none:   mvi    c,6      ;check for console input
0124 1EFF      mvi    e,0ffh
0126 CD0500      call   bdos
0129 B7      ora    a      ;set flags and check for 0 (no character)
012A CA0D01      jz     loop    ;if no character then loop back for more
012D FEO3      cpi    3      ;check for ctrl-C
012F CA0000      jz     0      ;exit program if ctrl-C typed from console
0132 4F      mov    c,a      ;otherwise output the character to
0133 0604      mvi    b,output ; serial port B
0135 CD4000      call   ssf
0138 C30D01      jmp    loop    ;then loop for more

013B          end
```

Figure 16.2 Example of Using Special System Functions from Assembler

### 16.3 Using Special System Functions from High Level Languages

When using the special system functions from a high level language, there are two techniques which may be used. If the language can set up the Z-80 registers and CALL 40h (64 decimal) then the same approach as outlined for assembly language is the most efficient. Unfortunately few high level languages can do this directly. In most cases, it is necessary to have a small assembly language interface between the

high level language and the special system function call. The difficulty of writing this interface depends on the language being used.

### Pascal MT+ Example (direct call)

One example of using the special system functions from a high level language is shown in Figure 16.3. This program performs the same functions as the one in Figure 16.2 but is written in Pascal/MT+. Note that the INLINE assembly feature of Pascal/MT+ is used to provide the necessary interface to the special system function entry point. Each special system function is accessed by a separate function or procedure.

```
program sterm;
(*      A simple terminal program to illustrate the use of the special
      system functions from Pascal/MT+. *)
const   bdos=5;          (*entry point for standard BDOS functions*)
        ssf=64;           (*entry point for special system functions*)
var     ch, temp:char;
function conchar : char;
begin
  inline ("MVI C / 6 /  "MVI E / $FF /  "CALL / BDOS /  "STA / TEMP);
  conchar:=temp;
end;

procedure conout (cc : char);
begin
  inline ("MVI C / 6 /  "LDA / CC /  "MOV E,A /  "CALL / BDOS);
end;

function status : boolean;
begin
  inline ("MVI B / 6 /  "CALL / SSF /  "STA / TEMP);
  status:=tstbit(temp,0);
end;

function inpt : char;
begin
  inline ("MVI B / 2 /  "CALL / SSF /  "STA / TEMP);
  input:=temp;
end;

procedure oput (cc:char);
begin
  inline("MVI B / 4 /  "LDA / CC /  "MOV C,A /  "CALL / SSF);
end;

procedure setup (bc, de, hl : integer);
begin
  inline ("LHLD / BC /  "MVI B / 0 /  "MOV C,L /
          "LHLD / DE /  "MOV D,H /  "MOV E,L /
          "LHLD / HL /  "CALL / SSF);
end;

begin      (*main program*)
  setup ($80, $e603, $0);
  repeat
    if status then conout(inpt);
    ch:=conchar;
    if (ord(ch)<>0) and (ord(ch)<>3) then oput(ch);
  until ord(ch)=3;
end.
```

Figure 16.3 Example of Using Special System Functions from Pascal/MT+

### Special High Level Language Interface

In many high level languages it is cumbersome or impossible to make use of assembly language routines to interface to the special system functions. P&T CP/M 2 provides a means to access the special system functions from these languages if the language has a means of storing and retrieving information from specific locations in memory and has the ability to CALL a specific location in memory. This technique is somewhat less efficient than the method previously described but is easier to use in many cases.

Seven bytes of memory (starting at 46h) are set aside for passing the parameters that are normally passed in the Z-80 registers. A program merely needs to deposit the parameters in the appropriate location in memory and make a CALL to address 43h. When the special system function returns to the calling program, any returned parameters will be in the seven byte parameter area. Figure 16.4 shows the location and organization of the special system function parameter passing area.

address hex	address decimal	register
46h	70	C
47h	71	B
48h	72	E
49h	73	D
4Ah	74	L
4Bh	75	H
4Ch	76	A

Figure 16.4 Special System Function Parameter Passing Area

Note that the contents of all seven of the parameter bytes will be modified by the special system function making it necessary for the calling program to deposit values in the parameter bytes before each special system function call. It is necessary to load only those memory locations which correspond to parameters needed for the function being called (for most functions only one or two bytes are needed).

The steps required to use this alternate type of call to the special system functions are described in Figure 16.5.

- 1. Load the parameter area with the parameters necessary to the function.
- 2. Perform a CALL to address 43h.
- 3. Retrieve any returned results from the parameter area.

Figure 16.5 Steps in Calling Special System Functions Using the Parameter Passing Area

### Basic-80 Example (special interface)

Figure 16.6 shows the simple terminal program already presented in this section modified to run with the Basic-80 interpreter. It makes use of the alternate form of calling the special system functions.

```

10 ! Simple terminal program to illustrate calling the special system functions
20 ! using the parameter passing area in memory from the Basic/80 interpreter.
30 !
40 SYSENT%=&H43           'entry point for special system functions
50 !
60 ! first set up serial port parameters
70 POKE &H47,&H80 : POKE &H47,0      'setup registers B & C
80 POKE &H48,&H13 : POKE &H49,&HE6    'setup registers D & E
90 POKE &H4A,0 : POKE &H4B,0      'setup registers H & L
100 CALL SYSENT$          'call special system function
110 !
120 ! now check serial port status
130 POKE &H47,6             'set up reg B for function 6
140 CALL SYSENT$          'set up reg B for function 2
150 IF (PEEK(&H4C) AND 1)=0 GOTO 230
160 !
170 ! read character from serial port
180 POKE &H47,2             'set up reg B for function 2
190 CALL SYSENT$          'send it to console
200 PRINT CHR$(PEEK(&H4C));;
210 !
220 ! now check for a character from the keyboard
230 CH$=INKEY$              'loop if no character
240 IF LEN(CH$)=0 GOTO 130
250 IF CH$=CHR$(3) THEN STOP 'stop if ctrl-C
260 POKE &H46,ASC(CH$) : POKE &H47,4    'set up to output character
270 CALL SYSENT$          'call special system function
280 GOTO 130

```

Figure 16.6 Example of Using Parameter Passing Area from Basic/80 Interpreter

The technique of calling the special system functions shown in Figure 16.6 may also be used from compiled Basic/80 with a couple of minor modifications. With the compiler, the variable name referred to in the CALL statement must be a global variable. It is impossible to generate a global variable from within a Basic program so a very short assembly language "program" whose sole purpose is to create a global variable having the value 43h must be written. Line 40 of the program shown in Figure 16.6 must be removed since the variable it creates will now be created by the assembly language "program". Such a "program" is shown in Figure 16.7.

```

public sysent
sysent equ 43h
end

```

Figure 16.7 "Program" to Create a Global Variable with the Value 43h

Note that it is necessary for this "program" to be linked with the compiled Basic program that calls the special system functions. Assuming that the Basic program is in a file named STERM.REL and the assembled version of the "program" shown in Figure 16.7 is in a file named SYENT.REL, the command line to the linker would be as shown in Figure 16.8. If the "program" shown in Figure 16.7 is added to the Basic library, it is not necessary to explicitly name it when linking programs which use the special system functions. (See the documentation for the Basic/80 compiler for information about adding routines to the Basic library.)

```
L80 STERM,SYENT,STERM/N/E
```

Figure 16.8 Command Line for Linking a Compiled Basic/80 Program Using Parameter Passing Ar

### dBASE II Example (special interface)

The special system functions can be accessed from just about any language. The following example shows how to use Special System Functions 15 and 24 to access the system date and time from a dBASE II command file. This can be particularly useful since dBASE II stores in each data file the date of the last change made to it.

Normally dBASE II prompts the user for the date when it begins execution. However, by installing the following code in the dBASE II command file and executing it directly (ie. with a command line like "DBASE CMDFILE") the command file will directly read the system date and will not bother the user with a prompt for the date. Although dBASE II does not inherently use time of day, you may want to make use of it for a particular application. For example, you might be using dBASE II to store information about sales and it might be important to know at what time of day each sale took place. Using the techniques shown in this example, the command file can directly access the system time and, thus, will not require the user to enter it for each transaction.

```

1: store ' ' to dummy
2: set call to 67
3: poke 71,15
4: call dummy
5: store peek(73) to sec
6: store peek(74) to min
7: store peek(75) to hr
8: poke 71,24
9: call dummy
10: store peek(72) to day
11: store peek(73) to wday
12: store peek(74) to yr
13: store peek(75) to mon
14: store str(mon,2) + str(day,3) + str(yr,3) to sdate
15: set date to &sdate
16: ? "Today's date is ", date()
17: store str(hr,2) + ":" + str(min,2) + ":" + str(sec,2) to st
18: ? "The time is ", st

```

Figure 16.9 Example of Using Special System Functions from dBASE II

Note that the line numbers shown in Figure 16.9 are not part of the dBASE code; they merely provide a means of referring to lines of the code in the discussion that follows.

The scheme used in this example is to use the dBASE II CALL function to call Special System Function 15 and 24 for time and date, respectively. After each call, the PEEK function is used to read the date from the special interface transfer area. Lines 16.9-1 and 16.9-2 are necessary to set up for the calls to be used later. Line 16.9-3 sets the special system function code to 15 to read the system time. Line 16.9-4 actually calls the special system function. Lines 16.9-5 to 16.9-7 store the results of the special system function into three dBASE II variables.

Lines 16.9-8 and 16.9-9 call Special System Function 24 to read the system date. Lines 16.9-10 to 16.9-13 store the returned date into four dBASE II variables.

Note that all of the variables used to store the results will be numeric variables. Line 16.9-14 converts the numeric variables for month, day, and year into strings and concatenates them together into a string of the form "mm dd yy". Line 16.9-15 uses this string to set the dBASE II date.

Lines 16.9-16 to 16.9-18 merely display the date and time on the console; you will probably want to eliminate them if you include this code in your dBASE II command files. After the date is set, the variables "day", "wkday", "yr", "mon", and "sdate" are no longer needed and may be released. If you have no need of reading the system time, Lines 16.9-3 to 16.9-7 may be omitted.

## 16.4 Special System Function #0

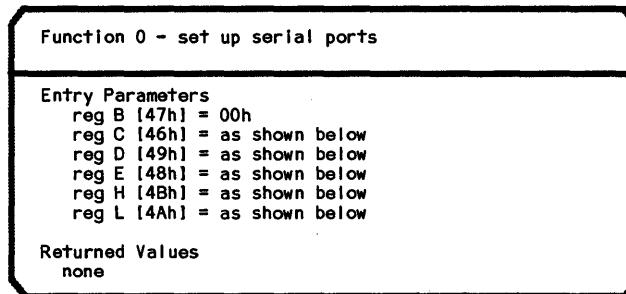


Figure 16.10

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function has two modes of operation. It can either completely setup the serial ports or it can affect only the Data Terminal Ready (DTR) and Request To Send (RTS) outputs from the serial ports. When programming the DTR and RTS outputs, only the B and C registers need to be setup. Register C should be set up as shown in Figure 16.11.

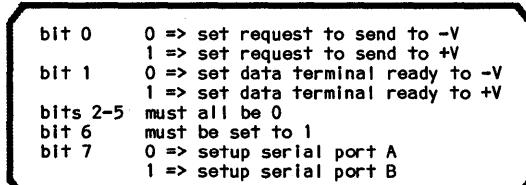


Figure 16.11 Register C Contents for Setting Status Lines Only

When fully programming the serial ports, registers C, D, E, H, and L should be set up as shown in Figure 16.12.

reg	use		
C	bit 0	1 => enable parity , 0 => disable parity	
	bit 1	1 => use even parity if parity is enabled	
	0 => use odd parity if parity is enabled		
	bits 2-6	must all be 0	
	bit 7	0 => setup serial port A 1 => setup serial port B	
D	bit 0	not used (should be set to 0)	
	bit 1	1 => set request to send (RTS) output high	
	0 => set request to send (RTS) output low		
	bits 2,3	determine the number of stop bits as follows	
		00b => not valid      10b => 1.5 stop bits	
		01b => 1 stop bit      11b => 2 stop bits	
	bit 4	not used (should be set to 0)	
	bits 5,6	determine the word length as follows	
		00b => 5 bits      10b => 7 bits	
		01b => 6 bits      11b => 8 bits	
	bit 7	1 => set data terminal ready output high 0 => set data terminal ready output low	
E	determines the baud rate as follows		
	0	=> 110	3 => 300
	1	=> 134.5	4 => 600
	2	=> 150	5 => 1200
			6 => 2400
			7 => 4800
			9 => 9600
H	contains the code for the Xmit Off character		
L	contains the code for the Xmit On character and selects the type of protocol as follows:		
	L=00h	=> no protocol is to be performed	
	L=01h	=> perform ETX/ACK protocol	
	L=02h	=> perform RS-232 status line protocol	
	02h < L < FEh	=> perform XON/XOFF protocol and use the value in L as the Xmit On character	
	L=FFh	=> do not change the Xmit On and Xmit Off characters	

Figure 16.12 Register Contents for Programming the Serial Ports

Examples showing the use of this function are given in Sections 16.2 and 16.3 of this manual.

## 16.5 Special System Function #1

Function 1 - read character from serial port A	
Entry Parameters	reg B [47h] = 01h
Returned Values	reg A [4Ch] = character reg B [47h] = status

Figure 16.13

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function allows a character to be read from serial port A. Control will not return to the calling program until a character is available. On return to the calling program, register B will contain the serial port status as defined under Function 5 below.

## 16.6 Special System Function #2

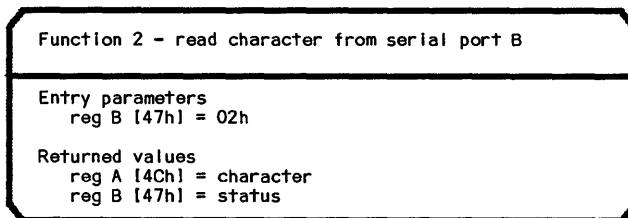


Figure 16.14

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function allows a character to be read from serial port B. Control will not return to the calling program until a character is available. On return to the calling program, register B will contain the serial port status as defined under Function 5 below.

Examples showing the use of this function are given in Sections 16.2 and 16.3 of this manual.

## 16.7 Special System Function #3

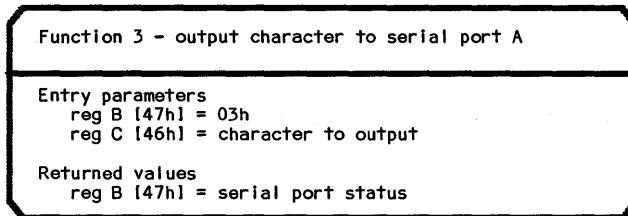


Figure 16.15

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function allows a character to be output to serial port A. Control will not return to the calling program until the character is sent. On return to the calling program, register B will contain the serial port status as defined under Function 5 below. If any form of handshaking has been enabled for port A, it will be performed. If the character cannot be sent due to the handshaking, the system will wait until it can be sent before returning to the calling program.

## 16.8 Special System Function #4

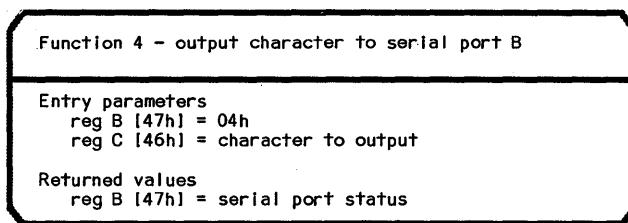


Figure 16.16

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function allows a character to be output to serial port B. Control will not return to the calling program until the character is sent. On return to the calling program, register B will contain the serial port status as defined under Function 5 below. If any form of handshaking has been enabled for port A, it will be performed. If the character cannot be sent due to the handshaking, the system will wait until it can be sent before returning to the calling program.

Examples showing the use of this function are given in Sections 16.2 and 16.3 of this manual.

## 16.9 Special System Function #5

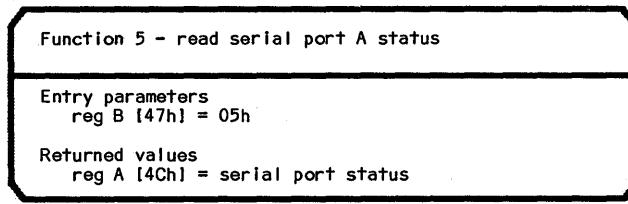


Figure 16.17

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function returns the status of serial port A. The bits in the returned status byte have the meanings shown in Figure 16.18.

bit	meaning
0	If 1 an input character is ready to be read
1	reflects the level of the clear to send input to the serial port 1 => CTS high (+V) 0 => CTS low (-V)
2	If 1 the serial port is ready to accept an output character
3	reflects the level of the carrier detect input to the serial port 1 => DCD high (+V) 0 => DCD low (-V)
4	If 1 a parity error has been detected
5	If 1 an overrun error has occurred
6	If 1 a framing error has occurred
7	If 1 a break has been detected

Figure 16.18 Serial Port Composite Status

## 16.10 Special System Function #6

Function 6 - read serial port B status	
Entry parameters	reg B [47h] = 06h
Returned values	reg A [4Ch] = serial port status

Figure 16.19

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function returns the status of serial port B. The bits in the returned status byte have the same meaning as defined under Function 5.

Examples showing the use of this function are given in Sections 16.2 and 16.3 of this manual.

## 16.11 Special System Function #7

Function 7 - read parallel printer port status	
Entry parameters	reg B [47h] = 07h
Returned values	reg A [4Ch] = OFFh if port can accept a character 00 if port not ready for a character reg C [46h] = extended status byte

Figure 16.20

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function allows direct access to status information for the parallel printer port. The extended status byte returned in register C has meaning as shown in Figure 16.21.

bit	meaning
4	0 => printer not busy, 1 => printer busy
5	0 => printer selected, 1 => printer not selected
6	0 => paper not out, 1 => paper out
7	0 => printer fault, 1 => no printer fault

Figure 16.21 Parallel Printer Port Extended Status Byte

## 16.12 Special System Function #8

Function 8 - send character to parallel printer port

Entry parameters  
reg B [47h] = 08h  
reg C [46h] = character to be sent

Returned values  
none

Figure 16.22

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function allows direct output to the parallel printer port. It outputs only to the parallel printer port and is not affected by the IOBYTE.

## 16.13 Special System Function #9

Function 9 - set parallel printer port options

Entry parameters  
reg B [47h] = 09h  
reg C [46h] = option select byte (see below)

Returned values  
none

Figure 16.23

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function sets up the parallel printer port option select byte according to the contents of register C. The bits of the option select byte should be set as shown in Figure 16.24.

bit	meaning
0	0 => pass all line feeds 1 => suppress line feeds which follow carriage returns
1	0 => pass form feed character directly to printer 1 => emulate form feeds with multiple line feeds
2	0 => no automatic form feed 1 => perform automatic formfeed
3-7	all set to 0

Figure 16.24 Parallel Printer Port Option Selection Byte

## 16.14 Special System Function #10

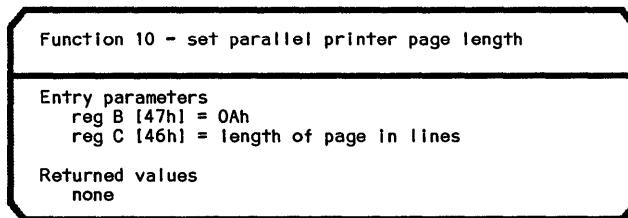


Figure 16.25

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function sets the page length for the parallel printer to the number given in register C. The page length is specified in lines and is used when emulating a form feed to determine the number of lines that must be advanced to get to the next page.

## 16.15 Special System Function #11

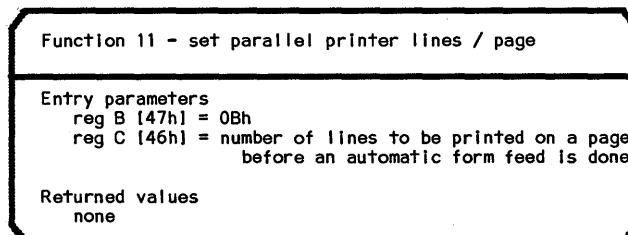


Figure 16.26

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function sets the number of lines that are actually to be printed on a page of the parallel printer. If the auto form feed option is selected, a form feed will automatically be sent to the parallel printer after this number of lines is printed. If form feeds are being emulated, the paper will be advanced by means of multiple line feeds.

## 16.16 Special System Function #12

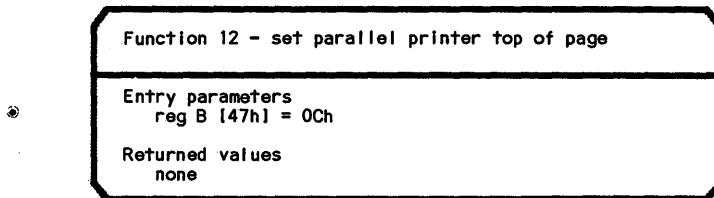


Figure 16.27

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function sets the parallel printer driver so that the current paper position is considered to be the top of form.

## 16.17 Special System Function #13

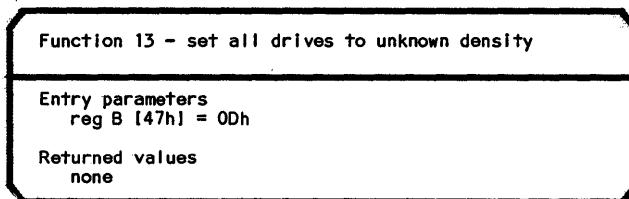


Figure 16.28

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

When changing diskettes in P&T CP/M 2, the disk system must be reset before attempting to write to the new diskette. A standard BDOS system call is available to perform the reset function, however the new diskette must be of the same density as the previously mounted diskette. In order to change densities, the system must be informed that the change is taking place so that it will redetermining the density of the diskette.

This special system function allows a program to tell the system to "forget" the density of all diskettes thus forcing the system to determine the diskette density the next time a diskette is accessed. Since the system will now redetermining the diskette density, the density of the diskettes may be changed. If, when changing diskettes, the new diskette might have a different density than the previous one, the program should call Special System Function 13 first then call the standard BDOS disk system reset.

## 16.18 Special System Function #14

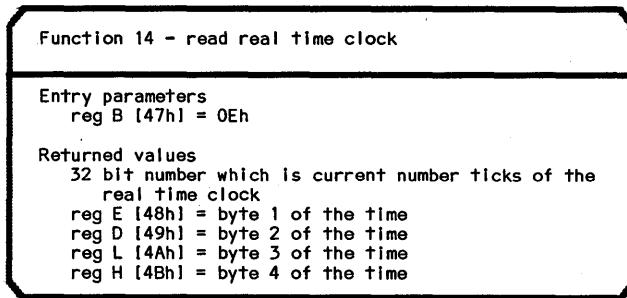


Figure 16.29

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

The real time clock starts running with a time of 0 when the system is loaded (power on boot or RESET). The time count is incremented every 1/100 second while the computer is running. The 32 bit count allows a time of 42,949,672 seconds to be accumulated (about 497 days). The main usefulness of this function is in performing timing. By noting the time when a process begins and when it ends, the total elapsed time can be computed. Note: the accuracy of this clock is determined by the main system clock and is typically within a few hundredths of a percent.

## 16.19 Special System Function #15

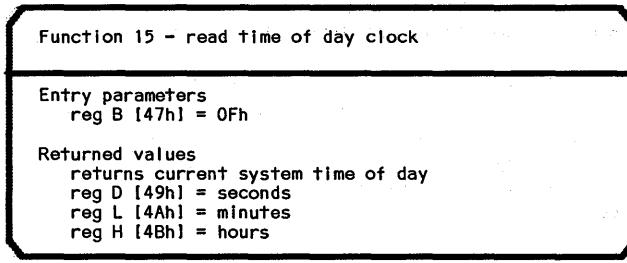


Figure 16.30

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function returns the current system time of day. When the system is loaded, the time of day is initialized to 00:00:00. It may be subsequently set with Function 16 or with the SETTIME utility routine. If the computer has a Pickles & Trout CCB clock/calendar/bell board installed, the system time of day will automatically be set to the time read from the CCB when the system is booted. Note: the system time of day will lose about 5 seconds per hour. If more accuracy is needed use Function 14 or, if a CCB is installed, read the CCB clock directly.

An example showing the use of this function is given in Section 16.3 of this manual.

## 16.20 Special System Function #16

Function 16 - set time of day clock	
Entry parameters	
reg B [47h] = 10h	
reg D [49h] = seconds to set	
reg L [4Ah] = minutes to set	
reg H [4Bh] = hours to set	
Returned values	
none	

Figure 16.31

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function allows a program to set the current value of the system time of day clock. The clock will be set immediately after the function is called. Note: no range checking is done on the entry parameters. It is the responsibility of the program to insure that the minutes and seconds are in the range 0 to 59 and that the hours are in the range 0 to 23.

## 16.21 Special System Function #17

Function 17 - read XY location of cursor	
Entry parameters	
reg B [47h] = 11h	
Returned values	
reg L [4Ah] = row address of cursor (0 - 23)	
reg H [4Bh] = column address of cursor (0 - 79)	

Figure 16.32

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function returns the current position of the cursor on the screen in row and column co-ordinates.

## 16.22 Special System Function #18

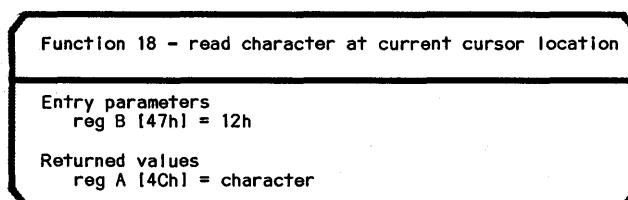


Figure 16.33

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This routine returns the character which resides on the screen at the current cursor location. No modification is made to the character when it is read from the screen; thus it will have bit 7 set if the character is in reverse video.

## 16.23 Special System Function #19

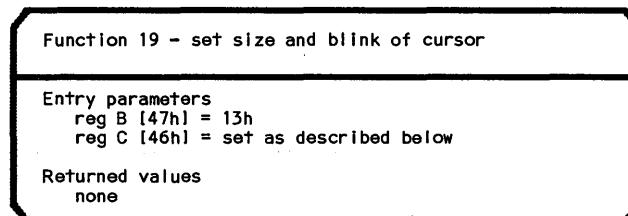


Figure 16.34

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function allows a program to control both the size and the blink rate of the cursor used on the video screen. Each character on the screen is composed of 10 rows (lines) of dots. The cursor is a reverse video block which can take from 1 line to all 10 lines of a character cell. The cursor always ends in the bottom line of a character cell (line 9) but the top line on which it starts can be modified by this function.

This function also controls the blink rate of the cursor and whether or not the cursor is displayed at all.

To get a feel for the effects of the various combinations of parameters for the cursor display, the SETMISC utility program (Section 8.25) and Option CP of the system MENU (Chapter 4) allow the user to modify the parameters at will and immediately see the results. To select the various options load register C as described in Figure 16.35.

bit	meaning
0-3	set to the line number of the character cell at which the cursor is to start. Note that no range checking is done and if a number larger than 9 is specified, the cursor will disappear altogether until a number less than or equal to 9 is specified. This is not the preferred method of turning the cursor off.
4	must be set to 0
5-6	00 => cursor does not blink 01 => cursor is not displayed 10 => cursor set to fast blink 11 => cursor set to slow blink

Figure 16.35 Cursor Setup Option Select Byte

## 16.24 Special System Function #20

Function 20 - set cursor blink and on/off	
Entry parameters	reg B [47h] = 14h reg C [46h] = 00h => no blink 20h => cursor off 40h => fast blink 60h => slow blink
Returned values	none

Figure 16.36

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function modifies only the blink rate of the cursor and whether the cursor is displayed or not. The previously defined cursor size is preserved.

## 16.25 Special System Function #21

Function 21 - enable access to screen	
Entry parameters	reg B [47h] = 15h
Returned values	none

Figure 16.37

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

For programs that have built in video drivers for a memory mapped video display, it is necessary that they enable and disable access to the screen memory via this special system function and Function 22. If a program enables the video memory on its own without the system's knowledge, there is a very strong likelihood that the system will crash. For this reason no system calls should be executed while access

to the video memory is enabled. For example, while the display memory is enabled for access, a program should not attempt to use any system functions like console input, disk I/O, etc. When the video memory is enabled for access it appears at addresses F800h through FF00h.

An example showing the use of this function is given in Section 10.11 of this manual.

## 16.26 Special System Function #22

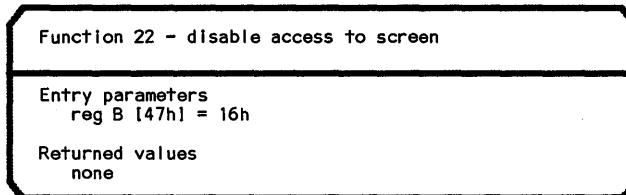


Figure 16.38

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function disables access to the screen memory.

An example showing the use of this function is given in Section 10.11 of this manual.

## 16.27 Special System Function #23

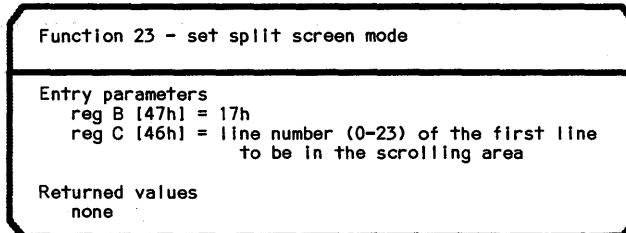


Figure 16.39

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function allows a number of lines at the top of the screen to be set to not scroll when a line feed or cursor down character is sent to the screen when the cursor is on the last line of the display. Only the lines in the scrolling portion of the display will scroll. This configuration will remain in effect until changed by another call to this function or until the system is reloaded (RESET) is done.

Note that the non-scrolling portion of the display is still accessible via normal cursor positioning commands and that the clear screen command will clear the entire screen.

## 16.28 Special System Function #24

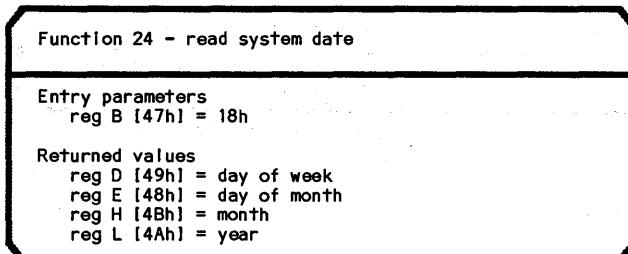


Figure 16.40

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function returns the system date in the registers as shown. Each value is returned as a binary number as shown in Figure 16.41.

item	values returned
day of week	0 - 6 (0=Sunday ... 6=Saturday)
day of month	1 - 31
month	1 - 12 (1=January ... 12=>December)
year	0 - 99 (last two digits of year)

Figure 16.41 Values Returned when Reading System Date

An example showing the use of this function is given in Section 16.3 of this manual.

## 16.29 Special System Function #25

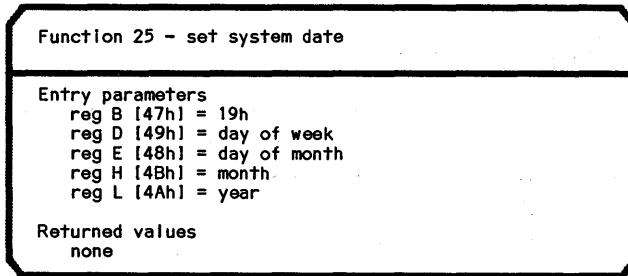


Figure 16.42

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function sets the system date according to the values in the registers as shown. No range checking is done on the date values. It is the responsibility of the program not to specify values that are out of the ranges shown in Figure 16.43.

Item	values returned
day of week	0 - 6 (0=Sunday ... 6=Saturday)
day of month	1 - 31
month	1 - 12 (1=January ... 12=>December)
year	0 - 99 (last two digits of year)

Figure 16.43 Values for Setting System Date

### 16.30 Special System Function #26

Function 26 - set console control-C trap	
Entry parameters	
reg B [47h] = 1Ah	
reg C [46h] = 00h to disable trap	
	FFh to enable trap
Returned values	
none	

Figure 16.44

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This special system function permits a program to enable and disable a <ctrl-C> (<break>) trap in the console handler. The system boots up with the <ctrl-C> trap disabled (<ctrl-C>'s allowed). If the trap is enabled, all <ctrl-C>'s will be replaced by an ASCII NUL character. While the trap is enabled it will not be possible to return to the operating system by typing the <break> key. It is the responsibility of the program to enable the trap when it does not want to be interrupted and to disable it when returning to the operating system.

### 16.31 Special System Function #27

Function 27 - set or clear drive access flag	
Entry Parameters	
reg B [47h] = 1Bh	
reg C [46h] = drive # (0 - 15)	
reg E [48h] = 00h to allow access to drive	
	FFh to deny access to drive
Returned Values	
none	

Figure 16.45

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function is available in hard disk systems only. It changes the drive access flag for the drive specified by register C. The contents of register E determine whether access to the drive is to be allowed or not. If the drive number specified is out of range, corresponds to a drive that is not on the system, or corresponds to

a diskette drive, no action is taken. Drive access control is available only for logical drives assigned to hard disks. If the drive access flag is set to deny access to a drive, any attempt to access the drive will result in the following error message being given:

Lock out error, code = HLnn

where nn is a hex representation of the disk drive number to which access was attempted.

### 16.32 Special System Function #28

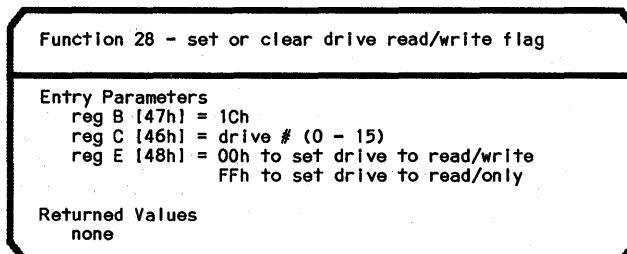


Figure 16.46

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function is available on hard disk drives only. It changes the drive read/write flag for the drive specified by register C. The contents of register E determine whether access to the drive is to be read/write or read/only. If the drive number specified is out of range, corresponds to a drive that is not on the system, or corresponds to a diskette drive, no action is taken. This function is available only for logical drives assigned to hard disks. Any attempt to write to a drive whose flag is set to read/only will result in the following error message being given:

Hard disk write prot error, code = HP00

### 16.33 Special System Function #29

Special System Function 29 is reserved for special hard disk systems. It is described in the supplementary documentation supplied with those systems.

### 16.34 Special System Function #30

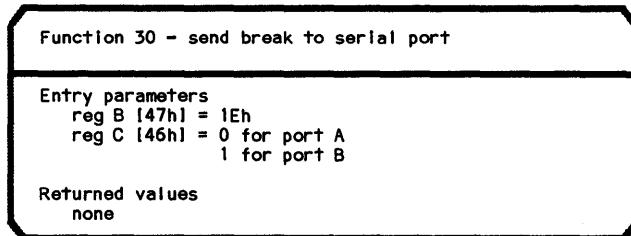


Figure 16.47

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This special system function causes a hardware break to be sent from one of the serial ports. This allows a program to issue a hardware break in cases where it is needed. The break consists of holding the transmitted data line of the serial port in the MARKING state for about 300 msec. The contents of register C determine which port is involved. Note that the system will not return control to the calling program until the break is completed.

### 16.35 Special System Function #31

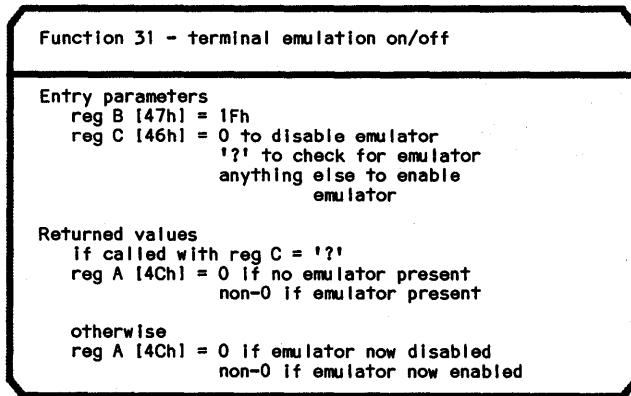


Figure 16.48

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function allows a program to determine whether a terminal emulation module (such as ADM3A) is active in the system. It also allows the program to enable or disable the emulation function of the module. Programs that wish to use the standard P&T CP/M 2 console display codes must disable any resident terminal emulator before using them.

Upon return from the function, register A will contain a code indicating the current state (after the action of the function) of the terminal emulator. If register C contains 0 when this function is called, the emulation is disabled. If register C

contains a non-zero value, the emulation is enabled unless it contains the question mark character ("?"). If register C contains a question mark, the function will have no affect other than to return the current state of the emulator.

An example showing the use of this function is given in Section 7.5 of this manual.

### 16.36 Special System Function #32

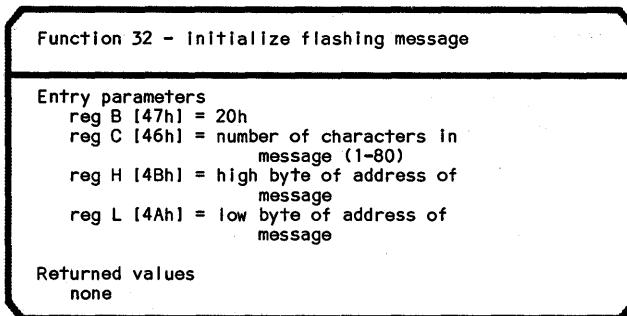


Figure 16.49

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

Special System Functions 32, 33, and 34 allow a program to access the flashing message function of the system. This function is used for many of the system error messages. It will flash (alternate between normal and reverse video) a given message in the center of the screen, temporarily replacing the text on that line. When the program is finished with the message, the screen is restored to its appearance before the message was flashed. This allows programs to put messages on the console display without disrupting forms or other information on the console displays. It also removes the burden of actually flashing the message from the program itself.

Function 32 is used to initialize the message to be flashed. When the function is called, register pair HL should contain the address of the beginning of the message and register C should contain the number of characters in the message. The message should not be longer than 80 characters. A longer message may cause information on the console display to be permanently overwritten (ie. it will not be completely restored by Special System Function 34).

An example showing the use of this function is given in Section 10.9 of this manual.

### 16.37 Special System Function #33

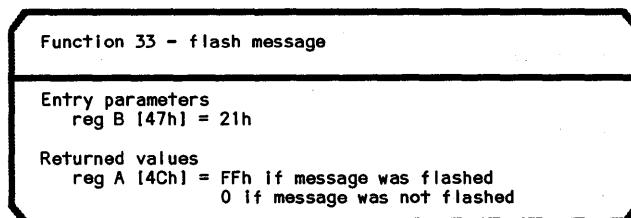


Figure 16.50

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function is used to allow the system to flash the message that was initialized by Special System Function 32. It must be called at least once every half second to insure the message will flash properly. Typically a program will enter a loop that calls Special System Function 33 and then checks for console input. It remains in this loop until the desired response is received from the keyboard. Such a loop is typically very short; hence there is no problem of too much time elapsing between calls to Function 33.

Note that Special System Function 33 returns an indicator in register A that indicates whether the message was flashed or not. This can be useful in a program that wants to flash a message a certain number of times and then continue with no user interaction. Such a program would initialize the message with Special System Function 32 and then enter a loop which repetitively call Special System Function 33. It need merely count the number of times FFh is returned to count the number of times the message has flashed. After it counts the desired number of flashes, the program would use Special System Function 34 to remove the message from the screen.

An example showing the use of this function is given in Section 10.9 of this manual.

### 16.38 Special System Function #34

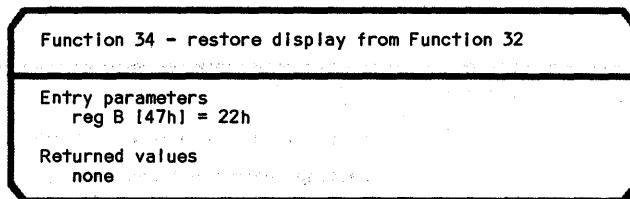


Figure 16.51

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function restores the middle line of the console display to its previous state before the flashing message was initialized by Special System Function 32. The cursor will not have moved and the screen will appear just as if the message had never been flashed.

An example showing the use of this function is given in Section 10.9 of this manual.

### 16.39 Special System Function #35

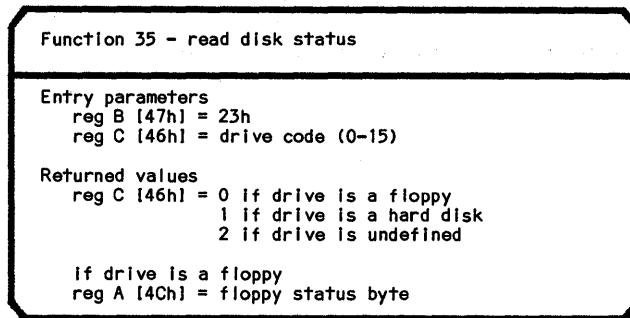


Figure 16.52

Memory locations in the high level language parameter passing area are shown in brackets ( [ ] ) for each of the registers used by this function.

This function allows a program to get a variety of information about a disk logical drive. The program can determine if the drive is present on the system, whether it is a hard or floppy drive, and, for floppy drives, read status information from the drive. This information can be useful to a program for preventing attempts to access non-existent drives, access to diskettes that are not ready, etc.

The logical drive code (0 for drive A, 1 for drive B, etc.) of the drive for which the information is desired is passed to the special system function in register C. Upon return from the function register C will contain 0 if the logical drive exists and is a floppy drive. Register C will contain 1 if the drive exists and is a hard disk drive. If 2 is returned in register C, the specified drive is not present on the system.

If the specified drive is a floppy drive, a drive status byte will be returned in

register A. Each bit of the status byte has a meaning as shown in Figure 16.53. The "Not Ready" (bit 7) and "Write Protect" (bit 6) indicators are probably the most useful to programs.

<u>bit</u>	<u>meaning if bit is set to 1</u>
0	The disk controller is busy with an operation. (should not occur)
1	The disk index hole is passing under the index sensor.
2	The read/write head is at the home position (track 0).
3	Not used.
4	Not used.
5	The read/write head is loaded against the disk.
6	The disk is write protected. The write protect notch must be covered to write enable a disk.
7	The drive is not ready. A disk is not mounted, the door is not closed, the disk is in backwards, or a double sided disk is mounted on a single sided drive.

Figure 16.53 Status Byte Returned by Special System Function 35

#### 16.40 Special System Function #36

This special system function is reserved for system usage.

## NOTES

**A.1 ASCII Character Codes**

The following table gives the decimal and hexadecimal representations of the numerical codes for ASCII characters. When a character is generated by a special key (eg. BREAK, HOLD) or a special control code on the TRS-80 Model II/12/16 keyboard, the key is noted along with the character.

<u>dec</u>	<u>hex</u>	<u>character</u>	<u>dec</u>	<u>hex</u>	<u>character</u>
0	00	NUL	64	40	@
1	01	SOH ctl-A <f1>	65	41	A
2	02	STX ctl-B <f2>	66	42	B
3	03	ETX ctl-C <break>	67	43	C
4	04	EOT ctl-D <f3>	68	44	D
5	05	ENQ ctl-E	69	45	E
6	06	ACK ctl-F	70	46	F
7	07	BELctl-G	71	47	G
8	08	BSctl-H <backspace>	72	48	H
9	09	HTctl-I <tab>	73	49	I
10	0A	LFctl-J	74	4A	J
11	0B	VTctl-K	75	4B	K
12	0C	FFctl-L <f4>	76	4C	L
13	0D	CRctl-M <enter>	77	4D	M
14	0E	SOctl-N <f7>	78	4E	N
15	0F	SIctl-O	79	4F	O
16	10	DLEctl-P <f6>	80	50	P
17	11	DC1ctl-Q	81	51	Q
18	12	DC2ctl-R	82	52	R
19	13	DC3ctl-S <hold> <f8>	83	53	S
20	14	DC4ctl-T	84	54	T
21	15	NAKctl-U <f5>	85	55	U
22	16	SYNctl-V	86	56	V
23	17	ETBctl-W	87	57	W
24	18	CANctl-X	88	58	X
25	19	EMctl-Y	89	59	Y
26	1A	SUBctl-Z	90	5A	Z
27	1B	ESC<esc>	91	5B	[
28	1C	FS<left arrow>	92	5C	\
29	1D	GS<right arrow>	93	5D	]
30	1E	RS<up arrow>	94	5E	^
31	1F	US<down arrow>	95	5F	(caret)
32	20	space	96	60	ˋ (accent grave)
33	21	!	97	61	a
34	22	"	98	62	b
35	23	#	99	63	c
36	24	\$	100	64	d
37	25	%	101	65	e
38	26	&	102	66	f
39	27	' (apostrophe)	103	67	g
40	28	(	104	68	h
41	29	)	105	69	i
42	2A	*	106	6A	j

APPENDIX A

43	2B	+	107	6B	k
44	2C	,	108	6C	l
45	2D	-	109	6D	m
46	2E	.	110	6E	n
47	2F	/	111	6F	o
48	30	0	112	70	p
49	31	1	113	71	q
50	32	2	114	72	r
51	33	3	115	73	s
52	34	4	116	74	t
53	35	5	117	75	u
54	36	6	118	76	v
55	37	7	119	77	w
56	38	8	120	78	x
57	39	9	121	79	y
58	3A	:	122	7A	z
59	3B	;	123	7B	{
60	3C	<	124	7C	<ctl-0>
61	3D	=	125	7D	}
62	3E	>	126	7E	~ <ctl-6>
63	3F	?	127	7F	DEL <ctl-minus>

## B.1 Assembling Z-80 Code

An 8080 assembler (ASM) is included with P&T CP/M 2 for developing machine language programs and routines. Since it is an 8080 assembler, it cannot directly assemble Z-80 code. If a program does not use any of the Z-80 instructions there is no problem. If Z-80 codes are desired, they can be simulated using various capabilities of the assembler.

If you do a large amount of assembly language programming with Z-80 instructions, it may be worthwhile to purchase the CP/M macro assembler (MAC). With MAC comes a library of macros which performs the simulation of the Z-80 instructions, thus freeing you from doing it directly.

To simulate Z-80 instructions using ASM, you will need to insert statements into the source code which will assemble to the desired instructions. The following table gives the substitutions to make in order to simulate Z-80 instructions. In the table, "r" is used to designate a single register (A,B,C,D,H,L,M) and "rr" is used to designate a double register (PSW, BC, DE, HL, SP). "NN" is used to designate a single byte constant. "NNNN" is used to designate a 2 byte constant (may be an address). "d" is used to designate a displacement (8 bit signed number) when used in the Z-80 instruction. "b" is used to specify a bit number (0,1,2,3,4,5,6,7) in a bit operation. "addr" is used to signify a 16 bit address (may be a label).

Z-80 instruction	simulate with
LD r,(IX+d)	DB 0DDh,r*8+46h,d
LD r,(IX+d)	DB 0FDh,r*8+46h,d
LD (IX+d),r	DB 0DDh,r+70h,d
LD (IY+d),r	DB 0FDh,r+70h,d
LD (IX+d),NN	DB 0DDh,36h,d,NN
LD (IY+d),NN	DB 0FDh,36h,d,NN
LD A,I	DB 0EDh,57h
LD A,R	DB 0EDh,5Fh
LD I,A	DB 0EDh,47h
LD R,A	DB 0EDh,4Fh
LD IX,NNNN	DB 0DDh,21h DW NNNN
LD IY,NNNN	DB 0FDh,21h DW NNNN
LD BC,(NNNN)	DB 0EDh,4Bh DW NNNN
LD DE,(NNNN)	DB 0EDh,5Bh DW NNNN
LD SP,(NNNN)	DB 0EDh,7Bh DW NNNN
LD IX,(NNNN)	DB 0DDh,2Ah DW NNNN

LD	IY,(NNNN)	DB 0FDh,2Ah DW NNNN
LD	(NNNN),BC	DB 0EDh,43h DW NNNN
LD	(NNNN),DE	DB 0EDh,53h DW NNNN
LD	(NNNN),SP	DB 0EDh,73h DW NNNN
LD	(NNNN),IX	DB 0DDh,22h DW NNNN
LD	(NNNN),IY	DB 0FDh,22h DW NNNN
LD	SP,IX	DB 0DDh,0F9h
LD	SP,IY	DB 0FDh,0F9h
PUSH	IX	DB 0DDh,0E5h
PUSH	IY	DB 0FDh,0E5h
POP	IX	DB 0DDh,0E1h
POP	IY	DB 0FDh,0E1h
EX	AF,AF'	DB 08h
EXX		DB 0D9h
EX	(SP),IX	DB 0DDh,0E3h
EX	(SP),IY	DB 0FDh,0E3h
LDI		DB 0EDh,0A0h
LDIR		DB 0EDh,0B0h
LDD		DB 0EDh,0A8h
LDDR		DB 0EDh,0B8h
CPI		DB 0EDh,0A1h
CPIR		DB 0EDh,0B1h
CPD		DB 0EDh,0A9h
CPDR		DB 0EDh,0B9h
ADD	(IX+d)	DB 0DDh,86h,d
ADD	(IY+d)	DB 0FDh,86h,d
ADC	(IX+d)	DB 0DDh,8Eh,d
ADC	(IY+d)	DB 0FDh,8Eh,d
SUB	(IX+d)	DB 0DDh,96h,d
SUB	(IY+d)	DB 0FDh,96h,d
SBC	(IX+d)	DB 0DDh,9Eh,d
SBC	(IY+d)	DB 0FDh,9Eh,d
AND	(IX+d)	DB 0DDh,0A9h,d
AND	(IY+d)	DB 0FDh,0A9h,d
XOR	(IX+d)	DB 0DDh,0AEh,d
XOR	(IY+d)	DB 0FDh,0AEh,d
OR	(IX+d)	DB 0DDh,0B6h,d
OR	(IY+d)	DB 0FDh,0B6h,d

CP	(IX+d)	DB 0DDh,0BEh,d
CP	(IY+d)	DB 0FDh,0BEh,d
INC	(IX+d)	DB 0DDh,34h,d
INC	(IY+d)	DB 0FDh,34h,d
DEC	(IX+d)	DB 0DDh,35h,d
DEC	(IY+d)	DB 0FDh,35h,d
NEG		DB 0EDh,44h
IM0		DB 0EDh,46h
IM1		DB 0EDh,56h
IM2		DB 0EDh,5Eh

for double register operations insert these equates

BC	EQU	0
DE	EQU	2
HL	EQU	4
IX	EQU	4
IY	EQU	4

ADC	HL,rr	DB 0EDh,rr*8+4Ah
ABC	HL,rr	DB 0EDh,rr*8+42h
ADD	IX,rr	DB 0DDh,rr*8+9h
ADD	IY,rr	DB 0FDh,rr*8+9h
INC	IX	DB 0DDh,23h
INC	IY	DB 0FDh,23h
DEC	IX	DB 0DDh,2Bh
DEC	IY	DB 0FDh,2Bh
BIT	b,r	DB 0CBh,b*8+r+40h
SET	b,r	DB 0CBh,b*8+r+0C0h
RES	b,r	DB 0CBh,b*8+r+80h
BIT	b,(IX+d)	DB 0DDh,0CBh,d,b*8+46h
BIT	b,(IY+d)	DB 0FDh,0CBh,d,b*8+46h
SET	b,(IX+d)	DB 0DDh,0CBh,d,b*8+0C6h
SET	b,(IY+d)	DB 0FDh,0CBh,d,b*8+0C6h
RES	b,(IX+d)	DB 0DDh,0CBh,d,b*8+86h
RES	b,(IY+d)	DB 0FDh,0CBh,d,b*8+86h
JR	addr-\$	DB 18h,addr-\$-1
JR	C,addr-\$	DB 38h,addr-\$-1
JR	NC,addr-\$	DB 30h,addr-\$-1
JR	Z,addr-\$	DB 28h,addr-\$-1
JR	NZ,addr-\$	DB 20h,addr-\$-1
DJNZ	addr-\$	DB 10h,addr-\$-1
JMP	(IX)	DB 0DDh,0E9h
JMP	(IY)	DB 0FDh,0E9h
RETI		DB 0EDh,4Dh

RETN	DB 0EDh,45h
IN r,(C)	DB 0EDh,r*8+40h
OUT (C),r	DB 0EDh,r*8+41h
INI	DB 0EDh,0A2h
INIR	DB 0EDh,0B2h
IND	DB 0EDh,0AAh
INDR	DB 0EDh,0BAh
OTI	DB 0EDh,0A3h
OTIR	DB 0EDh,0B3h
OTD	DB 0EDh,0ABh
OTDR	DB 0EDh,0BBh
RLC r	DB 0CBh,r
RLC (IX+d)	DB 0DDh,0CBh,d,6h
RLC (IY+d)	DB 0FDh,0CBh,d,6h
RL r	DB 0CBh,r+10h
RL (IX+d)	DB 0DDh,0CBh,d,16h
RL (IY+d)	DB 0FDh,0CBh,d,16h
RRC r	DB 0CBh,r+8h
RRC (IX+d)	DB 0DDh,0CBh,d,0Eh
RRC (IY+d)	DB 0FDh,0CBh,d,0Eh
RR r	DB 0CBh,r+18h
RR (IX+d)	DB 0DDh,0CBh,d,1Eh
RR (IY+d)	DB 0FDh,0CBh,d,1Eh
SLA r	DB 0CBh,r+20h
SLA (IX+d)	DB 0DDh,0CBh,d,26h
SLA (IY+d)	DB 0FDh,0CBh,d,26h
SRA r	DB 0CBh,r+28h
SRA (IX+d)	DB 0DDh,0CBh,d,2Eh
SRA (IY+d)	DB 0FDh,0CBh,d,2Eh
SRL r	DB 0CBh,r+38h
SRL (IX+d)	DB 0DDh,0CBh,d,3Eh
SRL (IY+d)	DB 0FDh,0CBh,d,3Eh
RLD	DB 0EDh,6Fh
RRD	DB 0EDh,67h

## C.1 Modules Included With Standard System

The following modules are included in the module library that comes with P&T CP/M 2. You may select from among them to customize the system to your needs. See Chapters 6 and 7 for more information about selecting modules.

**STDCORE1** This module contains the basis of the I/O routines onto which other modules can be added. This module is automatically selected for you by the MODSEL utility program.

**STDWB1** This module contains the code to perform a warm boot operation for a Model II or 16 floppy only system. This module requires that a system diskette be mounted to perform the warm boot. If you have a Model 12 or 16B use the M12WB1 module. If you have a hard disk system, there may be additional warm boot modules in the library. See the hard disk addendum for information.

**M12WB1** This module contains the code to perform a warm boot operation for a Model 12 or 16B. This module does not require that a system diskette be mounted to perform a warm boot.

**STDCRT1** This module contains the video display software for the Models II and 16. It will ring the bell on the P&T CCB board (if one is installed) when a <ctl-G> is sent to the console display.

**M12CRT1** This module contains the video display software for the Models 12 and 16B. It will ring the internal bell in the computer when a <ctl-G> is sent to the console display.

**1FLOPPY** This module contains drive parameters for a single floppy system. When this system is included in the system, it will perform disk swapping to emulate a 4 drive system. (See Section 3.3 for details of disk swapping.)

**2FLOPPY** This module contains drive parameters for a two floppy system.

**3FLOPPY** This module contains drive parameters for a three floppy system.

**4FLOPPY** This module contains drive parameters for a four floppy system.

**SIO** This module contains the standard serial port drivers. It is not required by the system but must be included to access the serial ports.

**PPSTD** This module contains the standard parallel printer port driver (for use with older Radio Shack printers). This driver includes options that attempt to correct for the automatic line feeds generated by Radio Shack printers. It does not, however, attempt to use a reverse line feed to correct for an isolated carriage return. If you have a Radio Shack printer that does not perform reverse line feed, you should use this module. If, for example, a program sends an isolated carriage return to the printer, the expected response is to return the print head to the left side of the paper so that the line can be overprinted. With this module, the printer will end up on the next line because of the automatic line feed. This module checks for the "select", "paper out", and "fault" errors and reports them as described in Chapter 13.

**PPDSTD** Same as PPSTD but for the newer "DMP" and "DWP" Radio Shack Printers. This module checks for the "paper out" and "fault" errors and reports them as described in Chapter 13.

**PPSTD** This module is the same as PPSTD except that it makes use of reverse line feed to correct for isolated carriage returns. It is for use with the older Radio Shack Printers. When an isolated carriage return is sent to the printer, the module detects it and issues a reverse line feed before sending any additional characters to the printer. In this case, the paper will advance to the next line when the carriage return is sent but the reverse carriage return will immediately return it to the line that was being printed when the isolated carriage return was sent. This allows overprinting of lines.

**PPDSTD** Same as PPSTD but for the newer "DMP" and "DWP" Radio Shack Printers. This module checks for the "paper out" and "fault" errors and reports them as described in Chapter 13.

**PPNORS** This module contains a parallel printer port driver that does not have options for accomodating Radio Shack printers. It does, however, check for the "select", "paper out", and "fault" errors and reports them as described in Chapter 13. You should use this module if you have a normal parallel printer or have deactivated the auto linefeed on your older Radio Shack printer.

**PPDNORS** Same as PPNORS but for the newer "DMP" and "DWP" Radio Shack Printers. This module checks for the "paper out" and "fault" errors and reports them as described in Chapter 13.

**PPMIN** This module contains the minimum parallel printer port driver. It merely sends all characters to the printer without modification. If any of the error conditions are detected when a character is being sent to the printer, it simply throws that character away and proceeds as if it had been sent to the printer.

**PPDMIN** Same as PPSTDR but for the newer "DMP" and "DWP" Radio Shack Printers.

**AUTOKEY** This module provides 5 programmable function keys. The keys can be programmed at any time either from a program or directly from the keyboard. See Chapter 7 for details.

**KEYXLATE** This module allows you to translate the character codes generated by up to 16 keys on the keyboard into other codes. This allows you to tailor the keyboard to your needs. You can set the character translations either from a program or by using the KXEDIT utility program. See Chapter 7 for details.

**SCRNDUMP** This module allows you to dump the contents of the console display to the system printer by pressing (ctl-equal). See Chapter 7 for details.

**ADM3A** This module causes the console display and keyboard to appear to a program as if it were a ADM3A terminal. This module is used in addition to the standard video display driver software. This can be useful when trying to run certain screen oriented programs. Note that it is usually better to install such programs for the standard P&T CP/M 2 display control codes than to use this module. See Chapter 7 for details.

## Glossary

**Active drive:** A drive which has been accessed since the system was last reset.

**Advanced Command Processor (ACP):** A Pickles & Trout supplied replacement for the standard CCP. It makes it easier to use SUBMIT files and user numbers, allows one to edit command lines, and includes eight extra built-in commands and three extra utilities.

**Allocation block:** The smallest section of disk space used by a file. An allocation block is 1 Kbyte on a single density diskette, 2 Kbytes on a double density diskette, and up to 16 Kbytes on a hard disk. A file may use as many allocation blocks as exist on a logical drive.

**ASCII (American Standard Code For Information Interchange):** A standard seven-bit character code used to represent characters.

**Assembler:** A computer program that translates an assembly language program into hex or binary (machine) code.

**Base page:** See Page Zero, System Parameter Area.

**Baud:** The number of signal changes per second in a communications system. Commonly considered a statement of how many bits can be communicated per second.

**BDOS (Basic Disk Operating System):** The part of the operating system that contains the general, machine independent programs for operating a computer's peripheral devices. It regulates input/output operations and maintains disk data structures. Stored on tracks 0 and 1.

**Binary:** Base 2 number system, using the digits 1 and 0, corresponding to the 'on' and 'off' states of digital circuits. Binary codes can thus be easily stored in a computer's memory and on a disk.

**BIOS (Basic Input/Output System):** That part of the operating system that contains the customized, hardware-related programs that control a computer's peripheral devices. Any significant change in these devices thus requires a change in the BIOS. CP/M 2.2m stores the BIOS programs in BIOSPARM.PNT and BIOSMODS.PNT.

**Block:** See allocation block.

**Buffer:** A storage unit or portion of internal memory used to store information to

compensate for a difference of speed between computer devices.

**Built-in commands:** The CCP commands (like DIR, ERA, REN, etc.) that are part of the CCP and thus do not have to be executed from a disk (as transient programs do).

**Byte:** A standard unit of memory or disk storage consisting of 8 bits. Typically, each byte contains the code for a letter, number, or other symbol.

**CCB (Clock/Calendar/Bell):** The P&T CCB is a combined clock, calendar, and audio alarm. The clock maintains the correct time and date (even when the microcomputer on which it is installed is off), and will sound an alarm if a program sends 1-G to the console.

**Cold boot:** The process of loading the entire operating system - that is, the BDOS, BIOS, and CCP - into memory and initializing various system operations. A cold boot requires a disk with the BDOS and CCP on tracks 0 and 1, and BIOSPARM.PNT and BIOSMODS.PNT on the data tracks. Contrast with 'warm boot'.

**COM:** A file extension (or type) that designates a CP/M file that may be executed. Hence, the phrase 'COM file' refers to an executable file.

**Command level:** The status when the CCP is waiting for a command from the console. The command level is denoted by a drive letter and command prompt (e.g., A).

**Command line:** A command sent to the CCP from the console or a SUB file which includes not only the program's name, but also arguments which shape how it is to be executed.

**Command line delimiter:** The special characters (. : ; < > — blank <cr>) which are used to separate different items on a command line.

**Compiler:** A program that converts the source code of a high level language into binary (machine) code so that it can be executed.

**CON:** A logical device name that represents the system console and keyboard.

**Console:** The console typically consists of a keyboard and CRT. It is the part of a microcomputer system which allows direct communications between a user and the microcomputer.

**Console Command Processor (CCP):** The part of the operating system that processes commands coming from the console or a SUB file. It contains the built-in commands (such as DIR, ERA, and REN) and initiates the execution of transient programs. (See Advanced Command Processor)

**CP/M prompt:** Two characters, the letter of the current drive and , which indicate that the system is ready to execute a command. When the ACP is installed, the command prompt also includes the current user number.

**Current drive:** The drive that is used for disk operations if no other drive is specified in a command. The current drive is sometimes called the 'logged on' drive, and will be activated as part of a warm boot operation.

**Cursor:** A symbol appearing on the console screen in reverse video. It indicates where the next typed character will appear. The cursor may vary in size and may blink.

**Data disk:** A disk on which only data files are stored. Since it does not have a copy of the operating system on tracks 0 and 1, it cannot be in the system drive if a diskette is needed during a warm boot operation. Contrast with system disk.

**Data file:** A file containing information that will be processed by a program. Contrast with an executable (COM) file.

**Density:** A measure of the number of useful storage positions per unit of length or area. For instance, there are 26 logical sectors per track on a single density diskette and 64 logical sectors per track on a double density diskette.

**Destination drive:** A logical drive on which information is to be stored.

**DIR (file attribute):** This attribute tags a file so that it will appear on a directory display. Contrast with SYS.

**Directory:** A catalog of the files stored on a logical drive. It contains addressing information which is used by the system for locating a file on the data tracks of a drive.

**Disk/diskette:** A thin disk of some sort of base material which is coated with a compound that can be easily magnetized to depict binary code and which can thus be used for mass storage. Diskette always refers to a 'floppy' mylar disk; disk may refer to either a soft mylar or a hard metal disk.

**Disk drive:** A device which reads and writes information on disks/diskettes.

**Editor:** A program that is used to write and/or modify ASCII text files on the console. ED is a

typical 'line' editor, while VEDIT is one of the better-known 'full-screen' editors.

**Extension:** The second part of a filename. It can be up to 3 characters in length and is typically used to denote the type of data in the file (e.g., a COM file consists of executable binary code and a PAS file consists of PASCAL source code). An extension is also called a secondary file name.

**Executable file:** A binary (machine) code file which can be executed by a computer without further modification. It is signified by a COM extension.

**Extent, logical:** A group of 16K consecutive bytes in a file. A file consists of one or more such extents.

**Extent, physical:** A directory entry which is added to a primary entry when a file requires more addressing information than the first entry can accommodate.

**File:** A collection of related data. It can be referenced with a unique name and treated as a unit.

**File name:** A unique series of characters that represent a file in storage. It consists of from 1-8 characters, and a 0-3 character extension. When writing a file name, the extension follows the primary name and is separated from it by a period. The extension is often used to indicate the type of information stored in the file. Programs will frequently require particular extensions for the files that they use.

**File specification:** When complete, a filename will include: (1) the letter of the logical drive on which it is stored followed by a colon, (2) a primary name of 1-8 characters, (3) a period, and (4) a file extension of 0-3 characters (e.g., B:SAMPLE.TXT).

**File type:** See extension.

**Hertz:** A measure of cycles per second. Commonly used with kilo- and mega- to refer to thousands and millions of cycles per second (respectively).

**HEX file:** A hexadecimal representation of a binary (machine) language file. Unlike the binary file, it may be printed.

**Hexadecimal:** A base 16 number system, with digits that range from 0 to F (the equivalent of decimal 15).

**High level language:** A programming language whose structure is independent of the design of the computer; for example, COBOL and Fortran.

**Image copy:** A direct byte-by-byte copy of data from one diskette to another.

**IOBYTE:** The byte at location 3 of memory which contains the current assignments of the four logical devices (the console, list, punch, and reader).

**List device:** Device on to which data can be listed or printed (signified by LST:).

**Logical drive:** A storage unit. A logical drive is the equivalent of a physical disk drive except in two situations: the physical drive in a single drive system is always assigned four logical drives and several logical drives may be assigned to a hard disk.

**Logical sector:** A 128 byte storage area on a disk. See sector.

**Logged on drive:** See active drive. **LST:** The logical name for a list device (usually a printer).

**Machine code:** Binary code that is used directly by a computer.

**Macro:** A statement in an assembly language that generates a number of machine language instructions.

**Millisecond:** One thousandth (0.001) of a second. Commonly abbreviated ms or msec.

**Modem:** A device that translates digital data so that it can be transmitted through a telephone network.

**Multiprocessing:** The simultaneous execution of two or more computer programs.

**Object file:** Binary or hex file. If binary, it may be executed directly; if hex, it must be translated into binary code (using, for instance, the LOAD utility). Contrast with source file.

**Parity bit:** A binary digit added to a group of bits to make the total count of the '1' bits either odd (odd parity) or even (even parity).

**Page:** 256 (100h) consecutive bytes in memory.

**Page zero:** The first page (or 256 bytes) of memory. Also called the base page. See System Parameter Area.

**Primary file name:** The first part of a file name. It may be 1-8 characters in length and is used to differentiate the file from other files listed in a directory.

**Program file:** A file containing a program that can be executed directly. It must have a COM extension. Such a program is executed by typing its primary name at the command prompt.

**PUN:** The logical name for an output-only device. Today it is commonly used to reference an alternative printer or a modem.

**Random access file:** A file composed of addressed records which can be read or written to individually.

**Random Access Memory (RAM):** Also, and more accurately, called read/write memory. The internal memory of a computer that can be both read and written to. Contrast with ROM.

**RDR:** The logical name for an input-only device, which is commonly used when moving files between microcomputers (using PIP for instance).

**Read-only or R/O (drive and file attribute):** This attribute tags a drive or file so that it can be read but not written to. A warm boot will re-tag a drive as R/W but will have no effect on a file tagged as R/O.

**Read-write or R/W (drive and file attribute):** This attributer tags a drive or file so that it can be both read and written to. A warm boot will re-tag a R/O drive as R/W, but will have no effect on a file tagged as R/O.

**Record, logical:** A group of related fields that form a unit of information. Combined to form a file.

**Record, physical:** A synonym for sector. Used by STAT for instance.

**Reverse video:** A reversal of the standard (white on black background) display on CRT screens so as to highlight characters, words, or lines.

**ROM (Read Only Memory):** Pre-programmed memory in a computer that can be read but not written to.

**RS-232 standard:** A standard set of pin signals on a data communications device.

**Secondary file name:** See extension.

**Sector:** A series of individual storage areas on a magnetic disk. The basic CP/M sector contains 128 bytes. These 'logical' sectors are often grouped into larger 'physical' sectors (on double density disks for instance). Sometimes called a 'record'.

**Serial:** Pertaining to a bit-by-bit transmission of data. A serial port is a port through which data are communicated serially.

**Source file:** An ASCII text file which is an input file to a program such as an assembler or compiler. Source files are normally written in languages such as COBOL and Fortran.

**SUB files:** A file containing prototype commands and program input lines which are to be submitted for batch processing.

**SYS (file attribute):** This attribute tags a file so that it will not appear on a directory display.

**System drive:** The diskette drive from which a cold boot is performed. This is always drive 0 on the Mods II/12/16.

**System Parameter Area (SPA):** The first 256 bytes of memory. It contains such key information as the current drive and user numbers, the addresses of BIOS and BDOS, the assignments of the peripheral devices, the restart locations, and the default buffer.

**System tracks:** Tracks 0 and 1 on a floppy disk. They are reserved for system use and will not be affected by operations on the data stored on the data tracks. P&T CP/M 2.2m stores the CCP and BDOS on the system tracks.

**Terminal:** A device like a keyboard (and normally a CRT) which allows a user to communicate directly with a computer.

**Tracks:** Concentric rings on a disk on which data are stored. Numbered from 0 to 77 on a single sided diskette and 0 to 154 on a double sided diskette.

**Transient Program Area (TPA):** The working area of memory where executable programs and their data are loaded for execution. It starts at location 100h of memory.

**Transient programs:** Programs that are stored on disks and thus have to be loaded into the TPA to be executed. For instance, CLONE, FORMAT, VEDIT, and other purchased programs.

**User file:** A file that is not a system file (with the SYS attribute) and is thus listed in a normal directory display.

**User number:** One of 16 (0-15) numbers assigned to files to distinguish them from files used by another person or for other purposes. When booted, the system defaults to user 0; it may be moved to another user number by executing the USER built-in command of the CCP.

**Utility:** One of several system programs which perform such operations as formatting and copying diskettes, setting system parameters, etc. Examples include CLONE, SETUP, and SYNCHRO.

**Verify:** Determine whether a data recording operation has been accomplished accurately.

**Warm boot:** A warm boot reloads the CCP and BDOS into memory, re-initializes the jumps into BDOS and BIOS, and activates drive A and the current disk drive. A warm boot must be performed when a disk to be written to is mounted. It is usually performed at the end of a program's execution.

**Wildcard characters:** Two special characters, \* and ?, used to replace one or more characters in a filename. An \* replaces one or more characters; a ? replaces one character.

**Wildcard file name:** A file name specification containing one or both of the wildcard characters, \* and ?. It can thus match several file names on a disk. For instance, \*.\* refers to every file in a directory.

## Index

- A option  
  DDT, 8.29-32(f)  
  ED, 8.56  
Abort character (See <ctl-C> and <break>)  
Accent grave character, 7.5, 8.80, 11.1  
Access attributes (ACC), 8.131  
Accessing a disk, 8.134  
ACP (See Advanced Command Processor)  
Addressing on disks, 5.3  
ADM3A module, 7.1, 7.6-7, 10.1, 16.23  
ADM-3A terminal, 7.1, 7.6-7, 10.1  
Advanced Command Processor (ACP), 8.95  
AE (Automatic execution, MENU), 4.8, 5.11  
Afn (See wildcard file name)  
AK (Save autokeys, MENU), 4.9, 7.3  
ALL command line option  
  CLONE, 8.21(f)  
  FASTCOPY, 8.59, 8.63(f), 8.65  
Allocation block, 5.7-8, 8.131, 8.134  
Application programs, 1.1  
Arrow keys, 11.2(f), 11.5  
AS (Assign devices, MENU), 4.10  
ASCII (American Standard Code for Information  
  Interchange), 5.4, 5.7, 8.31, 8.78, 8.97, 8.99, 8.123,  
  8.142, 10.5  
Control characters, 11.1  
Conversion to lower case, 8.96, 11.1  
Conversion to upper case, 8.96-98  
ASM (DRI assembler), 8.4-8, 8.100, 8.132  
ASM extension, 8.4-8  
Assembly language, 8.100, 8.138-139  
  Bibliography, 8.8(f)  
  Execution of programs, 8.5, 8.29-35, 8.52-52  
  Programming in, 8.4-8  
  Programs, 8.5, 8.31-35, 8.100, 14.2(f)-3(f), 15.2(f)  
  With special system functions, 16.1-4  
ASSIGN (P&T utility), 4.10, 5.10, 8.2, 8.9-12, 8.91,  
  8.122, 8.134, 12.3, 13.1, 14.2  
\* (wildcard character), 5.6  
Attribute set, 8.131-132(f)  
Auto form feed (See form feed)  
Auto new line, 10.6-7  
AUTOKEY (P&T module), 4.9, 4.22, 4.34, 6.7(f),  
  7.1-5, 11.5  
Automatic execution upon booting, 4.8
- B option of PIP, 8.94(f)  
Bad Sector, error message, 9.2  
Back slash character, 11.1(f)  
Backing up, 3.3, 3.6, 8.39 (See CLONE, FASTCOPY,  
  PIP)  
Backspace, 10.2-4  
  <backspace> key, 7.2, 11.2  
Bank switching video memory, 10.12  
Base page (See System Parameter Area)  
Basic Input/Output Subsystem (See BIOS)  
Basic-80 (MBasic), 11.4  
  IOBYTE, 14.3  
Program fragments, 8.156(f), 8.158(f)  
Video display, 10.5-12  
With special system functions, 16.4-5
- Basic programs, 7.2-3, 10.9(f)-10(f), 10.13(f)  
BAT: physical device, 5.9(f), 8.135(f)  
Baud Rate, 4.4, 8.122-133(f), 8.125(f)-126, 12.2-3, 16.2  
BDOS (Basic Disk Operating System), 5.1, 5.19, 6.1-2,  
  11.3-4, 15.1, 16.1-3(f)  
  Entry point, 15.4  
  Error messages, 8.57(f), 9.1-4  
BDOS Disk R/O Error, 8.134(f)  
BDOS functions, 10.12, 11.3, 16.1-3  
  IOBYTE, 14.2-3  
  #1: Console input, 11.2-3  
  #6: Direct console I/O, 11.2-3, 16.2  
  #7: Getting IOBYTE, 14.2(f)  
  #8: Setting IOBYTE, 14.2(f)  
  #9: Print a string, 14.2(f)  
  #10: Read console buffer, 8.151-152  
  #13: Disk system reset, 16.14  
  #28: Write protect disk, 3.5  
  #30: Set file attributes, 5.8-9  
BEL (ASCII code), 8.104  
Bell, 8.104, 10.2  
Binary code, 5.4, 8.82, 8.97 (See object code)  
BIOS (Basic Input/Output Subsystem), 5.1, 5.20,  
  15.1-4  
  Address, 15.3  
  Entry point, 15.1  
  Error messages, 8.57, 9.15  
  Jump table, 15.2-4  
BIOSMODS.PNT file, 6.1-4(f), 6.8, 8.81, 9.4-5  
BIOSPARM.PNT file, 4.4-5, 4.8-9, 4.18-19, 4.27, 4.29,  
  6.1-5, 6.7, 6.9, 7.3, 8.78-79, 9.4-5  
  Changing from MENU, 4.4-5  
BOOT (BIOS routine), 15.3(f)  
<break> key, 3.4, 8.32, 8.35, 9.1-2, 11.2  
Trap, 11.4-5, 16.21  
Breakpoints, and DDT, 8.32, 8.35  
Buffered input (type-ahead buffer), 5.20, 11.2  
Buffered output, 10.7  
Buffering, and PIP, 8.94  
Buffers, system, 5.1  
BUSY status line, 13.1  
Bytes (STAT header), 8.131
- CA (Change access to CCB, MENU), 4.11, 8.106  
Cables, to serial port, 12.4-5  
Calendar, 8.109  
Call commands, 16.1-6  
<caps> key, 11.1  
Carriage return, 2.1, 10.2, 11.2  
Carriage return-line feed (CRLF), 8.147, 10.6-7  
  Suppress, 10.9(f)  
Carrier detect pin, 12.4-5  
CBASIC, and random access files, 5.8  
Cbase, 5.1(f)  
CCB (P&T Clock/calendar/bell), 1.3, 4.11, 4.16, 4.33,  
  4.36-37, 5.19, 8.104-107, 8.140, 8.148, 10.2, 16.15  
  Manual, 4.11  
Port address, 4.11, 8.106, 8.111, 8.113, 8.117  
CCP (See Console Command Processor)  
CD (Copy diskette, MENU), 4.12

CE (Disk check, MENU), 4.13  
 Central Processing Unit (CPU), 1.1 (See 8080, Z-80)  
 Centron physical device, 5.9-10(f), 8.10-11(f), 8.123(f), 8.135(f), 13.1  
 Centronics port, 4.32, 13.1-4 (See Parallel port)  
 CH (Clean heads, MENU), 4.14  
 Chaining facility, and the command line mode, 8.1  
 Checking a diskette, 3.2, 4.12-13, 4.23 (See CE, TE, DISKCHK, DISKTEST)  
 CHR\$ Basic function, 10.5, 10.8(f)  
 CLEAN (P&T utility), 3.4, 4.14, 8.13-14  
 Clear screen, 7.6(f), 10.2, 16.19  
 Clear to Send (CTS) line, 12.2, 16.10(f)  
 Pin, 12.4-5  
 Clock/calendar/bell (See CCB)  
 CLONE (P&T utility), 3.2, 4.12, 4.25-26, 5.7, 8.15-26, 8.22, 8.155, 9.4  
 Cold boot, 4.8-9, 5.19-20 (See load)  
 CMD dBase II file, 8.160(f), 16.6-7  
 COM file extension, 4.23-24, 5.19, 8.4, 8.82-83, 8.97  
 Command buffer, 8.139  
 Command file, 4.23-24, 8.137-139, 8.154-160 (See SUBMIT)  
 Command level (or command mode), 3.4, 4.21, 5.11-12, 5.19  
 Command line, limitation on length of, 8.116, 8.139  
 Command line arguments  
   ASM, 8.5  
   CLONE, 8.19(f)-21(f), 8.155(f)  
   CCP, 8.152  
   DATIME, 8.28(f)  
   DISKCHK, 8.40-41(f), 8.156  
   DISKTEST, 8.49-50(f), 8.159  
   FASTCOPY, 8.63-65(f)  
   FORMAT, 8.72-73(f), 8.154-155(f), 8.159  
   STAT, 8.135  
   SUBMIT, 8.137  
   VERIFY, 8.149  
 Command line mode, 1.2-3, 4.8, 8.1-2, 8.130, 8.137, 8.139, 8.154-160  
 Comment lines (examples), 8.137, 8.152, 8.157  
 Command prompt, 5.11  
 Computer, maintenance and starting, 3.4  
 CON: command line option  
   CLONE, 8.20(f)-21  
   DATIME, 8.27-28(f)  
   DISKCHK, 8.42(f)-43  
   DISKTEST, 8.49(f)-50  
   FASTCOPY, 8.63(f), 8.65  
   FORMAT, 8.72(f)-73  
 CON: logical device, 5.10-11(f), 8.9-12(f), 8.92(f), 8.96, 8.99, 8.123(f), 8.135(f), 10.1, 12.3, 13.1, 14.2  
 Concatenation, using PIP, 8.90, 8.93, 8.99  
 CONCHAR\$ CBASIC function, 11.3  
 CONIN (BIOS routine), 15.3(f)  
 CONOUT (BIOS routine), 15.3(f)  
 Console, 5.9-10  
   Assignment, 14.1  
   Control codes, 7.6-7(f), 10.2-4(f)  
   Handler, 16.21  
   Input, 16.2, 16.25  
   Interface, 1.3  
   Modules, 6.2  
   (See video)  
 Console Command Processor (CCP), 5.1, 5.11-20, 6.1-2, 8.29, 8.34-35, 8.137-138, 8.152, 9.3-4, 11.2-4, 13.4, 14.3, 15.4(f)  
   And file names, 5.7  
   Functions, 13.4  
   Console display, 10.13  
     Functions, 10.8-9  
     Input, 5.20  
     Memory, 4.30  
   CONST (BIOS routine), 15.3(f)  
   Control characters, 2.1, 5.20, 7.2, 8.139, 11.1  
   Control codes, video, 2.1, 7.6-7, 10.1-4  
   (ctl) key, 11.1  
   (ctl-C), 9.2, 11.4 (See break, warm boot)  
   (ctl-E), 7.6  
   (ctl-P), 5.20  
   (ctl-S) (same as <hold>), 5.13, 5.17, 5.20, 11.3-4, 12.2  
   (ctl-T), 13.2  
   (ctl-X), 11.3-4  
   (ctl-Z) end-of-file marker, 8.97, 8.99, 8.141-2  
     Adding to file, 8.92(f), 8.99  
   Copying a file or diskette (See CD, CLONE, FASTCOPY, PIP)  
     Using a hard disk, 8.59  
   Core Modules, 6.7 (See STDCORE1)  
   CP (CRT parameters, MENU), 4.15, 10.11, 16.17  
   CP/M books, 1.5  
   CRT Modules, 6.2 (See STDCRT1)  
   CRT physical device, 5.9-11(f), 8.9-11(f), 8.92(f), 8.123(f), 8.135(f), 10.1, 14.1-2  
     Parameters, 8.111-112  
   CRUN, and random access files, 5.8  
   CS (Set CCB, MENU), 4.15  
   Current default drive, 5.11  
     Single drive system, 3.6-7  
   Cursor  
     Addressing, 7.6(f), 10.4-5, 16.19  
     Blink, 4.15(f), 8.111-112, 8.117, 8.119, 16.17-18(f)  
     Home, 7.6(f), 10.2-3  
     Location of, 16.16-17  
     Movement, 7.6(f), 10.2-4  
     On/off, 10.2, 10.4, 16.18  
     Size, 4.15(f), 8.111-112, 8.117, 16.17-18(f)  
 D option  
   DDT, 8.30(f)-31(f)  
   PIP, 8.94(f)  
 Data area on diskette, 5.3-4(f)  
 Data diskette, 4.23  
   And drive A, 3.5  
 Data files, 3.3, 8.141-2, 8.146(f)  
 Data set ready pin, 12.4-5  
 Data terminal ready (DTR:), 8.123(f), 8.125(f), 8.127, 12.2-5, 16.2, 16.7-8(f)  
 Data tracks, copying, 8.16(f) (See FASTCOPY, PIP)  
 Date, system, 8.108-110, 16.20  
 DATIME (P&T utility), 4.20, 8.27-28, 8.55(f)  
 dBase II, 8.159-160  
   With special system function, 16.6-7  
   (See CMD dBase II file)  
 DDT (DRI utility), 8.4, 8.29-35, 8.95, 8.96(f), 8.100, 8.142, 15.4(f)  
 Debugging (See DDT)  
 Decimal mode of KXEDIT, 8.79-80  
 Default file control block, with DDT, 8.34  
 Default drive (See current default drive)  
 Default system configuration (See system configuration)  
 Del character (same as <ctl-minus>)  
 Delete line, 10.2-3  
 DENSITY (P&T utility), 8.36-38

Density  
And CLONE, 8.17-18  
And storage capacity of diskette, 8.69(f)  
Changing, 3.5, 16.14  
Flag, 8.36  
Setting, 4.23, 4.31  
Destination drive, 4.4  
DEV: STAT option, 8.135  
Device character detect (DCD) lines, 12.2, 16.11(f)  
Diablo printer, 12.2  
DIR (CCP built-in command), 5.12-13(f)  
DIR file option, 5.9, 5.13, 8.58 (See \$DIR)  
Direct console input, 11.2  
Direct mode of KXEDIT, 8.79-80  
Directory, 3.4-6, 5.3-4(f), 5.8-9, 5.13, 5.18, 8.69, 8.131, 8.133(f)-134, 9.2  
Entries, 1.3, 5.7(f) (See extents)  
Disk/diskette, 2.2  
Changing, 3.4-6, 8.134(f) (See disk-swapping)  
Controller, 16.27(f)  
Damage, 3.3  
Errors, 9.1-5  
File attributes, 8.130  
Index hole, 16.27(f)  
Parameter Modules, 6.2, 6.5(f)-6 (See FLOPPY)  
Parameter tables, 15.1  
Permanent error, 8.46  
Space mapping upon boot, 3.4-6  
Status determination, 8.135(f), 16.26(f)  
Storage capacity, 5.2-7, 8.69(f), 8.134(f)  
Usage, 3.3-4  
Write protect notch, 3.1, 3.5, 16.27(f)  
Disk-swapping, 3.1, 3.6-7, 8.18, 8.58, 8.144-145 (See single-drive system)  
DISKCHK (P&T utility), 4.13, 8.39-45, 8.154, 8.156, 8.157(f)  
Disk system reset, 3.5-6  
DISKTEST (P&T utility), 4.38, 8.46-54, 8.159-160  
DM (Disable MENU option, MENU), 4.17  
\$ option  
  STAT, 8.132  
  SUBMIT, 8.137  
\$DIR attribute, 8.58, 8.131-132(f)  
\$R/O attribute, 8.58, 8.98, 8.131-133(f)  
\$R/W attribute, 8.58, 8.131-132(f)  
\$SYS attribute, 8.58, 8.97, 8.131-132(f)  
Double density, 1.3(f), 5.6-7(f)  
Double spacing on Radio Shack printers, 13.1-2  
  (See extra line feed)  
DP (Drive parameter, MENU), 4.18  
Drive  
  Active, 8.130  
  Designation, 5.6  
  Not Ready message, 3.7, 9.1-2, 16.27(f)  
  Parameters, 8.133(f)  
  Selection function, 15.4  
  Status information, 16.26-27(f)  
  Step rate (speed of heads), 4.18, 8.113-114, 8.117, 8.119  
  Thinline, 4.18, 8.13, 8.113  
DSK: (STAT command), 8.133, 8.135(f)  
DT (Display time, MENU), 4.20  
DUMP (DRI utility), 8.55, 8.138(f)

E option of PIP, 8.94(f)-95  
Echoing unwanted characters to console, 11.3  
ED (DRI utility), 5.20, 8.56, 8.97  
8080  
  Assembly language, 8.4-6, 8.29-35, 8.52-53  
  Chip, 1.2  
  Registers with DDT, 8.33(f)  
Emulate form feed (See form feed)  
End-of-file, 8.97 (See 1-Z)  
(enter) key, 2.1, 7.2, 11.2(f)  
EOF: option of PIP, 8.92(f), 8.99  
ERA (CCP built-in command), 5.6, 5.14, 5.18  
Erasing diskettes, 4.38, 8.46 (See DISKTEST, FORMAT)  
Erasing files, problem with lower case characters, 5.7, 5.14  
ERROR (P&T utility), 8.57, 9.3  
Error checking program, 8.157(f)-158(f)  
Error code, 9.3  
Error log, 9.1  
Error messages  
  ASM, 8.6-7  
  ASSIGN, 8.12  
  BDOS, 9.2  
  BIOS, 9.3  
  CLEAN, 8.14  
  CLONE, 8.22-26  
  DATIME, 8.28  
  DDT, 8.35  
  DENSITY, 8.37-38  
  DISKCHK, 8.43-45  
  DISKTEST, 8.51-54  
  DUMP, 8.55  
  ERROR, 8.57  
  Error file, 8.1, 8.156-157  
  FASTCOPY, 8.65-68  
  FORMAT, 8.74-77  
  Hard disk, 16.22  
  KXEDIT, 8.81  
  LOAD, 8.83  
  MENU, 4.4-5  
  Mismatched system disk, 5.20  
  MODSEL, 6.8-9  
  Module, 9.5  
  Parallel port, 13.3-4  
  PATCH, 8.87-89  
  PIP, 8.101-103  
  Programming, 16.24  
  SETDATE, 8.109-110  
  SETMISC, 8.119  
  SETTIME, 8.121  
  SUBMIT, 8.139  
  System, 9.1-5  
  System load, 9.4-5  
  TRS2CPM, 8.148  
  VERIFY, 8.150  
  Warm boot, 9.3-4  
(esc) key, 4.3-4(f), 7.2, 11.2(f)  
Escape sequence  
  Cursor positioning, 10.5  
  Key translation, 7.4  
  Program keys, 7.2  
  Terminal emulation, 7.7  
ETX/ACK protocol, 1.3, 8.124, 12.2, 16.8(f)  
EX (Exit option, MENU), 4.21  
Exchange register set, 15.1

Execution address, 8.142, 8.147  
 Execution of program or utility, 5.20  
 Extent (directory extension), 5.8, 8.131, 8.133(f)  
     (See directory)  
 Extra line feed (suppress), 4.32, 8.114, 8.118-119,  
     13.2(f), 16.11(f)  
 EXX, for swapping registers, 15.2

**F option**  
     DDT, 8.30(f), 8.33(f)  
     PIP, 8.94(f)-95  
     {f1}-{f8} keys, 3.7, 4.18-19, 6.5-6, 7.1, 9.1-2, 11.2(f),  
     13.3-4  
     Problems when programmed, 7.1, 11.5  
 FASTCOPY (P&T utility), 3.2, 4.12, 4.23-26, 5.7,  
     8.15, 8.24, 8.58-69  
 Fbase, 5.1(f)  
 FDOS, 5.1  
 File, 5.4-7  
     Attributes, 5.8-9, 8.132  
     Fragmentation, 4.12, 5.7, 8.15, 8.58  
     Names, 5.7  
     Random access, 5.8, 8.15, 8.58, 8.142  
     Statistics, 8.131 (See STAT)  
     Type, 5.5  
 File R/O error message, 9.2  
 Flashing message, programming of (See Special  
     System Functions #32-34)  
 FLOPPY (P&T module), 6.5-6(f)  
 Form feed  
     Auto, 4.32, 8.114-115, 8.118, 16.12(f)-13  
     Emulation, 4.32, 8.114-115, 8.118, 13.2(f), 16.12(f)-13  
 File, 8.151-152(f)  
 Insertion of using PIP, 8.97  
 Removal of using PIP, 8.95  
 FORMAT (P&T utility), 1.2-3(f), 3.2(f), 4.12, 4.23,  
     4.25-26, 4.31, 8.1, 8.15, 8.36-38, 8.69-77, 8.154,  
     8.159  
 Formatting a diskette, 5.2  
 FR (Freeze system parameters, MENU), 4.22, 8.9,  
     8.106, 8.111, 13.3  
 Framing error, 16.10(f)

**G option**  
     DDT, 8.30(f), 8.32-33(f), 8.35, 8.59, 8.152  
     PIP, 8.91, 8.94-95(f), 8.96(f), 8.101  
 GD (Generate data diskette, MENU), 4.23-24  
 Graphics, 7.6, 10.1-3, 10.6  
 GS (Generate system diskette, MENU), 4.25-26

**H option**  
     DDT, 8.30(f), 8.35(f)  
     PIP, 8.94(f), 8.96, 8.101-102  
 Handshaking, with serial ports, 8.123-124  
 Hard disk drive, 3.1, 3.6, 4.30  
     Access flag, 16.21-22  
     Read/write flag, 16.22  
     System, 6.1-2  
     Write protect, 16.22  
     Module, 6.2  
 HCONFIG (P&T hard-disk utility), 6.2  
 Hex  
     File, 8.4-5, 8.101, 8.147  
     Intel file, 8.4, 8.29, 8.82-83, 8.96  
     Mode with KXEDIT, 8.79-80  
 HEX extension, 8.4, 8.29, 8.82, 8.92

Hexadecimal, number code, 2.1, 4.29, 8.29, 8.79  
 Calculator with DDT, 8.30(f), 8.35(f)  
     With DUMP, 8.55  
 HFORMAT (P&T hard-disk utility), 8.74, 8.134  
     {hold} key, 11.2-4  
 HOME (BIOS routine), 15.3  
 Home cursor, 10.2-3  
 HZ (Set hertz, MENU), 4.27

**I option**  
     DDT, 8.30(f), 8.34-35(f)  
     PIP, 8.92(f), 8.96  
 INKEY\$ Basic-80 function, 11.3  
 INLINE (Pascal MT+) command, 16.3  
 INP: option of PIP, 8.92(f), 8.99-100  
 Input/Output (I/O)  
     Device assignment, 8.122, 8.130, 8.135, 14.1-3  
     Device names, 8.135(f)  
     Drivers, 1.2, 6.2, 8.1  
     Facilities, 12.3  
     Parameters, 8.111-121  
     User-supplied routine, 8.99-100  
 Integrated applications, 8.1 (See command line mode)  
 Interrupts, 15.4  
 IOBYTE, 5.10, 8.122-129, 8.135, 12.3, 14.1-3, 15.4(f)

**Keyboard**, 5.9-10, 7.4, 11.1-5, 16.2, 16.25  
 Keyboard input function, 15.4  
 Keys  
     Code, 8.78  
     Control, 2.1  
     Named, 11.2(f)  
     Programmable, 1.2, 4.9, 6.3, 7.1-4, 11.5 (See  
         AUTOKEY)  
 KEYXLATE (P&T module), 4.22, 4.28, 4.34, 6.7(f),  
     7.4-7, 8.78-81, 11.1, 11.5  
 KT (Save key translations, MENU), 4.28  
 KXEDIT (P&T utility), 4.28, 7.4, 8.78-81, 11.5

**L option**  
     DDT, 8.29-30(f), 8.31-32(f), 8.35  
     ED, 8.56  
     PIP, 8.94(f), 8.96  
 LA (Move CP/M, MENU), 4.29, 8.75  
 Line feed, 8.141, 10.2-3, 16.12-19  
     Insertion, 8.147  
     Suppress, 16.12(f)  
 Line numbers, PIP alteration of, 8.97  
 Line truncation, by PIP, 8.94  
 Line wrap, 4.15(f), 8.111-112, 8.116, 8.119, 10.2-4  
     Programming, 10.11-12  
 LIST (BIOS routine), 15.2(f)  
 List device assignment, 14.1  
 LISTST (BIOS routine), 15.3(f)  
 LOAD (DRI utility), 8.4-5(f), 8.82-83, 8.142  
 Load address, 8.29, 8.83, 8.142, 8.147-148  
 Load error messages, 9.4-5  
 Load files, 8.142, 8.147-148  
 Load operating system (or cold boot), 4.8-9, 6.1-3  
     {lock} key, 11.1  
 Lockout error, 16.22  
 Logical drive, 5.2  
 Logical extents, 5.8  
 Logical input/output devices, 8.9-11, 8.130,  
     8.134-135(f), 12.1  
 PIP, 8.91-92

- Synonyms, 8.11(f)  
Logical OR character, 11.1  
Logical sectors (128 bytes), 5.2-3, 8.82, 8.131, 8.134, 8.141  
Logical-to-physical I/O device assignment, 4.10, 5.11(f), 8.9-11, 8.122, 8.134-135  
Logical tracks, 5.3  
Logged on drive (See current default drive)  
Lower case, conversion to, 8.96, 11.1  
LPT: physical device, 5.9(f), 8.92(f), 8.135(f)  
LST: command line option  
    CLONE, 8.20(f)-21  
    DATIME, 8.27-28(f)  
    DISKCHK, 8.42(f)-43  
    DISKTEST, 8.49(f)-50  
    FASTCOPY, 8.63(f), 8.65  
    FORMAT, 8.72(f)-73  
LST: logical device, 5.10-11(f), 5.20, 7.6, 8.9-12(f), 8.19(f), 8.92(f), 8.103, 8.123(f), 8.135(f), 10.1, 10.13, 12.3, 13.1, 14.1
- M option of DDT, 8.30(f), 8.34(f)  
Macros, 8.4  
Magnetic disks, 5.2 (See disk/diskette)  
Marking state, 16.23  
Master diskette from Pickles & Trout, 3.1, 4.17  
Memory  
    Above CP/M, 4.29  
    And DDT, 8.29-35  
    And modules, 6.1, 6.8  
    Banks, 4.30  
    FASTCOPY, 8.58  
    Location 0 of, 15.3  
    Location 1 of, 15.3  
    Location 2 of, 15.3  
    Location 3 of, 14.1  
    Location 5 of, 15.4(f)  
    Page, 5.16  
    Page zero, 15.1  
    Test, 4.30  
    Usage by operating system, 5.1  
Memory map (the system parameter area), 15.4(f)  
MENU (P&T program), 4.1-38  
    Initial selection list, 4.6(f)  
    Disk and file operations, 4.1, 4.6(f)  
    System configuration options, 4.1-2, 4.6-7(f)  
    Clock options, 4.2, 4.7(f)  
MENU.COM, 4.3-4  
MENUOLY.COM, 4.3-4, 4.15  
Mismatched system diskette, error message, 5.2(f), 9.4(f)  
Model II, problem when starting, 3.4  
Model 12, 3.6, 4.30, 4.48, 8.13, 10.2(f), 11.2(f)  
Model 16, 4.18, 8.13  
Modem, 12.4  
MODSEL (P&T utility), 1.2, 4.34, 6.1, 6.3-5, 8.84, 12.1, 13.1  
Modules, 1.2, 6.1-9, 7.1-7  
    Error messages, 9.5  
    Library, 6.8-9  
    Selecting, 6.4-5  
MOVCPM, 15.1 (See MENU option LA)  
MT (Memory test, MENU), 4.30  
Multi-drive system, 3.6-7
- N option of PIP, 8.94(f), 8.97
- N2 option of PIP, 8.94(f), 8.97  
ND (New diskette, MENU), 4.23, 4.31  
NUL: option of PIP, 8.92(f)  
NULL: (or NUL:) command line option  
    CLONE, 8.20(f)  
    DATIME, 8.27-28(f)  
    DISKCHK, 8.41-42(f)  
    DISKTEST, 8.49(f)  
    FORMAT, 8.72(f)  
Null character, 11.2, 11.4, 12.2
- O option of PIP, 8.94(f), 8.97, 8.99  
Object code files, 8.82-83, 8.141 (See binary files, hex files)  
Operating system, 1.1-2  
    Functions, 10.12 (See BDOS functions)  
Overrun error, 16.10(f)  
OUT: option of PIP, 8.92(f), 8.100  
Output redirection (See command line mode)
- P command line option of VERIFY, 8.149(f)  
Page zero of memory (See System Parameter Area)  
Pagination, 13.2  
Paper length, setting of, 4.32, 8.114-115, 8.118-119  
Paper tape, 8.96  
Parallel printer port, 1.2-3, 5.9-10, 6.1, 8.84, 8.114, 13.1-4, 14.1  
    Driver, 1.2  
    Modules, 6.2 (See PPSTD)  
Parameter passing area (See special system functions)  
Parity bit, 8.98, 8.123(f), 8.125(f), 8.127-128, 12.2-3, 16.2, 16.8(f), 16.10(f)  
Printer problem if set to 1, 12.2  
Problem if set to 0, 7.5  
PASCAL MT +, 7.3-4, 16.3  
PATCH (P&T utility), 8.85-89  
PBOTH command line option  
    CLONE, 8.20-21(f)  
    DISKCHK, 8.42(f)  
    DISKTEST, 8.49-50(f)  
    FASTCOPY, 8.63-64(f)  
    FORMAT, 8.73(f)  
PC (program counter), and DDT, 8.30(f)  
Peripheral devices, 1.1  
Physical drive, 5.2  
Physical I/O devices, 5.9  
Physical sectors, 5.2  
Physical track, 5.2  
PIP (DRI utility), 2.2, 4.12, 4.23-26, 5.7, 8.15, 8.24, 8.58-59, 8.90-103, 8.130-131(f), 12.3  
Physical input/output devices, 8.9-12, 8.130, 8.134  
Assignment, 14.1  
Correspondence with P&T names, 8.92(f), 8.135(f)  
Synonyms, 8.11(f)  
+(See /filename +)  
PMOUNT command line option, 8.154  
    CLONE, 8.19-20(f)  
    DISKCHK, 8.42(f)-43  
    DISKTEST, 8.49-50(f)  
    FASTCOPY, 8.63-64(f)  
    FORMAT, 8.72-73(f)  
PNT extension, 4.26  
PP (Parallel port options, MENU), 4.32, 13.2  
PPSTD (P&T module), 6.7(f), 13.1-4  
Primary file name, 5.4-6, 5.20

PRINT Basic command, 10.7  
 PRINT@ TRSDOS Basic command, 10.7  
 Printer status, 16.11(f)  
 Printers (See form feed for top of page, paper length, line feed, extra line feed, serial port, and parallel printer port)  
 Printing the screen (See (ctl-P), SCRNDUMP)  
 PRN file, 8.4-6(f), 16.1  
 PRN: option of PIP 8.92(f)  
 Production of diskettes, 8.18, 8.154-155  
 Profile II program, 8.142(f)  
 Protocols, communications, 12.1  
 Protocols, printer, 1.3, 12.1-2  
 PSYS command line option  
     CLONE, 8.20-22(f)  
     DISKCHK, 8.42(f)  
     DISKTEST, 8.49-50(f)  
     FASTCOPY, 8.63-64(f)  
     FORMAT, 8.72-73(f)  
 PTP: physical devise, 5.9(f), 8.92(f), 8.101, 8.135(f)  
 PTR: physical device, 5.9(f), 8.92(f), 8.135(f)  
 PUN: logical device, 5.10(f), 8.9-12(f), 8.92(f), 8.99, 8.101, 8.123(f), 8.135(f), 12.3  
 PUNCH (BIOS routine), 15.3(f)  
 Punch assignment, 14.1

Q command line option  
     DISKTEXT, 8.50(f)  
     FASTCOPY, 8.64(f)-65  
     PIP, 8.94(f), 8.97, 8.102  
 ? (wildcard character), 5.5-6

R option  
     CLONE, 8.18  
     DDT, 8.30(f), 8.34-35(f)  
     DISKCHK, 8.40  
     DISKTEST, 8.47(f)-48  
     FASTCOPY, 8.60(f)-61  
     FORMAT, 8.70(f)-71  
     PIP, 8.94(f), 8.97  
 Radio Shack  
     Manual error, 13.1-4  
     Printers, 8.114, 13.1-4  
     Programs, 8.142  
 Random access files, 5.8, 8.15, 8.58, 8.142  
 RDR: logical device, 5.10(f), 8.9-12(f), 8.92(f), 8.99, 8.103, 8.123(f), 8.135(f), 12.3  
 READ (BIOS routine), 15.3(f)  
 R/O (Read-only)  
     Drive option, 3.5, 5.24-26, 16.22  
     File option, 5.8, 8.58, 8.90, 8.98, 8.131-133(f)  
     (See \$R/O)  
 Read only error message, 3.5, 9.2  
 Read/write heads, 4.14, 5.2, 5.7, 8.13, 8.113  
     Cleaning, 3.4, 4.14, 8.13-14, 8.46 (See CLEAN)  
 R/W (read/write)  
     File option, 8.58, 8.131-132  
     Drive option, 3.5, 16.22  
     (See \$R/W)  
 READER (BIOS routine), 15.3(f)  
 Reader assignment, 14.1  
 Received data pin, 12.4-5  
 Record length  
     Fixed, 8.147  
     Variable, 8.141, 8.147  
 Records per block, 8.133(f)

Records per extent, 8.133(f)  
 RECS (STAT header), 8.131 (See logical sectors)  
 Registers, 8.33(f), 10.11, 15.1-2, 15.4(f), 16.1-4, 16.7-27  
 REN (CCP built-in command), 5.12(f), 5.15  
     (repeat) key, 11.1  
 Request to send (RTS) line, 8.123(f), 8.125(f), 8.127, 12.2-3, 16.2, 16.7-8  
     Pin, 12.4-5  
 Reserved tracks, 5.3-4(f), 8.133(f) (See system tracks)  
 RESET (or cold boot), 3.1, 3.5, 5.19-20  
 Reverse line feed (Radio Shack printers), 13.3  
 Reverse video, 7.6, 10.1-3, 10.13  
 RSCOBOL, 8.141  
 RS-232 serial port, 5.9, 12.1-5  
     Connector, 8.123  
     Protocols, 1.3, 8.123-124  
     Status line, 8.124  
     Status line protocol, 12.2, 16.8(f)  
     (See serial ports)  
 RST 7 instruction, with DDT, 8.32, 8.35

S option  
     DDT, 8.30(f), 8.32(f), 8.152  
     PIP, 8.94(f), 8.97, 8.102  
 SAVE (CCP built-in command), 5.12(f), 5.16, 8.95-96(f), 8.152  
 Screen  
     Access, 16.18-19  
     Driver 10.1  
     Memory, 10.12-13, 16.18-19  
     Printing (See (ctl-P), SCRNDUMP)  
     (See Console, Video)  
 Script program, 8.142  
 SCRNDUMP (P&T module), 6.7(f), 7.6, 10.13  
 Scrolling, 11.4, 16.19  
 SD (Set date, MENU), 4.33  
 Secondary file name, 5.4-6 (See extensions)  
 Sectors per track, 5.7-8(f), 8.133(f)  
 SECTRAN (BIOS routine), 15.3(f)  
 Seek errors, and head step rate, 8.113  
 SELDSK (BIOS routine), 15.3(f)  
 Select error message, 9.2  
 Serial input/output, 12.1-5  
 Serial ports, 1.1-3, 4.4, 5.19, 8.9-12, 8.84  
     Accessing, 12.3  
     Cables, 12.4-5  
     Configuration, 8.122-129, 16.7-11  
     Connector pins, 12.4-5  
     Drivers, 6.1  
     Hardware break, 16.23  
     Input Module, 6.2 (See SIO)  
     Status, 16.2, 16.8-11  
 SETCCB (P&T utility), 4.16, 8.104-107  
 SETDATE (P&T utility), 4.33, 8.108-110, 8.140  
 SETDMA (BIOS routine), 15.3(f)  
 SETMISC (P&T utility), 4.15, 4.32, 8.106, 8.111-121, 10.11, 13.2, 16.17  
 SETSEC (BIOS routine), 15.3(f)  
 SETTIME (P&T utility), 4.36, 8.82(f), 8.121-122, 8.140  
 SETTRK (BIOS routine), 15.3(f)  
 SETUP (P&T utility), 4.35, 5.10, 8.9, 8.91, 8.122-129, 8.134, 11.2, 12.1, 12.3 13.1, 14.2 (See serial ports)  
     (shift) key, 11.1-2  
 SID program, 15.9(f)  
 Signal ground pin, 12.4-5  
 Single density diskette, 1.3(f), 3.5, 5.7-8, 8.154  
 Single drive system, 3.4, 3.6-7, 8.18-19, 8.58, 8.143

(See disk-swapping)  
**SIO** (P&T module), 12.1  
**SIOA** physical device, 5.9-11(f), 8.9-11(f), 8.123(f), 8.135(f), 14.1(f)  
**SIOB** physical device, 5.9-11(f), 8.10-11(f), 8.123(f), 8.135(f), 14.1(f)  
**/CON:** (See CON: command line option)  
**(filename)**  
**CLONE**, 8.20(f)-22  
**DATIME**, 8.28(f)  
**DISKCHK**, 8.42(f)-43  
**DISKTEST**, 8.49(f)-50  
**FASTCOPY**, 8.63(f), 8.65  
**FORMAT**, 8.72(f)-73  
**(filename) +**  
**CLONE**, 8.20(f)-22  
**DATIME**, 8.28(f)  
**DISKCHK**, 8.42(f)-43  
**DISKTEST**, 8.49(f)-50  
**FASTCOPY**, 8.63(f), 8.65  
**FORMAT**, 8.72(f)-73  
**/LST:** (See LST: command line option)  
**/NULL:** (See NULL: command line option)  
**SM** (Select module, MENU), 4.34  
**SP** (Serial port configuration, MENU), 4.35  
Soft error, 8.39, 8.46, 8.156-157  
Software production, 8.154-155  
SOROC terminal, 7.6  
Source drive, 4.4  
Source file, 8.4-5  
Special I/O functions, with PIP, 8.92(f)  
Special System Functions  
#0: set serial port parameters, 12.3, 16.7-8  
#1: read from SIOA, 12.3(f), 16.8  
#2: read from SIOB, 12.3(f), 16.9  
#3: output to SIOA, 12.3(f), 16.9  
#4: output to SIOB, 12.3(f), 16.10  
#5: read SIOA status, 12.3(f), 16.10  
#6: read SIOB status, 12.3(f), 16.11  
#7: read parallel port status, 13.3(f), 16.11  
#8: output to parallel port, 13.3(f), 16.12  
#9: set parallel port options, 13.3(f), 16.12  
#10: set page length, 13.3(f), 16.13  
#11: set number of lines per page, 13.3(f), 16.13  
#12: set top of page, 13.3(f), 16.14  
#13: set drives to unknown density, 3.5, 16.14  
#14: read real time clock, 16.15  
#15: read time of day clock, 16.6, 16.15  
#16: set time of day clock, 16.16  
#17: read xy position of cursor, 10.10(f), 16.16  
#18: read character at cursor, 10.10(f), 16.17  
#19: set size and blink of cursor, 10.10(f), 16.17-18  
#20: set cursor blink and on/off, 10.10(f), 16.18  
#21: enable access to screen memory, 10.10(f), 16.18-19  
#22: disable access to screen memory, 10.10(f), 10.12, 16.18-19  
#23: set split screen mode, 10.10(f), 10.12, 16.19  
#24: read system date, 16.6, 16.20  
#25: set system date, 16.20-21  
#26: <ctl-C> trap, 11.5, 16.21  
#27: set/clear drive access flag, 16.21-22  
#28: set/clear drive read/write flag, 16.22  
#29: hard disk function, 16.22  
#30: send break to serial port, 16.23  
#31: enable/disable terminal emulation, 7.7, 10.10(f), 16.23-24  
#32: set up flashing message, 10.10(f)-11, 16.24-26  
#33: flash message on screen, 10.10(f)-11, 16.24-26  
#34: restore screen after flashing message, 10.10(f)-11, 16.24-26  
#35: read disk status, 16.25-26  
#36: reserved, 16.27  
Split screen, 10.10, 16.10  
**ST** (System time, MENU), 4.36  
**Stack**, 15.1  
**STAT** (DRI utility), 5.8-10, 8.91, 8.130-136, 9.2, 13.4, 14.2  
**STD CORE1** (P&T module), 6.5-7(f)  
**STD CRT1** (P&T module), 6.5-7(f)  
**STD WB1** (P&T module), 6.5-7(f)  
Stop bit, 8.122-123(f), 8.125(f)-126, 12.2-3, 16.2, 16.8(f)  
Storage capacity, 1.3(f), 8.69, 8.133-134  
Strobe, line and pulse, 13.1  
SUB extension, 8.137-139  
SUB file, 8.99, 8.137-138, 8.154-155  
Circular, 8.138-139  
**SUBMIT** (DRI utility), 8.1, 8.3, 8.20, 8.39, 8.46, 8.49, 8.56, 8.69, 8.72, 8.99, 8.111, 8.122, 8.128-130, 8.137-139, 8.140, 8.151-156 (See command file)  
**SY** (Synchronize time, MENU), 4.37  
Symbol table, 8.6  
**SYNCHRO** (P&T utility), 8.108, 8.140  
SYS file attribute, 5.8-9, 5.13-14, 8.58, 8.131-132 (See \$SYS)  
**SYSGEN**, 15.1  
System clock, 15.4, 16.15-16  
**SYSTEM** command line option to copy system tracks using CLONE, 8.21(f)  
System configuration, 4.4, 4.17, 4.22  
System console, 8.9  
System date, 1.3, 4.20, 4.33, 8.108-110, 16.6, 16.20-21  
System disk, 1.4-5, 3.1-3, 3.5, 4.4, 4.25, 6.1, 6.7, 8.17, 9.3-4  
Mismatched, 6.1, 9.4  
Switching, 5.19  
System files (list), 1.4-5  
System generation procedure, 1.5, 3.1-3  
System Parameter Area (SPA or page zero), 5.1, 15.1  
System Parameters  
Drive, 8.133  
Freezing, 4.22  
Passing, 15.4  
Serial port, 12.1-5  
Substitution, 8.137, 8.139  
System statistics (See STAT)  
System time, 4.20, 8.121-122, 15.4, 16.6  
System tracks, 5.3-4(f), 5.18, 6.1, 8.38, 8.134  
Copying, 8.15-17(f)

**T** option  
**CLONE**, 8.21-22, 8.155  
**DDT**, 8.29-30(f), 8.34  
**ED**, 8.56  
**PIP**, 8.94(f), 8.98  
**Tab**, 8.98, 10.2-3  
<tab> key, 7.2-3, 7.5, 11.2(f)  
**Tbase**, 5.1(f)  
**TE** (Test diskette, MENU), 4.33  
Terminal, 8.9-12, 10.1, 12.4-5(f)  
Emulation, 6.3, 7.6-7, 10.1, 10.10, 16.23-24  
Texas Instrument's 820 printer, 8.125(f)  
Text file, 5.4  
Tilde character, 11.1  
Time, system, 15.4 (See SETTIME, SYNCHRO)

- Timing an activity, 16.15  
 Top of page, setting of, 13.2, 16.14  
 Track, 5.2  
**Transient Program Area (TPA)**, 5.1, 5.12, 5.16, 8.29, 8.152  
 Transient programs, 5.12, 5.19, 8.82, 15.4(f)  
 Transmitted data pin, 12.4-5  
**TRSDOS**, 5.23, 8.114, 8.124, 8.141-148, 12.2  
**TRSDOS Basic**, 8.141-142, 10.7-8  
**TRS2CPM** (P&T utility), 8.141-148  
**TTY**: physical device, 5.9(f), 8.92(f), 8.135(f)  
**TYPE** (CCP built-in command), 5.12(f), 5.17  
 Type-ahead buffer, 1.3, 8.99, 11.3-4
- U option**  
 DDT, 8.30(f), 8.34(f)  
 ED, 8.56  
 PIP, 8.94(f), 8.98  
**UC1**: physical device, 5.9(f), 8.92(f), 8.135(f)  
**Ufn** (See unique file name)  
**UL1**: physical device, 5.9(f), 8.92(f), 8.135(f)  
 Unique file name (unqfn), 5.6, 5.13(f), 5.15-17, 8.91, 8.149  
**UP1**: physical device, 5.9(f)  
**UP2**: physical device, 5.9(f)  
 Upper case, conversion to, 8.96-98  
**UPUN1**: physical device, 5.9-10(f), 8.10-11(f), 8.92(f), 8.123(f), 8.135(f)  
**UPUN2**: physical device, 5.9-10(f), 8.10-11(f), 8.92(f), 8.123(f), 8.135(f)  
**UR1**: physical device, 5.9(f)  
**UR2**: physical device, 5.9(f)  
**URDR1**: physical device, 5.9-10(f), 8.10-11(f), 8.92(f), 8.123(f), 8.135(f)  
**URDR2**: physical device, 5.9-10(f), 8.10-11(f), 8.92(f), 8.123(f), 8.135(f)  
**USER** (CCP built-in command), 5.12(f), 5.18, 8.95-96(f)  
**USER(S)** command line option of FASTCOPY, 8.59-60, 8.62, 8.64-65  
 User area, 8.59-65, 8.130  
 User number, 4.25-26, 5.17, 8.95, 8.134, 15.4(f)  
**USR**: (STAT option), 8.130, 8.134-135(f)  
 Utility Modules, 6.2 (See AUTOKEY, KEYSLATE, SCRNDUMP)
- V command line option**  
 CLONE, 8.15-16(f), 8.21-22, 8.155  
 FASTCOPY, 8.58, 8.60(f)-61  
 FORMAT, 8.69, 8.71, 8.154  
 PIP, 8.93(f)-94(f), 8.98, 8.101  
**VAL**: (STAT option), 8.135-136  
**VERIFY** (P&T utility), 8.149-150  
**Video**  
 Control codes, 10.1-4  
 Drivers, 16.18  
**Video display**, 1.1, 14.2  
 Diagram, 10.5(f)  
 Handler, 10.1-13  
 Memory mapped, 16.18  
**Video memory**, access to, 10.12-13, 16.18-19
- W option of PIP**, 8.90, 8.94(f), 8.98  
 Warm boot, 3.5-7, 4.8, 5.14-15, 5.19-20, 6.1-2, 8.133, 9.1-5, 11.4-5, 13.4  
 Error message, 9.3-4
- Modules, 6.2 (See STDWB1)  
**WBOOT** (BIOS routine), 15.3(f)  
**WIDTH** Basic command, 10.7  
 Wildcard file name (wcfm or afn), 5.5-6, 5.13(f)-15, 8.59, 8.91, 8.93, 8.102, 8.132, 8.149-150  
 Word length, 8.123(f), 8.125(f), 8.128, 12.2-3, 16.2, 16.8(f)  
 Working system disk (See system disk)  
**WRITE** (BIOS routine), 15.3(f)
- X option of DDT**, 8.29-30(f), 8.33(f)  
**Xmit-off**, 8.124, 12.2, 16.8(f)  
**Xmit-on**, 8.123-124, 12.2, 16.8(f)  
**XON/XOFF protocol**, 1.3, 8.123-128, 12.1-2, 16.8(f)  
 Problem if parity bit set to 1, 12.2  
**XSUB** (DRI utility), 8.138, 8.151-153
- Z option of ED**, 8.56  
**Z command line option**  
 FASTCOPY, 8.64(f)  
 PIP, 8.94(f), 8.98  
**Z-80**  
 Chip, 1.2  
 Code, 8.4, 8.29, 8.35  
 Index registers, 15.1  
 Zero-length files, 8.58-60(f), 8.64

# Pickles & Trout®

## End User Software License Agreement

All Pickles & Trout 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 in this license, return the packaged diskette unopened with the documentation to your dealer and the purchase price will be refunded. If you agree to the terms contained in this license, fill out the registration card completely and return it by mail to Pickles & Trout. Returning the completed registration card licenses you, the user, as the authorized user of the product as well as entitles you to receive announcements of software updates and the newsletter C:What's.New™.

Breaking the seal on the packaged diskette and/or use of the product constitutes acceptance of the terms and conditions set forth in this license. Purchaser recognizes that any use of Pickles & Trout products without the return of the registration card will be considered a breach of contract and is the unlawful and unauthorized use of Pickles & Trout's trade secrets and proprietary products.

**PICKLES & TROUT AGREES TO GRANT AND THE END USER AGREES TO ACCEPT ON THE FOLLOWING TERMS AND CONDITIONS A NON-TRANSFERRABLE AND NON-EXCLUSIVE LICENSE TO USE THE SOFTWARE HEREIN DELIVERED WITH THIS AGREEMENT.**

### DEFINITIONS

**Pickles & Trout** means Pickles & Trout, a California corporation, P.O. Box 1206, Goleta, California 93116, United States of America, the author and owner of the copyright of the BIOS implementation and reformat of CP/M 2.2 with its supporting utility programs, acting on behalf of Digital Research Inc., PO Box 597, Pacific Grove, California 93950, United States of America, the author and owner of the copyright of CP/M 2.2 BDOS and CCP and the owner of the registered trademark CP/M, and acting on behalf of CompuView Products, Inc., 618 Louise, Ann Arbor, Michigan 48103, United States of America, the author and owner of the copyright of VEDIT.

**End User** means the individual purchaser and the company End User works for, if the company paid for this software.

**Software** is the set of computer programs and manuals in this package, regardless of the form in which End User may subsequently use it, and regardless of any modification which End User may make to it.

**License** means this agreement and the rights and obligations which it creates under the United States copyright law and California laws.

### LICENSE

Pickles & Trout grants End User the right to use this serialized copy of the Software on a single computer at a single location. A separate license is required for each computer on which the Software will be used.

This agreement and any of the licenses, Software, or materials to which it applies may not be assigned, sublicensed, or otherwise transferred by the End User without prior written consent from Pickles & Trout. No right to print or copy, in whole or in part, the Software is granted except as hereinafter expressly provided.

### PERMISSION TO COPY LICENSED PROGRAMS

The End User shall not copy, in whole or in part, any Software which is provided by Pickles & Trout in printed form under this agreement. Additional copies of the printed materials may be purchased from Pickles & Trout.

Any Software which is provided by Pickles & Trout in machine readable form may be copied, in whole or in part, for back-up purposes, provided, however, that no more than five (5) copies are in existence under any license at any one time without prior written consent from Pickles & Trout. The End User agrees to reproduce and affix to the outside of each backup copy diskette a copy of the entire label attached to the original diskette, including copyright and trademark information, program name, version number, and serial number. The End User also agrees to maintain appropriate records of the number and locations of all such copies of Software. The original, and any copies of the Software, in whole or in part, which are made by the End User shall be the property of Pickles & Trout. This does not, of course, imply that Pickles & Trout owns the media on which the Software is recorded.

### TRANSFER OR REPRODUCTION

End User understands that unauthorized reproduction of copies of the Software and/or unauthorized transfer of any copy may be a serious crime, as well as subjecting End User to damages and attorney fees. End User may not transfer any copy of the Software to another person unless End User transfers all copies, including the original, and advises Pickles & Trout of the name and address of that person, who must sign a copy of the registration card, pay the then current transfer fee, and agree to the terms of this License in order to use the Software. Pickles & Trout will provide additional copies of the card and License upon request. Pickles & Trout has the right to terminate the License, to trace serial numbers, and to take legal action if these conditions are violated.

## **PROTECTION AND SECURITY**

The End User agrees not to provide or otherwise make available any part of this Software including but not limited to program listings, object code, source code, in any form, to any person other than End User or Pickles & Trout employees, without prior written consent from Pickles & Trout.

## **TERM & DISCONTINUANCE**

This License is effective from the date of receipt of the Software and shall remain in force until terminated by End User upon one (1) month's written notice, or by Pickles & Trout as provided below. License may be discontinued by End User at any time upon one (1) month's prior written notice. Pickles & Trout may discontinue or terminate this License if the End User fails to comply with any of its terms and conditions.

Within one (1) month after the date of discontinuance of the License, the End User will furnish Pickles & Trout a certificate certifying that through his best effort, and to the best of his knowledge, the original and all copies, in whole or in part, in any form, of the Software have been destroyed, except that, on prior written authorization from Pickles & Trout, the End User may retain a copy for archival purposes.

## **LIMITED WARRANTY**

The only warranty Pickles & Trout makes is that the diskette(s) on which the Software is recorded will be replaced without charge, if Pickles & Trout in good faith determines that the media was defective and not subject to misuse, and that the diskette is returned with a copy of the original registration card within 30 days of the purchase.

Software and manuals are sold "as is" and Pickles & Trout makes no warranties with respect to the Software contained therein. The sole obligation of Pickles & Trout shall be to make available all published modifications made by Pickles & Trout to Software which are published within one (1) year from date of purchase, provided End User has completed the registration card delivered with the Software and returned it to Pickles & Trout offices.

The foregoing warranty is in lieu of all other warranties, expressed or implied, including, but not limited to, the implied warranties of performance, merchantability or fitness for a particular purpose. In no event will Pickles & Trout be liable for indirect, special or consequential damages such as loss of profits or inability to use the software. The entire risk as to the results and performance of this Software is assumed by the End User.

Some states may not allow this disclaimer so this language may not apply to End User. In such case, our liability shall be limited to a refund of the Pickles & Trout list price. End User may have other rights which vary from state to state. End User and Pickles & Trout agree that this product is not intended as "consumer goods" under state or federal warranty laws.

## **GENERAL**

This License is governed by the laws of the State of California and is deemed entered into in Goleta, Santa Barbara, California, United States of America, by both parties.

If any of the provisions, or portions thereof, of this License are invalid under any applicable statute or rule of law, they are to that extent to be deemed omitted. This License will then be construed as if such unenforceable provision or provisions had never been contained herein.

By signing and returning the registration card delivered with the Software, the End User hereby accepts all the terms and conditions of the License without exception, deletion, or alteration. This License is the complete and exclusive statement of the agreement between End User and Pickles & Trout, which supercedes any proposal or prior license, written or oral, and other communications between us relating to the subject matter of this License.

**PICKLES & TROUT®**

**P.O. BOX 1206, GOLETA, CA 93116**