DIGITAL RESEARCH

# CP/M®-85

593-0059
CONSISTS OF

MANUAL
595-2824
FLYSHEET
597-2895

Printed in the
United States of America

ZENITH data systems

HEATH

**NOTICE**

This software is licensed (not sold). It is licensed to sublicensees, including end-users, without either express or implied warranties of any kind on an "as is" basis.

The owner and distributors make no express or implied warranties to sublicensees, including end-users, with regard to this software, including merchantability, fitness for any purpose or non-infringement of patents, copyrights or other proprietary rights of others. Neither of them shall have any liability or responsibility to sublicensees, including end-users, for damages of any kind, including special, indirect or consequential damages, arising out of or resulting from any program, services or materials made available hereunder or the use or modification thereof.

Technical consultation is available for any problems you encounter in verifying the proper operation of these products. Sorry, but we are not able to evaluate or assist in the debugging of any programs you may develop. For technical assistance, call:

(616) 982-3884 Application Software Softstuff Products
(616) 982-3860 Operating System Language Software/Utilites

Consultation is available from 8:00 AM to 4:30 PM (Eastern Time Zone) on regular business days.

Zenith Data Systems
Software Consultation
Hilltop Road
St. Joseph, Michigan 49085



HEATH COMPANY
BENTON HARBOR, MICHIGAN 49022

ZENITH DATA SYSTEMS
ST. JOSEPH, MICHIGAN 49085

# Preface

CP/M* is an acronym for the Control Program for Microcomputers, the operating system that has quickly become the industry standard because of its convenience and wide range of practical applications.

This manual is designed to help you to make CP/M productive in your microcomputer environment. To provide instruction for CP/M users with various levels of microcomputer experience, this manual is divided into two guides. Elsewhere you may see references to Volume I and II. These numbers refer to these two guides:

- "The CP/M Introductory Guide", explains beginning concepts necessary to use CP/M with Heath/Zenith hardware, and provides step-by-step procedures for starting up and preparing CP/M software.

- "The CP/M Reference Guide" is a comprehensive description of each command included in your CP/M software package.

If you are not familiar with CP/M practices and conventions, you will probably find it useful to first read the overview and concept sections of the first guide. Even individuals with no experience in computers will be able to understand and use CP/M by reading this portion of the manual.

All users should perform the procedures in the CP/M Introductory Guide upon receipt of CP/M Distribution Disks.

*CP/M is a registered trademark of Digital Research Corp.

When you finish performing the procedures in the first guide, you will be ready to use CP/M with the application programs of your choice. But we urge you to make the most of your CP/M software package by exploring the reference guide to CP/M resident commands and transient commands.

This manual will be more effective throughout if you read it while you are using your microcomputer. Test each concept through your terminal to reinforce your learning.

A second, separate volume contains a system interface guide, an alteration guide, and appropriate appendices. These reprints from Digital Research are general in nature and references to CP/M-85 and specific computers may not be accurate. Since the source code listed in these reprints was not used to create CP/M-85, refer to Distribution Disk II for the actual source code.

**Essential Requirements**

Use of CP/M-85 Version 2.2.100 is limited to the following specifications:

1. Distribution Media — Two, soft-sectored, 5.25 inch, 48-TPI, floppy disks at double density.

2. Disk Format — Soft-sectored, 5.25-inch, 48-TPI floppy disks at double density; or soft-sectored, 8-inch, 48-TPI floppy disks.

3. Minimum Hardware — Z-100 or H-100 microcomputer with attached or separate video screen.

# Introductory Guide

# Table of Contents

### THE CP/M-85 INTRODUCTORY GUIDE



### THE CP/M-85 REFERENCE GUIDE

Disk error status codes
8" Diskette Step Rate
Memory Checking

*Section One*

# Beginning Concepts

This part of the manual explains concepts that are important when you are using the CP/M Operating System within your Heath/Zenith microcomputer environment.

The concepts explained here are grouped under the headings:

- "Microcomputer Concepts:" Provided for individuals who have never before used microcomputers, and for individuals with limited microcomputer experience who wish to review.

- "CP/M Concepts:" Provided for individuals who are unfamiliar with the CP/M Operating System.

Individuals who are proficient at using both microcomputers and CP/M can skip this section of the manual, and proceed directly to "Software Preparation Procedures" on Page 1-27.

# Examples of User/Computer Dialog

This text contains examples of user interaction with a microcomputer. In these examples, displays presented on the microcomputer terminal will be represented by the following typestyle:

```
THIS TYPESTYLE represents terminal displays

0123456789#$*?:=.A>()
```

User input (the characters that you type through the terminal) will be represented by boldface type, as shown:

**BOLDFACE TYPE represents the things you type**

**0123456789#$*?:=[.]()**

When you should enter a carriage return by pressing the key marked "RETURN," the example will show the symbol ↵ .

In many instances, the exact text of a display will vary by a few characters. This manual often substitutes a few letters in place of exact characters where variations are likely to occur. For instance, this manual will illustrate a program's serial number as "Serial number sss-sssss," while your terminal might actually display it as "Serial number 357-81469."

In cases where the exact characters you type will vary, this manual presents a description of the necessary characters within curved braces, { }.

Hence, this manual might explain that an entry should be made in the following form: **B:=A:{filename.ext}** ↵, when you actually type the characters **B:=A:CONFIGUR.COM** ↵ .

Hardware device model numbers beginning with the "H/Z" prefix are references to either a Heath device, a Zenith device, or both. For example "H/Z-100" in this manual refers to hardware devices that are labeled either "H-100" or "Z-100"

# MICROCOMPUTER CONCEPTS

Your microcomputer is a sophisticated piece of equipment that reflects the latest technical advances in the computer field. But this machine is practically useless without the programs (instructions) that tell it what to do.

These programs are stored on disks, and used by your microcomputer when they are required to perform a task. This section explains how the programs are stored on disks, shows you how to handle your disks, and defines two important types of program: the operating system and the application program.

# Disks

Stored information, or data, is arranged in concentric rings on the surface of a disk. These rings are called "tracks." Each track is divided into areas called "sectors." Each sector contains data measured in units called "bytes." A byte of data could be one letter typed at the terminal keyboard or one instruction in a program. But since a byte is such a small unit, you will more often see data measured in "kilobytes." A kilobyte (abbreviated as "K") is equal to 1024 bytes.

Data is transferred to disks in the form of magnetic impulses generated by an electromagnet called the "read-write head." As the name implies, the read-write head can read data from the disk or write data on the disk — similar to the way in which the head in a tape recorder transfers magnetic impulses to and from a cassette tape.

But unlike a tape recorder, a disk drive unit can transfer data at any location on the disk surface almost instantly, because the drive is usually spinning the disk at a high rate of speed. Whenever the read-write head is instructed to read or write data at a particular location on the disk, it positions itself along the appropriate track and skims across the surface of the disk as the appropriate sector spins by. Each disk has a directory that tells the read-write head which track and sector it should access to transfer the necessary information in the proper sequence.

There are two different kinds of tracks on every disk: system tracks and file tracks. System tracks are reserved for part of your operating system, and they are usually the two or three outermost tracks on the disk. File tracks are reserved for files, and they are the inner tracks on the disk. (Upcoming text will explain the concepts behind operating systems and files.)

NOTE: Before data can be written on a disk, the surface of the disk must be prepared. Disk preparation is performed by a program called "FORMAT," which is stored on your Distribution Disk. FORMAT prepares the disk surface by dividing it into tracks and sectors. The procedures for constructing backups and customizing the operating system will show you how to prepare disks using FORMAT.

## FLOPPY DISKS

A floppy disk is a circular sheet of mylar plastic with a magnetic oxide on its surface and a square plastic cover.

Floppy disks, and the data stored on them, are fragile. Therefore, you should adhere to the following precautions to ensure that disks and stored data are not damaged.

### Floppy Disk Handling Precautions

- When you are holding the disk, touch only the protective square disk cover. Do not touch the brownish disk surface that shows through the read-write access slots in the disk cover.

- Keep the disk in the protective paper envelope whenever it is not within a disk drive.

- Do not allow dust, ashes, liquid, or any other foreign material to contact the disk surface.

- Keep the disk away from electric motors, appliances, telephones, etc., as these devices contain magnets that could alter the magnetic impressions on the disk.

- Never put a disk into a drive unit before turning on hardware equipment; and never leave a disk in a drive unit while the power is being turned off. Sudden fluctuations in the power supply to your hardware environment could cause the read-write head to write on the disk surface, and destroy stored information.

- Do not expose disks to temperatures above 125 degrees Fahrenheit (52 Centigrade), or temperatures below 40 degrees Fahrenheit (10 Centigrade).

- Never press a ball-point pen or a pencil directly against the cover of a disk. Instead, mark disk labels before adhering them to the disk cover, or mark them using a felt-tip pen while they are on the disk cover.

- Do not allow the disk or its cover to be bent, creased, or torn.

- Do not attach paper clips to the disk.

### Write-Protecting and Write-Enabling Floppy Disks

You can mechanically prevent or allow the writing or erasing of information to or from your disks by covering or uncovering the notch in the disk cover with specially-provided tabs. The way that you use these tabs depends on the size of the disk.

#### 5.25-inch Disks

With 5.25-inch disks, the notch is covered to **prevent** you from writing to or erasing from the disk. Therefore, by putting the tab on a 5.25-inch disk, you are "write-protecting" the disk. A 5.25-inch disk with a notch that is not covered can be written to or erased from. Therefore, a 5.25-inch disk with an uncovered notch is "write-enabled." Figure 1-1 illustrates this distinction.

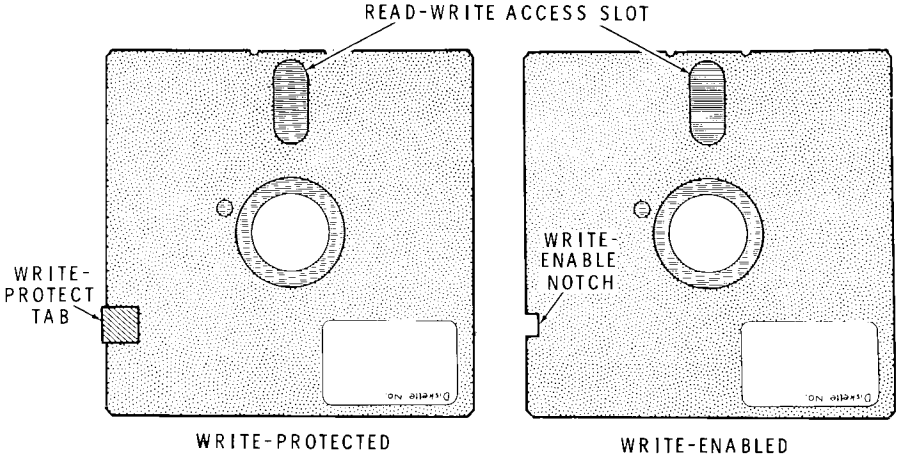



**Figure 1-1**
5.25-inch Floppy Disks

## 8-inch Disks

With 8-inch disks, the notch is covered to **allow** you to write to or erase from the disk. Therefore, by putting the tab on an 8-inch disk, you are "write-enabling" the disk. An 8-inch disk with a notch that is not covered can not be written to or erased from. Therefore, an 8-inch disk with an uncovered notch is "write-protected." Figure 1-2 illustrates this distinction.

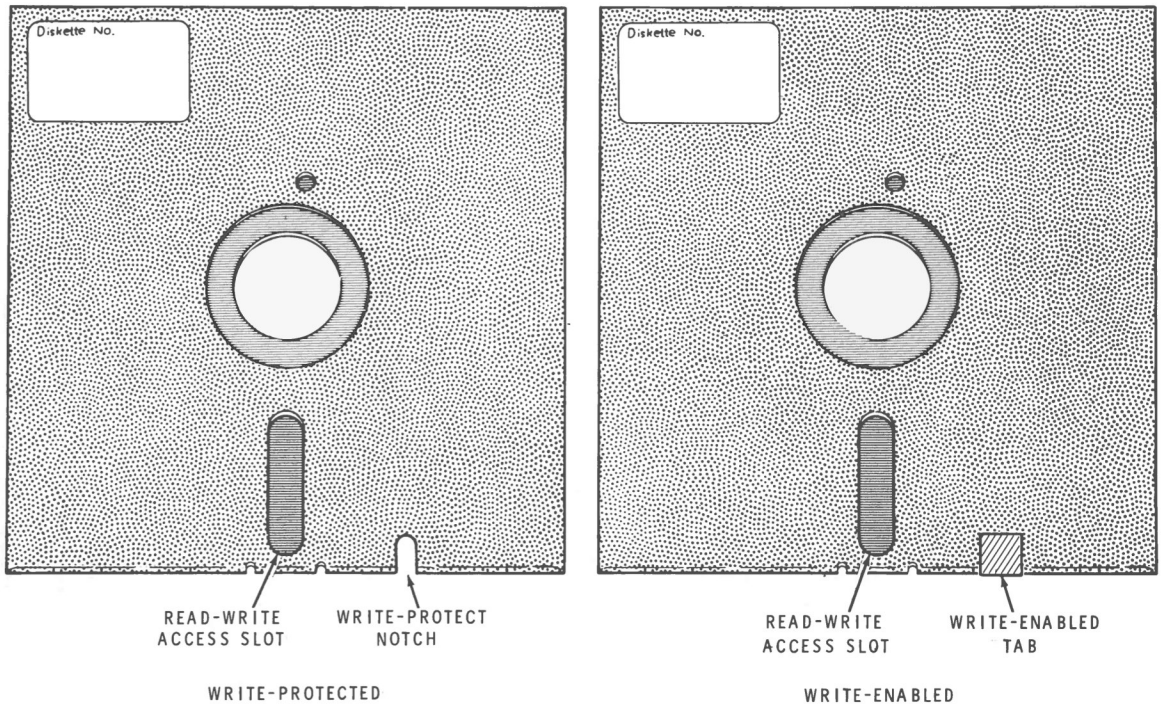



**Figure 1-2**
8-inch Floppy Disks

NOTE: Whether a disk is write-protected or write-enabled, you can usually read or copy data from it.

### Inserting Floppy Disks

To insure that data stored on your floppy disks can be safely and efficiently accessed, the disks must be inserted into drives carefully and correctly.

When you insert a floppy disk into a disk drive, point the oblong holes in the square plastic cover towards the back of the drive. A label is usually affixed to one side of the plastic disk cover, and this label should face upward on the right side as the disk is inserted into the drive. When the disk is fully inserted in the drive slot, close the drive latch.

Figure 1-3 illustrates the proper technique for inserting floppy disks into some of the many drives that Heath/Zenith offers.

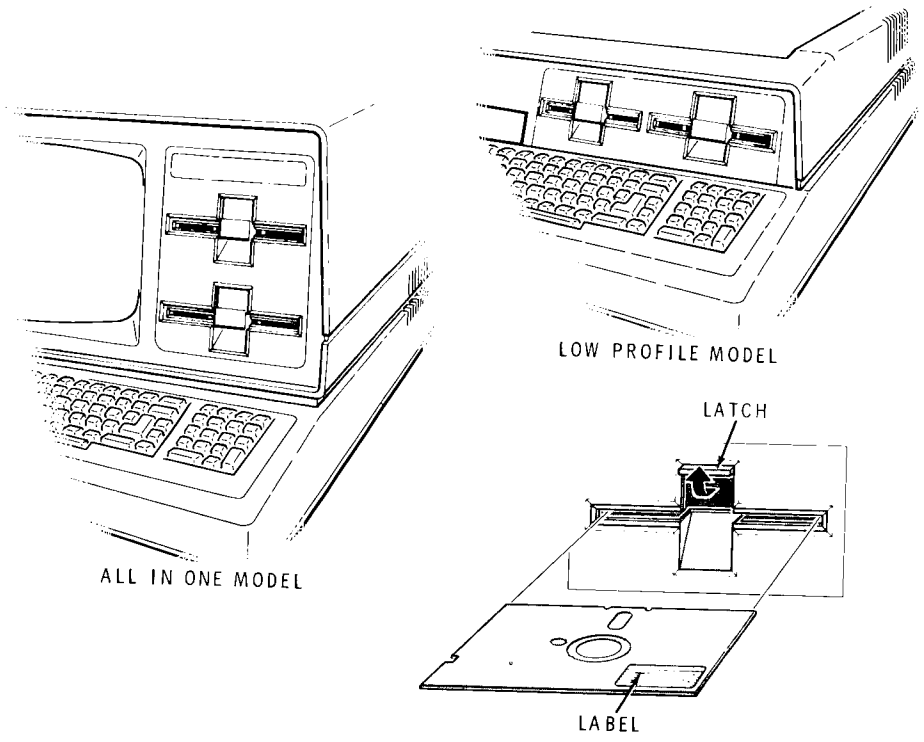



**Figure 1-3**
Inserting Floppy Disks

# The Operating System

An operating system is a computer program that controls both the components of your hardware environment and subordinate programs ("application programs") that perform specific tasks.

It provides a vital link between your keyboard and your application programs, and between your application programs and your peripheral hardware. Thus it is essential that you use an operating system whenever you use an application program in your microcomputer environment.

The way that you use the CP/M Operating System is to copy an image of the system from a disk, and put it inside the memory of your microcomputer. This activity (called "bootstrapping," "booting up," or "cold booting") occurs automatically when you power up the computer and insert a disk containing the CP/M Operating System.

CP/M can perform several useful functions when it is alone in your computer's memory, but it also has "hollow" areas into which you can load application programs as you need them.

## OPERATING SYSTEM COMPONENTS

The Heath/Zenith CP/M operating system is divided into two software components: the "system kernel" and the "BIOS files." A disk that contains both of these components is said to contain the CP/M Operating System. Such a disk can also be said to be "bootable", which means that it contains an image of the operating system that can be inserted into the computer and used.

### The System Kernel

The "system kernel" is a set of programs that reside on the reserved system tracks of a disk. Special data transfer utilities (usually SYSGEN and/or MVCPM207) are used to copy the system kernel from one disk to another. The system kernel manages files, translates the commands you enter at the keyboard, and performs other functions that do not depend upon specific hardware characteristics.

### The BIOS Files

"BIOS" stands for Basic Input/Output System, which is the part of the operating system that enables CP/M to work in a Heath/Zenith hardware environment. The BIOS used with this CP/M release consists of the files "BIOS85.SYS" and "BIOS88.SYS", which reside on the file tracks of the disk. The BIOS files can be manipulated by the same methods used to manipulate other files that are stored on the disk. The portion of the BIOS stored in the file BIOS85.SYS uses the 8085 processor of the Z-100. The portion stored in BIOS88.SYS uses the 8088 processor of the Z-100.

### Operating System Requirements

To use the operating system, you must transfer a copy of it from a disk to the computer by performing an activity called "bootstrapping" or "booting-up," or "cold booting." Both the system kernel and the BIOS files **must** be together on a disk for that disk to be usable for bootstrapping (the activity that puts CP/M in the computer).

To protect your software investment, we strongly suggest that you make backup copies of your CP/M distribution software before using it for any practical applications. Once you have made backup copies, your CP/M software will be ready for use in a Heath/Zenith hardware environment. "Section Two: Software Preparation Procedures" (on Page 1-27) shows you how to start up and back up your CP/M software.

The CP/M Operating System on your distribution software is ready to accommodate a standard configuration of hardware devices. If your hardware configuration differs from these standards, you must adjust the system before it will accommodate all of your devices. Your distribution software includes utilities to help you adjust the system for your devices.

Your CP/M-85 distribution software contains a serial number to prevent the mixing of this software with software from other CP/M distribution packages. Therefore, do not try to use this CP/M system with other CP/M systems or utilities, and do not try to use these utilities with other CP/M systems.

# Application Programs

An application program is a set of instructions that tells your microcomputer how to perform a specific function.

Application programs are stored on the file tracks of a disk in units of data called files (as explained in "Files", Page 1-13). An application program might consist of several files that automatically access each other under certain circumstances. Whenever an application program file is needed to perform a specific task, an image of this file is copied from the disk and inserted into computer memory, where the CP/M Operating System has a reserved "hollow" area for it.

After the application program files have served their purpose, CP/M moves them aside and either reserves its hollow memory areas for new application programs or executes some of the programs within the operating system itself.

**Examples of Application Programs**

Your Distribution Disk contains several application programs, which are often referred to as "transient commands" or "utilities." These programs, stored on the Distribution Disk, are identified by names that end with the file extension "COM." This extension indicates that these files are valid COMmands that can execute under CP/M.

The following list shows some of the types of application programs, languages, and utilities which are or shortly will become available from Heath/Zenith. Refer to your User's Manual for a list of existing CP/M software which may be converted to run under CP/M-85 on the Z-100.

BUSINESS:

- Electronic Spread Sheet
- Inventory Management
- General Ledger
- Accounts Receivable
- Accounts Payable
- Sales Invoicing
- Payroll
- Client Posting & Accounting
- Property Management
- Word Processing
- Electronic Dictionary
- Mail List Management
- Data-based Manager
- Automatic Letter Generator

Same as following page

PROGRAMMING LANGUAGES:

- BASIC-80 Interpreter
- BASIC-80 Compiler
- C-BASIC
- FORTRAN-80
- COBOL-80

SYSTEM UTILITIES:

- Macro Assembler
- Printerspooling operation
- Symbolic Debugger

## APPLICATION PROGRAM REQUIREMENTS

It is important to remember that an application program cannot produce the results you desire without the presence of an operating system. Therefore, you must always load the CP/M operating system into your microcomputer before you can use any application program. ("Start-Up Procedure" will show you how to put CP/M in your computer.)

Furthermore, the CP/M Operating System must be capable of controlling all of your hardware devices before any application program can use these devices. Sometimes you must adjust the system (as described in "Backup Procedures", Page 1-32) so it can control your hardware devices.

Each application program consists of at least one file with a .COM file name extension. Some application programs also consist of additional files. All of the files included in an application program must be present on a disk within a drive whenever you wish to use the application program.

# Application Programs

An application program is a set of instructions that tells your microcomputer how to perform a specific function.

Application programs are stored on the file tracks of a disk in units of data called files (as explained in "Files", Page 1-13). An application program might consist of several files that automatically access each other under certain circumstances. Whenever an application program file is needed to perform a specific task, an image of this file is copied from the disk and inserted into computer memory, where the CP/M Operating System has a reserved "hollow" area for it.

After the application program files have served their purpose, CP/M moves them aside and either reserves its hollow memory areas for new application programs or executes some of the programs within the operating system itself.

**Examples of Application Programs**

Your Distribution Disk contains several application programs, which are often referred to as "transient commands" or "utilities." These programs, stored on the Distribution Disk, are identified by names that end with the file extension "COM." This extension indicates that these files are valid COMmands that can execute under CP/M.

The following list shows just some of the other application programs available through Heath/Zenith. All programs in this list perform specific tasks when controlled by the CP/M Operating System.

BUSINESS:

- Electronic Spread Sheet
- Inventory Management
- General Ledger
- Accounts Receivable
- Accounts Payable
- Sales Invoicing
- Payroll
- Client Posting & Accounting
- Property Management
- WordStar*
- MailMerge*
- SpellStar*
- Magic Wand*
- Mailpro*
- DataStar*
- SuperSort*

PROGRAMMING LANGUAGES:

- BASIC-80 Interpreter
- BASIC-80 Compiler
- C-BASIC
- FORTRAN-80
- COBOL-80

SYSTEM UTILITIES:

- MAC (Macro Assembler)
- DESPOOL (Printer Operation)
- SID/ZSID (Symbolic Debugger)
- CPS (Computerized Phone System)
- RTTY (Radio Teletype)

* Registered Trademark

## APPLICATION PROGRAM REQUIREMENTS

It is important to remember that an application program cannot produce the results you desire without the presence of an operating system. Therefore, you must always load the CP/M operating system into your microcomputer before you can use any application program. ("Start-Up Procedure" will show you how to put CP/M in your computer.)

Furthermore, the CP/M Operating System must be capable of controlling all of your hardware devices before any application program can use these devices. Sometimes you must adjust the system (as described in "Backup Procedures", Page 1-32) so it can control your hardware devices.

Each application program consists of at least one file with a .COM file name extension. Some application programs also consist of additional files. All of the files included in an application program must be present on a disk within a drive whenever you wish to use the application program.

# CP/M CONCEPTS

CP/M Concepts explains some of the basic properties of the CP/M Operating System and some of the conventions you must follow when using the system. After reading "Microcomputer Concepts" and "CP/M Concepts" you should be ready to perform the Software Preparation Procedures.

Specifically, this text shows you how CP/M stores data in units called files, how CP/M accesses these files through disk drives, and how you should issue commands to the CP/M Operating System.

## Files

The CP/M Operating System enables you to create, analyze, and manipulate data by storing this data in units called "files." These files are stored on the surface of a disk and given names that conform to CP/M file naming conventions.

When you issue a command that refers to a particular file by name, CP/M goes to the appropriate disk, makes a copy of the file, and puts the copy into one of its hollow areas inside the computer.

### CP/M FILE NAMING CONVENTIONS

A file name consists of two parts: the primary name and the extension. The primary name has between one and eight characters, and is essential in all file names. The extension can have between one and three characters, or it can be omitted entirely. The primary name and the extension are separated by a period (.), in the following form:

{**primary name**}.{**extension**}

The characters used in the primary name and extension can be any character on the console keyboard except the following special characters:

<>.,;: =?*[]

The following example file names all conform to these conventions:

memo.doc BIOS85.SYS PROGRAM.ASM 4/27/81.TXT

FORMAT.COM FILE#1 33%-RATE.DAT WSMSGS.OVR

Although the extension is optional, you'll probably find it useful to give your files extensions that somehow describe the type or purpose of the file you are naming. The following list shows several extensions that are applied to files used for a specific purpose:

| EXTENSION | FILE PURPOSE |
|---|---|
| COM | Command file (executable under CP/M) |
| ASM | Assembler source file |
| HEX | Intel HEX object file |
| PRN | Print file of program listing |
| BAS | BASIC source file |
| INT | Intermediate BASIC file (for BASIC-E, CBASIC) |
| BAK | Backup file |
| LIB | Library source file |
| SUB | A list of commands to be executed with SUBMIT |
| $$$ | Temporary work file |
| FOR | FORTRAN source file |
| DAT | ASCII data file |
| DOC | ASCII document file |

## REFERENCING SEVERAL FILES AT ONCE (WILDCARD FILE NAMES)

Many of the commands that you will issue refer to files by name. But when you want to issue the same command for several files with similar names, it is often more convenient to enter a "wildcard file name" with the command.

A "wildcard file name" represents several file names — much like a "joker" in a deck of cards can stand for any card in the deck. (Wildcard file names are sometimes called "ambiguous file names".) A wildcard file name contains either asterisks (*), or question marks (?), or both.

An asterisk (*) is entered in place of the entire primary name and/or extension of a file, as shown in the following example:

16MAY82.*

In this example, the asterisk replaces the extension so that this wildcard refers to files on the disk with the primary name "16MAY82" and **any** extension.

In the following example, the wildcard asterisk takes the place of the primary name:

*.COM

Therefore, all files with the extension "COM" and **any** primary name would be referenced.

The wildcard "*.*" stands for **any** file on the disk. (Actually, there are exceptional circumstances under which all of the files on a disk cannot be referenced at one time, but these circumstances will be explained in later text.)

Question marks (?) can be used in a wildcard file name to take the place of single characters at fixed file name positions. For instance, the wild card

JOB?.HEX

would refer to any file that has the extention "HEX," a primary name with the characters "J," "O," "B," and one or fewer additional characters. Hence the files "JOB0.HEX," "JOB1.HEX," and "JOBY.HEX" are just a few of the files that could be referenced by such a wildcard (if these files existed on the disk).

You can use any number of question marks in a wildcard (up to 11) and the actual file names referenced will be those with the same characters as those that are explicitly stated in the wildcard and any characters in the place of the question mark characters.

Thus the wildcard "????????.COM" will reference any file on the disk with the "COM" extension — just like the "*.COM" wildcard.

# Disk Drives

A disk drive is a device that transfers data to and from disk storage media.

### DRIVE NAMES

To allow you to refer to disks and files within your disk drives, the CP/M Operating System recognizes each drive in your hardware environment by a distinct "drive name." A drive name consists of a letter of the alphabet in the range A through D and a colon (:).

Possible drive names are A: B: C: and D:

### DEFAULT DRIVE

The default drive is the drive to which the system will refer, unless you specifically tell the system to refer to a different drive. The default drive is also the drive named in the system prompt, which is displayed when the CP/M system is in control. (Drive A is always the default drive when you boot up, as shown by the "A>" system prompt.)

You can execute an application program that is stored as a file in the default drive by typing the primary file name of the application program file (the part without the "COM"extension) in response to the system prompt, as shown:

A>{**primary name**}CR

Where "A>" is the system prompt; and

where {**primary name**} is the primary name of the file that you want to execute. For this command to be valid, the file must reside on a disk in default drive A:, and have a "COM" extension.

If the file represented by {**primary name**} does not reside on default drive A:, then the system will display a message repeating your unfound entry with a question mark, as shown:

A>{**primary name**}CR
{primary name}?

This kind of error message will also occur if you use improper syntax in your entry or misspell your entry.

### Changing the Default Drive

You can change the default drive by typing the name of another drive and a carriage return at the "A>" prompt, as shown:

```
A>B:
```

Such an entry will produce a new system prompt, indicating that drive B: is now the default drive, as shown:

```
A>B:
B>
```

NOTE: Any drive that is changed to the default drive in this fashion must be a valid drive within your hardware environment, and it must contain a floppy disk prepared by the FORMAT program. When you switch default drives in this fashion, the CP/M system will assume that any Application Program you wish to execute should be found on a disk in the new default drive.

### Accessing a Non-Default Drive

A "non-default drive" is a disk drive whose name is not displayed in the system prompt. For instance, if the "A>" system prompt is displayed (meaning A: is the default drive), then your non-default drives are any valid drives with names "B:" through "D:."

When you want to execute an application program that resides on a disk in a non-default drive, type the name of the appropriate non-default drive immediately before the program's primary name and a carriage return:

A>**B:{primary name}** ⟨CR⟩

Where "A>" is the system prompt;

where **B:** is the name of the desired non-default drive; and

where {**primary name**} is the primary name of the file that you want to execute. The file must reside on a disk in non-default drive B:, and have a "COM" extension.

The CP/M system would respond to such an entry by "logging-in" disk B: to get the Application Program indicated by {**primary name**}, inserting an image of this program into computer memory, and executing the program.

## SWITCHING FLOPPY DISKS BETWEEN DRIVES

When you reference a floppy disk drive (by entering a command at a default drive, by changing default drives, or by logging-in a non-default drive with a command), the CP/M system remembers some of the characteristics of the disk in the referenced drives. Switching disks between drives can cause problems unless you tell the system to forget the old characteristics.

You can make the system forget about old disk characteristics by performing a "warm boot" (by holding down the "CTRL" key and pressing the "C"). This entry is often abbreviated as **CTRL-C,** and it is usually entered in response to the "A>" system prompt. It tells the operating system to forget what it knew about the disks that used to be in the drives. Then it redisplays the system prompt.

We suggest that you perform a warm boot whenever you remove a disk from a drive and replace it with another disk (unless the system prompts you to insert a different disk).

## USING CP/M WITH ONE DRIVE

This release of the CP/M Operating System enables you to access disks in four drives: two 5.25-inch soft-sectored drives and two 8-inch drives. However, you do not need this many drive slots to perform multi-drive operations, because CP/M can treat your single drive as if it were two drives.

CP/M actually believes that you have two of each size drive you have. The drives that CP/M believes you have are called "logical" drives. The drive slots that you can actually see in front of you are called "physical" drives. CP/M enables you to access information from either kind of drive by allowing the single drive to access information from different disks at different times during the same operation.

When you enter a valid command that requires more disks than you have physical drives, CP/M displays prompts that instruct you to insert the required disks into the physical drive at the appropriate time. The prompts appear in the form:

```
PUT DISK x IN DRIVE y: AND PRESS RETURN
```

Where "x" is the logical name that is assigned to a particular disk for the duration of the operation; and
where "y:" is the physical name given to the single drive, which is temporarily set to read data from or write data to a particular disk.

When such a prompt appears, you should remove the disk that is already in the drive, insert the disk that the prompt indicates, and enter a carriage return. Execution of the program will resume until the drive needs to write data to or read data from a different disk. Then a similar prompt will appear, requesting that you insert a different disk into the drive.

When switching disks between a drive in this manner, you do not have to perform a warm boot after a disk switch. However, you must keep track of which disk is which. If you insert the wrong disk, the required data will not be found and you might have to start the operation over.

The text on "Backup Procedures One" and "Working Disk Procedures One" (intended solely for users with a single-drive Z-100 or H-100) contains instructions for putting this "logical/physical drive" concept to work.

# Commands

In general, a command is a program that can help you to create, change, analyze, or move data. Commands are entered in response to a "system prompt."

A system prompt consists of the letter for the default drive and the greater-than (>) character. When you start-up CP/M, the system prompt is displayed on your console, as shown:

```
A>
```

The system prompt tells you that CP/M is ready to receive a command in the form of a "command line."

## COMMAND LINES

A command line is the form of response you make to the system prompt to bring up, or "invoke," a command. A command line usually consists of three components: the "function," the "argument(s)," and the "carriage return." The function is entered first, and it indicates the activity that will be performed. The first argument is entered one space after the function. The argument indicates what data (files, systems, disks, drives, etc.) the function's activity should be performed upon. Each following argument is separated by one space from the preceding one. After entering the function and argument, you must enter a carriage return to tell CP/M that the entire command line is ready for execution.

You will enter command lines in the following form:

A>{**function**} {**argument**} {**argument**}... ⓒⓡ

Where "A>" is the system prompt;

where {**function**} is mandatory for all commands;

where {**argument**} {**argument**}... is optional for some commands; and

where ⓒⓡ is mandatory for all commands.

Always separate the command line function and the command line argument with one space. Furthermore, any command entered at a prompt that includes the ">" character must end with a carriage return. However, commands themselves often display prompts as well. And when such a prompt ends with a colon (:), a carriage return is usually not required for CP/M commands and utilities.

NOTE: In this text, the entry of a carriage return will be illustrated with the symbol "↵."

There are two kinds of commands that can be executed in a CP/M operating environment: Resident Commands and Transient Commands.

**Resident Commands**

Resident commands reside within the CP/M operating system. Therefore, CP/M doesn't have to refer to a disk to know how to execute these commands — although the commands themselves might affect data that **is** on a disk.

The CP/M Operating System contains the following resident commands:

DIR Displays the names of files that reside on a disk.

ERA Erases specified files from a disk.

REN Renames a specified file on a disk.

SAVE Saves the contents of computer memory space by creating a file on a disk.

TYPE Displays the contents of a file on the terminal.

USER Enables you to divide the space on a disk into separate areas for different users.

This list shows only the command line function of the resident commands. See "Volume II: The CP/M-85 Reference Guide" for a comprehensive explanation of the arguments used when these commands are entered.

### Transient Commands

Transient commands are application programs that are supplied with the CP/M Operating System on your CP/M distribution software. These application programs help you to manipulate the operating system and to perform several other useful microcomputer activities.

These commands (also known as "utilities") are stored on the disk as files with the "COM" extension. When you issue a command that makes reference to one of these files, CP/M copies an image of this file from the appropriate disk, puts this image into one of CP/M's hollow areas inside the computer, and begins execution of the transient command.

The following list shows the files containing all of the transient commands offered by Heath/Zenith for CP/M Version 2.2.100.

| | |
|---|---|
| ASM.COM | LOAD.COM |
| BSYSGEN.COM | MVCPM207.COM |
| CONFIGUR.COM | PIP.COM |
| DDT.COM | PREL.COM |
| DUMP.COM | STAT.COM |
| DUP.COM | SUBMIT.COM |
| ED.COM | SYSGEN.COM |
| FORMAT.COM | XSUB.COM |
| LIST.COM | |

To use a transient command in a command line, you type the primary file name of the file that contains this command. This primary name is the command line function.

"Section Two: Software Preparation Procedures" provides step-by-step instructions for entering transient command lines. "Volume II: Reference Guide" provides further details on all aspects of these commands.

### BIOS Source Files

Distribution Disk II contains the source code files used to produce BIOS85.SYS and BIOS88.SYS. These files are supplied for the benefit of assembly language system programmers, and they will not be needed by most customers. Many of the files on Disk II are written in 8086 assembly language and require the Digital Research 8086 assembler (not included, available from Digital Research as part of CP/M-86) for re-assembly.

## COMMAND LINE ENTRY

The CP/M Operating System is very picky when it comes to accepting command lines. You must spell all components of a command line correctly and include the names of non-default disk drives whenever a referenced file is not on the default disk. If you don't, CP/M will not be able to execute your command and will respond by redisplaying the invalid command line with a question mark (?). However, CP/M does allow some flexibility in the way you may respond to the system prompt, as the following special entry explanations show.

NOTE: In this text, "CTRL" followed by a hyphen and a letter (CTRL-S, CTRL-C, CTRL-P for example) indicates that you should hold down the key marked CTRL (control key) while pressing the key marked with the letter.

The following list explains the single keys and combinations of keys that you can press to edit a command line before submitting it to CP/M for execution.

DELETE — Removes the previous character typed from the command line. Depending on how your Operating System is adjusted, the removed characters might be echoed (repeated in reverse) on the video console display, or erased from the display.

BACK SPACE — Removes the previous typed character. Also removes any "DELETED" characters that were echoed in the line immediately to the left of the cursor.

CTRL-H — Same as "BACK SPACE."

CTRL-X — Removes all characters typed in the command line, as if you used "BACK SPACE" all the way to the beginning of the line.

CTRL-U — Effectively removes all characters typed in the command line, and allows you to try again on the line beneath the old line. It leaves the display of the old command line on the console, and displays the "#" character at the end of this old line to label it as a nullified entry.

CTRL-R — Redisplays the edited version of a command line below the "scratch pad" version of the line without any of the "DELETED" characters that might have been echoed in the line. Also displays the "#" character at the end of the "scratch pad" version.

The following list explains the single keys and combinations of keys that you can press to end a command line and submit it to CP/M for execution.

RETURN Ends the command line, sends the command to the system for execution, displays nothing on the console. After execution of the command, CP/M redisplays system prompt. In this manual, the symbol ⓒⓡ will often be used in examples and instructions to remind you to make this entry at the end of a command line.

CTRL-J Same as "RETURN."

CTRL-M Same as "RETURN."

The following list explains the single keys and combinations of keys that you can press to change the way in which CP/M executes your command line:

CTRL-S Interrupts the display of data to the console when pressed once. Allows CP/M to resume data display when pressed a second time. This entry is useful when data scrolls by on a console too quickly for you to read it. Volume II: "The CP/M Reference Guide" explains the commands during which it is safe and useful to make this entry.

CTRL-P Causes CP/M to send everything it displays on the console device to the list device (LST:) at the same time. (The list device is usually a printer.) Making this entry a second time will stop the display to the list device. This entry is useful when you want to record the displays that appear during the execution of a command on paper.

The routing of console displays to the list device will continue during and after the execution of any resident command or transient commands STAT and DUMP. This data routing will remain in effect during (not after) the execution of any other transient command except SUBMIT and XSUB. This entry will not cause data routing to the list device during the execution of most other application programs either.

If you type a CTRL-P entry while your list device is disconnected, turned off, in a local mode, or off line, the system will not be able to send data to the device. Also neither the ^P entry nor CTRL-P will work if your system is not adjusted to operate your printer.

The following entry enables you to enter an unusually long command line:

CTRL-E Enables you to see the entire display of a command line that is longer than your screen is wide. When you type this entry, the remaining portion of your command line will be displayed at the left-hand end of the next screen line.

This entry will not send your command line to the system for execution (as a "RETURN" entry would). It is not essential that you enter CTRL-E when typing a command line that exceeds console display range because CP/M will process your command line even if it does not fit on one screen line.

Even if you type CTRL-E entries, a command line cannot exceed 127 characters in length. If you type a 128th character in a command line, CP/M will automatically interpret it as a carriage return and try to execute your command line based upon the first 127 characters.

The following entries enable you to enter comments that CP/M will ignore:

; (The semicolon.) Enables you to enter comments not intended for execution without receiving error feedback from CP/M. To cause CP/M to ignore a comment, you must make the ";" the first character entered at the system prompt. Comments can consist of any characters you wish, typed after the ";" entry, and followed by a ⓒⓡ or "CTRL-U" or "CTRL-X."

: (The colon) Same as ";" (semicolon), except that you should not begin a comment line with two consecutive colons.

The following entries enable you to rapidly skip several spaces in a command line or comment:

TAB As with a regular typewriter, this key enables you to advance several spaces without pressing the space bar several times. It skips to the eighth column of the console display range or to some column numbered by a multiple of eight. Hence if you enter a TAB at the beginning of a command line, you will skip six columns (because the system prompt takes up two columns). If you immediately enter another TAB, you will skip an additional eight columns, and so forth.

CTRL-I Same as "TAB."

*Section Two*

# Software Preparation Procedures

This part of the manual provides the following three procedures for preparing your CP/M software so that it works efficiently with your hardware:

- **Start-Up Procedure:** Helps you to prepare your microcomputer hardware for use, and to load the CP/M Operating System into your microcomputer.
- **Backup Procedures:** Help you to copy and customize your CP/M distribution software onto backup disks, to protect your software investment.
- **Working Disk Procedures:** Help you to combine useful application programs on a customized bootable disk.

Every user should perform one start-up procedure, one backup procedure, and one working disk procedure after receiving CP/M distribution software.

These procedures contain several examples of user interaction with a microcomputer. In these examples, displays presented on the microcomputer console will be represented by the following typestyle:

```
THIS TYPESTYLE represents console displays

        (0123456789#$*?:=.A>)
```

User input (the characters that you type through the console) will be represented by boldface type, as shown:

**BOLDFACE TYPE represents the things you type**

**(0123456789#$*?:=[.])**

When you should enter a carriage return by pressing the key marked "RETURN," the example will show the symbol ⓒ .

In many instances, the exact text of a display will vary by a few characters. This manual often substitutes a few letters in place of exact characters where variations are likely to occur. For instance, this manual will illustrate a program's serial number as "Serial number sss-sssss," while your terminal might actually display it as "Serial number 357-81469."

In cases where the exact characters you type will vary, this manual presents a description of the necessary characters within curved braces, { }. Hence, this manual might instruct you to make an entry in the form "**B:=A:{filename.ext}** ⓒ ", when your entry might actually be "**B:=A:CONFIGUR.COM** ⓒ ".

If you have trouble performing a procedure or if you obtain an error message, then refer to "Volume II: The CP/M-85 Reference Guide." The reference guide contains comprehensive explanations of each utility. If you encounter an error message that is not explained in the reference guide, then consult "Appendix A: Operating System Error Messages."

# START-UP PROCEDURE

This procedure will explain the sequence of steps necessary for starting up a session of CP/M use. This sequence includes the preparation of your hardware devices, the insertion of a bootable disk into the appropriate drive, and the movement of a copy of the CP/M operating system from a disk into your microcomputer's memory.

The most significant step in this sequence is the movement of CP/M from a disk into the microcomputer. This step is known as "bootstrapping" or "cold booting." You will perform this step at least once each time you use CP/M in your microcomputer. Once inside your microcomputer, CP/M can control an application program or perform one of the many tasks within its own repertoire.

This procedure consists of several steps to help you start-up the CP/M Operating System in your Z-100 microcomputer. After you have unpacked your computer and connected any peripheral device to it according to the procedures given in the Z-100 Series User's Manual, proceed with the following steps.

NOTE: Your Z-100 computer is equipped with an automatic bootstrap feature. When your Z-100 is shipped, this feature is set to boot up immediately when the computer is powered up or reset. Therefore, a manual bootstrap command is not necessary during the start-up procedure. For more information about booting up with the Z-100, see "Appendix B: Bootstrap."

1. Make certain that the power cords and transmission cables to all of your hardware devices are securely plugged into the proper receptacles. (Refer to the manual of each hardware device for assistance.)

2. Turn on the power switches to all of your hardware devices. Within a few seconds, the red light on your drive will glow.

3. Place CP/M Distribution Disk I into the lefthand drive (if you have a Low-Profile Z-100) or into the upper drive (if you have an All-in-One Z-100), and close the drive door.

   Within a few more seconds, a CP/M-85 identification message and system prompt will be displayed in the following form:

```
CP/M-85 VERSION 2.2.100 07/07/82

A>
```

You have successfully booted the CP/M-85 Operating System. The "A>" system prompt indicates that the system is waiting for you to enter a command. Proceed to "Backup Procedures", Page 1-32.

NOTE: If a message and prompt in this form do not appear, take one of the following steps:

- If nothing appears on the screen, hold down the **CTRL** key while pressing the **RESET** key to reset the computer. The bootstrapping procedure should resume automatically and cause display of the message and prompt.

- If a pointed finger prompt appears on the upper lefthand corner of the screen, then the automatic bootstrapping attempt was aborted. You must now enter a manual bootstrap command, as explained in "Appendix B: Bootstrap". (In general, manual bootstrapping can be performed by pressing the **B** key and then ㊐ .)

- If these steps continually fail to produce any screen display, refer to the section entitled "In Case of Difficulty" in the Z-100 Series User's Manual.

# BACKUP PROCEDURES

A CP/M backup disk is a disk that contains all of the data that is stored on a CP/M distribution disk.

We strongly recommend that you make backup disks the first time you boot up with CP/M distribution software. A backup disk will help you to protect your software investment, because it is subjected to daily use while your distribution disks are stored safely away.

Because Heath/Zenith offers such a wide range of hardware, this manual contains two different backup procedures:

**Backup Procedure One is for users with a single-drive Z-100 or H-100.**

**Backup Procedure Two is for users with a double-drive Z-100 or H-100.**

Turn to and follow the backup procedure specified for your computer. Use only one backup procedure.

If you feel that you can perform the procedure without step-by-step instructions, then you can use the "Procedure Synopsis" at the beginning of the procedure for an overview of the activities involved.

# Backup Procedure One

**For users with a single-drive Z-100 or H-100**

To perform this procedure, you must have the following:

- A Z-100 or H-100 microcomputer with only one inboard drive
- Two CP/M-85 Distribution Disks (version 2.2.100)
- Two blank, 5.25-inch, soft-sectored, double-sided, double-density, disks

During this procedure you will prepare the blank disks for data storage, copy the CP/M Operating System and files to the blank disks, and customize the system. Thus the blank disks will become replicas of the CP/M Distribution Disks.

To prepare for this procedure, label one blank disk "CP/M-85 Backup Disk I" and the other "CP/M-85 Backup Disk II".

The single-drive backup procedure requires a large number of disk exchanges. To avoid confusion, clearly label your destination (backup) disks prior to starting this process.

NOTE: The blank disks that you will convert into CP/M Backup Disks should be write-enabled during this entire procedure. Therefore, do not cover the notches of these blank disks with write-protect tabs. However, the CP/M Distribution Disks should be write-protected; so leave the notches of these disks covered by tabs, as shipped.

**PROCEDURE SYNOPSIS**

This procedure requires you to perform the following activities in sequence:

booting up
FORMAT
DUP
CONFIGUR

To begin Procedure One, boot up with Distribution Disk I in the computer's inboard disk drive. Proceed to the FORMAT activity.

## FORMAT

This FORMAT activity helps you prepare blank disks for data storage.

1. At the A> system prompt, type **FORMAT** and press ⓒⓡ . This entry invokes FORMAT, which displays the following:

```
CP/M-85 Format Version 2.2.100
This program is used to initialize a disk.
All information currently on the disk will be destroyed.
Is that what you want? (y/n):
```

2. Type **Y**. FORMAT will display:

```
Which drive do you wish to use for this operation?
```

3. Type **B**. FORMAT will display:

```
Number of sides? (1=single, 2=double):
```

4. Type **2**. FORMAT will display the following message:

```
Put the disk you wish to be formatted in drive B.
Press RETURN to begin, anything else to abort.
```

5. Remove Distribution Disk I, and replace it with Backup Disk I. Then close the disk drive, and press ⓒⓡ . Then FORMAT will display:

```
PUT DISK B IN DRIVE A: AND PRESS RETURN
```

6. Leave Backup Disk I in the drive, and press ⓒⓡ . The light on the disk drive will glow for more than one minute while the disk is being formatted.

```
Do you have more disks to format? (y/n):
```

7. Type **Y** at this prompt. FORMAT will display:

```
Which drive do you wish to use for this operation?:
```

8. Type **B**. FORMAT will display:

```
Number of sides? (1=Single, 2=Double):
```

9. Type **2**. FORMAT will display the following message:

```
Put the disk you wish to be formatted in drive B.
Press RETURN to begin, anything else to abort.
```

10. Remove Backup Disk I and replace it with Backup Disk II. Then close the disk drive, and press . The light on the disk drive will glow for several seconds. Then FORMAT will display:

```
Do you have more disks to format? (y/n):
```

11. Type **N** at this prompt. FORMAT will display:

```
PUT DISK A IN DRIVE A: AND PRESS RETURN
```

12. Remove Backup Disk II and insert Distribution Disk I. Then press ⏎ . CP/M will display the system prompt:

```
A>
```

With Distribution Disk I in the drive, proceed to DUP.

## DUP

The DUP utility copies all of the data from one disk to another disk of the exact same type.

1. At the A> prompt, type **DUP** and press ⓒ. DUP will display the following:

```
Disk Utility Program
Version 2.2.100

Do you want to:

   A  copy and verify
   B  copy only
   C  verify only

   Z  exit to operating system

Selection:
```

2. Type **A**. DUP will display:

```
Source unit:
```

3. Type **A**. DUP will display:

```
Destination unit:
```

4. Type **B**. DUP will display:

```
Put source disk in drive A.
Put destination disk in drive B.

Press RETURN to begin:
```

5. Ignore the "Put destination disk" prompt (since there is no physical drive B). Since Distribution Disk I is the "source disk" for this operation (and since it already resides in the drive), press ⓒ. The following prompt will appear:

```
PUT DISK B IN DRIVE A: AND PRESS RETURN
```

6. Remove Distribution Disk I from the drive, insert Backup Disk I, close the drive latch, and press ⓒ. Soon a similar prompt will appear:

```
PUT DISK A IN DRIVE A: AND PRESS RETURN
```

7. Temporarily assign the following identities to your disks:

   Distribution Disk I is "DISK A" and
   Backup Disk I is "DISK B".

   Insert these disks alternately, as specified in the prompts.

   When the copy operation is finished, DUP will display:

   ```
   Copy finished.
   ```

   After copying, DUP will automatically start to verify the accuracy of the copy operation. Continue to insert the appropriate disks as prompted. When the verification operation is finished, DUP will display:

   ```
   Verification finished.
   ```

   Then DUP will redisplay the selection menu.

8. Type **A**. DUP will display:

   ```
   Source unit:
   ```

9. Type **A**. DUP will display:

   ```
   Destination unit:
   ```

10. Type **B**. DUP will display:

    ```
    Put source disk in drive A.
    Put destination disk in drive B.

    Press RETURN to begin:
    ```

11. Ignore the "Put destination disk" prompt (since there is no physical drive B). Insert Distribution Disk II as the "source disk" for this duplication operation and press ⓒⓡ. Soon the following prompt will appear:

    ```
    PUT DISK B IN DRIVE A: AND PRESS RETURN
    ```

12. Remove Distribution Disk II from the drive, insert Backup Disk II, close the drive latch, and press ⓒⓡ. Soon a similar prompt will appear:

    ```
    PUT DISK A IN DRIVE A: AND PRESS RETURN
    ```

13. Temporarily assign the following identities to your disks:

    Distribution Disk II is "DISK A"; and
    Backup Disk II is "DISK B".

    Insert these disks alternately, as specified in the prompts.

    When the copy operation is finished, DUP will display:

    ```
    Copy finished.
    ```

    After copying, DUP will automatically start to verify the accuracy of the copy operation. Continue to insert the appropriate disks as prompted. When the verification operation is finished, DUP will display:

    ```
    Verification finished.
    ```

    Then DUP will redisplay the selection menu.

14. Type **Z** at the DUP selection menu. DUP will display the following prompt:

    ```
    Place a bootable disk in drive A and type any character:
    ```

15. Insert Backup Disk I in the drive, and type any keyboard character. The following prompt will be displayed:

    ```
    PUT DISK A IN DRIVE A: AND PRESS RETURN
    ```

16. Leave Backup Disk I in the drive, and press ⏎. CP/M will display the system prompt, as shown:

    ```
    A>
    ```

17. Store your distribution software away in a safe place. Use your backup software for all future activities.

Proceed to CONFIGUR.

## CONFIGUR

The CONFIGUR utility adjusts the CP/M Operating System on Backup Disk I for your hardware.

NOTE: You should skip the CONFIGUR activity and proceed to "Working Disk Procedure One" on Page 1-51 if you do not have a printer or modem, or if you have one of the following:

- A serial printer (such as the Z-25 or the H-25) that runs at 4800 baud, accepts 8 bits per character with no parity bit, handshakes with RTS pin number 4, is ready when handshaking signal is High, and has no protocol.

- A modem (such as the WH-13, the Lexicon WH-23, the UDS WH-33, or the Hayes WH-43) that runs at 300 baud, accepts 8 bits per character with no parity bit, and uses no handshaking.

If you have a printer and/or modem that is **not** listed in these descriptions, then begin this activity at Step 1.

1. Type **CONFIGUR** ㊐ . CONFIGUR will display the following menu:

```
CP/M-85 System Configuration Utility version 2.2.100
      Copyright© 1982 by Zenith Data Systems

                *** MAIN MENU ***

          P - Printer Configuration
          M - Modem Configuration
          C - Command Configuration
          I - I/O Map Configuration
          ? - Brief Help Message

          X - Exit

      Selection [P,M,C,I,X or ?] :
```

Type **P** and ⏎. CONFIGUR will display the following menu:

```
                    *** Printer Configuration ***

  1 - MX-80 or other PARALLEL Centronics-interface printer
  2 - H/Z-25
  3 - H-14 or TI-810(WH-24)
  4 - Dec LA-34 or LA-36
  5 - Diablo 620
  6 - Diablo 630,1610,1620,1630 or 1640(WH-44)
  7 - MX-80 Serial
  8 - Votrax Type 'n Talk
  9 - User-defined SERIAL Printer

Please choose the number that corresponds to your printer :
```

3. Enter the number to the left of your printer's name and ⏎. CONFIGUR will display a message listing some characteristics of your printer. If these characteristics do not match those of your printer, either change your printer settings (see printer manual) or specify characteristics of a user-defined printer (see CONFIGUR section in "Reference Guide").

   NOTE: If your printer is not listed by name in the "Printer Selection" menu, press **9** and ⏎, and refer to the Reference Guide for instructions.

4. When the message listing the characteristics of your printer is displayed, press ⏎ at the "Press RETURN to see Main menu:" prompt. CONFIGUR will redisplay the "MAIN MENU".

NOTE: If you have a modem that is **not** set with standard Heath/Zenith characteristics, refer to the CONFIGUR text in "Volume II: Reference Guide" on Page 2-43 and Page 2-51.

5. Type **X** and ⏎ at the "MAIN MENU". CONFIGUR will display the following menu:

```
                    *** EXIT OPTIONS ***

T - Make changes temporary (to memory only)
P - Make changes permanent (to memory and disk)
Q - Make no changes

? - Brief Help Message

Choice [T,P,Q or ?]:
```

6. Type **P** and ⏎ to apply the specified changes to the system, and to record this system on the disk in drive A. CONFIGUR will either display one or two graphics depicting the rear panel of your Z-100 computer, or relinquish control to the CP/M system.

7. If CONFIGUR displays the "Z-100 Rear Panel" graphic, then attach your printer and/or modem cables as shown in the graphic. Then press ⏎. CP/M will then display the system prompt.

   If CP/M immediately displays the system prompt, proceed to Step 8.

8. If you have a printer, test it by holding down the **CTRL** and pressing the **P** key. Then press ⏎ a few times. Your printer (if properly configured and connected) should print system prompts just as they are displayed to the video screen. Then enter **CTRL-P** again to discontinue this printer test.

You have just finished your backup procedure, in which you duplicated the information from your distribution software and configured the system on one of the backup disks.

Proceed to "Working Disk Procedure One" on Page 1-51.

# Backup Procedure Two

**For users with a double-drive Z-100 or H-100**

To perform this procedure, you must have the following:

- A Z-100 or H-100 microcomputer with two inboard drives
- Two CP/M-85 Distribution Disks (version 2.2.100)
- Two blank, 5.25-inch, soft-sectored, double-sided, double-density, disks

During this procedure you will prepare the blank disks for data storage, copy the CP/M Operating System and files to the blank disks, and customize the system. Thus the blank disks will eventually become usable replicas of the CP/M Distribution Disks.

To prepare for this procedure, label one blank disk "CP/M-85 Backup Disk I", and the other "CP/M-85 Backup Disk II".

NOTE: The blank disks that you will convert into CP/M Backup disks should be write-enabled during this entire procedure. Therefore, do not cover the notches of these blank disks with write-protect tabs. However, the CP/M Distribution Disks should be write-protected; so leave the notches of these disks covered by tabs, as shipped.

**Procedure Synopsis**

This procedure requires you to perform the following activities in sequence:

booting up
FORMAT
DUP
CONFIGUR

To begin Procedure Two, insert Distribution Disk I in the left or upper disk drive of the computer (drive A), and insert Backup Disk I in the computer's remaining drive (drive B). Boot up the system.

Proceed to the FORMAT activity.

## FORMAT

This FORMAT activity helps you prepare blank disks for data storage.

1. At the A> System Prompt, type **FORMAT** and press CR . This entry invokes FORMAT, which displays the following:

```
CP/M-85 Format Version 2.2.100

This program is used to initialize a disk.
All information currently on the disk will be destroyed.
Is that what you want? (y/n):
```

2. Type **Y**. FORMAT will display:

```
Which drive do you wish to use for this operation?
```

3. Type **B**. FORMAT will display:

```
Number of sides? (1=single, 2=double):
```

4. Type **2**. FORMAT will display the following message:

```
Put the disk you wish to be formatted in drive B.
Press RETURN to begin, anything else to abort.
```

5. Check to make certain that Backup Disk I is in drive B, and press CR. The light on the disk drive will glow for more than a minute, as the disk is being formatted. Then FORMAT will display:

```
Do you have more disks to format? (y/n):
```

6. Type **Y** at this prompt. FORMAT will display:

```
Which drive do you wish to use for this operation?
```

7. Type **B**. FORMAT will display:

```
Number of sides? (1=Single, 2=Double):
```

8. Type **2**. FORMAT will display the following message:

```
Put the disk you wish to be formatted in drive B.
Press RETURN to begin, anything else to abort.
```

9. Remove Backup Disk I, from drive B, and replace it with Backup Disk II. Then close the disk drive, and press ⓒⓡ. The light on the disk drive will glow for more than a minute while the disk is being formatted. Then FORMAT will display:

```
Do you have more disks to format? (y/n):
```

10. Type **N** at this prompt. CP/M will display:

```
A>
```

Leave Distribution Disk I in drive A, insert Backup Disk I into drive B, and proceed to DUP.

## DUP

The DUP utility copies all of the data from one disk to another disk of the exact same type.

1. At the A> prompt, type **DUP** and press ⓒ. DUP will display the following:

```
Disk Utility Program
Version 2.2.100

Do you want to:

   A  copy and verify
   B  copy only
   C  verify only

   Z  exit to operating system

Selection:
```

2. Type **A**. DUP will display:

```
Source unit:
```

3. Type **A**. DUP will display:

```
Destination unit:
```

4. Type **B**. DUP will display:

```
Put source disk in drive A.
Put destination disk in drive B.

Press RETURN to begin:
```

5. Check to make certain that Distribution Disk I is in drive A, and that Backup Disk I is in drive B. Then press ⓒ. When the copy operation is finished, DUP will display:

```
Copy finished.
```

   After copying, DUP will automatically start to verify the accuracy of the copy operation. When the verification operation is finished, DUP will display:

```
Verification finished.
```

   Then DUP will redisplay the selection menu.

6. Type **A**. DUP will display:

```
Source unit:
```

7. Type **A**. DUP will display:

```
Destination unit:
```

8. Type **B**. DUP will display:

```
Put source disk in drive A.
Put destination disk in drive B.

Press RETURN to begin:
```

9. Insert Distribution Disk II into drive A, insert Backup Disk II into drive B, and press CR. When the copy operation is finished, DUP will display:

```
Copy finished.
```

   After copying, DUP will automatically start to verify the accuracy of the copy operation. When the verification operation is finished, DUP will display:

```
Verification finished.
```

   Then DUP will redisplay the selection menu.

10. Type **Z** at the DUP selection menu. CP/M will display the system prompt, as shown:

```
A>
```

11. Store your distribution software in a safe place, and insert Backup Disk I in drive A.

Proceed to CONFIGUR.

## CONFIGUR

The CONFIGUR utility adjusts the CP/M Operating System on Backup Disk I for your hardware.

NOTE: You should skip the CONFIGUR activity and proceed to "Working Disk Procedure Two" on Page 1-56 if you do not have a printer or modem, or if you have one of the following:

- A serial printer (such as the Z-25 or the H-25) that runs at 4800 baud, accepts 8 bits per character with no parity bit, handshakes with RTS pin number 4, is ready when handshaking signal is High, and has no protocol.

- A modem (such as the WH-13, the Lexicon WH-23, UDS WH-33, or the Hayes WH-43) that runs at 300 baud, accepts 8 bits per character with no parity bit, and uses no handshaking.

If you have a printer and/or modem that is **not** listed in these descriptions, then begin this activity at Step 1.

1. Type **CONFIGUR** ㊞. CONFIGUR will display the following menu:

```
CP/M-85 System Configuration Utility version 2.2.100
     Copyright(c) 1982 by Zenith Data Systems

                *** MAIN MENU ***

          P - Printer Configuration
          M - Modem Configuration
          C - Command Configuration
          I - I/O map Configuration
          ? - Brief Help Message

          X - Exit

     Selection [P,M,C,I,X or ?]:
```

2. Type **P** and ⓒ. CONFIGUR will display the following menu:

```
                    *** Printer Configuration ***

     1 - MX-80 or other PARALLEL Centronics-interface printer
     2 - H/Z-25
     3 - H-14 or TI-810(WH-24)
     4 - Dec LA-34 or LA-36
     5 - Diablo 620
     6 - Diablo 630,1610,1620,1630 or 1640(WH-44)
     7 - MX-80 Serial
     8 - Votrax Type 'n Talk
     9 - User-defined SERIAL Printer

   Choose the number that corresponds to your printer :
```

3. Enter the number to the left of your printer's name and ⓒ. CONFIGUR will display a message listing some characteristics of your printer. If these characteristics do not match those of your printer, either change your printer settings (see printer manual) or specify characteristics of a user-defined printer (see CONFIGUR section in the "Reference Guide").

   NOTE: If your printer is not listed by name in the "Printer Configuration" menu, press **9** and ⓒ, and refer to the "Reference Guide" for instructions.

4. When the message listing the characteristics of your printer is displayed, press ⓒ at the "Press RETURN to access Main menu:" prompt. CONFIGUR will redisplay the "Configur Main Menu".

NOTE: If you have a modem that is **not** set with standard Heath/Zenith characteristics, refer to the CONFIGUR text in "Volume II: Reference Guide" on Page 2-43 and Page 2-51.

5. Type **X** and ⏎ at the "Configur Main Menu". CONFIGUR will display the following prompt:

```
                    *** EXIT OPTIONS ***

T - Make changes temporary (to memory only)
P - Make changes permanent (to memory and disk)
Q - Make no changes

? - Brief Help Message

Choice [T,P,Q or ?]:
```

6. Type **P** and ⏎ to apply the specified changes to the system, and to record this system on the disk in drive A. CONFIGUR will either display a graphic depicting the rear panel of your Z-100 computer, or relinquish control to the CP/M system.

7. If CONFIGUR displays any "Z-100 Rear Panel" graphics, then attach your printer and/or modem cables as shown in the graphic. Then press ⏎. CP/M will then display the system prompt.

   If CONFIGUR immediately relinquishes control to the CP/M system, CP/M will display the system prompt.

8. If you have a printer, test it out by holding down the **CTRL** key and pressing the **P** key. Then press ⏎ a few times. Your printer (if properly configured and connected) should print system prompts just as they are displayed to the video screen. Then enter **CTRL-P** again to discontinue this printer test.

You have just finished your backup procedure, in which you duplicated the information from your distribution software and configured the system on one of the backup disks.

Proceed to "Working Disk Procedure Two" on Page 1-56.

# WORKING DISK PROCEDURES

A Working Disk is a disk that contains both a customized CP/M Operating System and useful application programs.

Most users find working disks to be convenient, if not essential, because combining the operating system and application programs on the same disk makes it quicker and easier to access vital programs and/or data.

Because Heath/Zenith CP/M-85 supports such a wide range of hardware items, this manual contains two procedures for making a Working Disk:

**Working Disk Procedure One is for users with a single-drive Z-100 or H-100.**

**Working Disk Procedure Two is for users with a double-drive Z-100 or H-100.**

Turn to and follow the working disk procedure specified for your hardware environment. Use only one procedure.

If you feel you can perform the procedure without step-by-step instructions, then you can use the "Procedure Synopsis" at the beginning of the procedure for an overview of the activities involved.

# Working Disk Procedure One

### For users with a single-drive Z-100 or H-100

To perform this procedure, you must have the following:

- A Z-100 or H-100 microcomputer with only one inboard drive.
- CP/M-85 Backup Disk I (version 2.2.100).
- A blank, 5.25-inch, soft-sectored, double-sided, double-density, disk.
- Any number of Application Program Disks containing software designed for use under Heath/Zenith CP/M-85.

During this procedure, you will prepare the blank disk for data storage, copy the CP/M Operating System to it, and copy application program files to it from an Application Program Disk.

To prepare for this procedure, label the blank disk with the name(s) of the application program(s) you plan to copy, and with the words "Working Disk".

NOTE: The blank disk that you will convert into a Working Disk should be write-enabled during this entire procedure. Therefore, do not cover the notch of this blank disk with a write-protect tab.

**PROCEDURE SYNOPSIS**

This procedure requires you to perform the following activities in sequence:

booting up
FORMAT
SYSGEN
PIP

To begin Procedure One, boot up with CP/M Backup Disk I in the computer's left or upper disk drive. Proceed to the FORMAT activity.

## FORMAT

This FORMAT activity helps you prepare blank disks for data storage.

1. At the A> system prompt, type **FORMAT** and press ⓒⓡ. This entry invokes FORMAT, which displays:

```
CP/M-85 Format Version 2.2.100

This program is used to initialize a disk.
All information currently on the disk will be destroyed.
Is that what you want? (y/n):
```

2. Type **Y**. FORMAT will display:

```
Which drive do you wish to use for this operation?
```

3. Type **B**. FORMAT will display:

```
Number of sides? (1=single, 2=double):
```

4. Type **2**. FORMAT will display the following message:

```
Put the disk you wish to be formatted in drive B.
Press RETURN to begin, anything else to abort.
```

5. Remove Backup Disk I and replace it with the Working Disk. Then close the disk drive door, and press ⓒⓡ. Then FORMAT will display:

```
PUT DISK B IN DRIVE A: AND PRESS RETURN
```

6. Leave the Working Disk in the drive, and press ⓒⓡ. The light on the disk drive will glow for more than one minute while the disk is being formatted. Then FORMAT will display:

```
Do you have more disks to format? (y/n):
```

7. Type **N** at this prompt. FORMAT will display:

```
PUT DISK A IN DRIVE A: AND PRESS RETURN
```

8. Remove the Working Disk, and insert Backup Disk I. Then press ⓒⓡ. Then CP/M will display the system prompt.

```
A>
```

With Backup Disk I in the drive, proceed to SYSGEN.

## SYSGEN

Use the SYSGEN utility to copy the CP/M Operating System to the Working Disk from the Backup Disk.

1. At the A> prompt, type **SYSGEN** and press ⓒ. The SYSGEN utility will display the message:

```
CP/M-85 SYSGEN VER 2.2.100
SOURCE DRIVE NAME (OR RETURN TO SKIP):
```

2. At this prompt type **A**. SYSGEN will display:

```
SOURCE ON A, THEN TYPE RETURN
```

3. Press ⓒ. SYSGEN will display:

```
FUNCTION COMPLETE.
COPY BIOS88.SYS & BIOS85.SYS (Y/N):
```

4. Type **Y**. SYSGEN will display:

```
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT):
```

5. Type **B**. SYSGEN will display:

```
DESTINATION ON B, THEN TYPE RETURN
```

6. Press ⓒⓡ. SYSGEN will display the following prompt:

```
PUT DISK B IN DRIVE A: AND PRESS RETURN
```

7. Remove Backup Disk I, insert the Working Disk, and press ⓒⓡ . SYSGEN will display:

```
FUNCTION COMPLETE.
DESTINATION DRIVE NAME (OR RETURN TO REBOOT):
```

8. Press ⓒⓡ . SYSGEN will display the following prompt:

```
PUT DISK A IN DRIVE A: AND PRESS RETURN
```

9. Remove the Working Disk, and insert Backup Disk I. Then press ⓒⓡ . CP/M will display:

```
A>
```

With Backup I in the drive, proceed to PIP.

**PIP**

The PIP activity will help you to copy application program files from Application Program Disks to your Working Disk.

1. Type **PIP B:=A:PIP.COM[V]** ↵ .

2. Insert "Disk A" (Backup Disk I) and "Disk B" (Working Disk) as prompted, until the system prompt appears.

3. Reset the computer by holding down the **CTRL** key while pressing the **RESET** key.

4. Boot up with the Working Disk.

5. Type **PIP** ↵ at the "A>" system prompt. This entry invokes PIP, which will display the "*" prompt.

6. Type **A:=B: {filename.ext}[V]** ↵ at the "*" prompt; where {filename.ext} is the name of a file you wish to copy from the Application Program Disk to the Working Disk.

7. Insert "DISK A" (Working Disk) and "DISK B" (Application Program Disk) as prompted. When finished copying the file, PIP will redisplay the "*" prompt.

8. For each application program you wish to copy from the **same** Application Program Disk, repeat Steps 6 and 7.

   For each application program you wish to copy from a **different** Application Program Disk, press ↵ at the "*" prompt and repeat Steps 5, 6, and 7.

You now have a Working Disk, containing a fully customized CP/M Operating System and some useful application programs.

# Working Disk Procedure Two

**For users with a double-drive Z-100 or H-100.**

To perform this procedure, you must have the following:

- A Z-100 or H-100 microcomputer with two inboard drives
- CP/M-85 Backup Disk I (version 2.2.100)
- A blank, 5.25-inch, soft-sectored, double-sided, double-density, disk
- Any number of Application Program Disks containing software designed to run under Heath/Zenith CP/M-85.

During this procedure you will prepare the blank disk for data storage, copy the CP/M Operating System to it, and copy application program files to it from an Application Program Disk.

To prepare for this procedure, label the blank disk with the names of the application program(s) you plan to copy and with the words "Working Disk".

NOTE: The blank disk that you will convert into a Working Disk should be write-enabled during this entire procedure. Therefore, do not cover the notch of this blank disk with a write-protect tab.

**PROCEDURE SYNOPSIS**

This procedure requires you to perform the following activities in sequence:

booting up
FORMAT
SYSGEN
PIP

To begin Procedure Two, boot up with CP/M Backup Disk I in the left or upper disk drive of the computer (drive A). Then insert the Working Disk into the computer's remaining drive (drive B).

Proceed to the FORMAT activity.

**FORMAT**

This FORMAT activity helps you prepare blank disks for data storage.

1. At the A> System Prompt, type **FORMAT** and press ⓒ . This entry invokes FORMAT, which displays the following:

```
CP/M-85 Format Version 2.2.100

This program is used to initialize a disk.
All information currently on the disk will be destroyed.
Is that what you want? (y/n):
```

2. Type **Y**. FORMAT will display:

```
Which drive do you wish to use for this operation?:
```

3. Type **B**. FORMAT will display:

```
Number of sides? (1=single, 2=double):
```

4. Type **2**. FORMAT will display the following message:

```
Put the disk you wish to be formatted in drive B.
Press RETURN to begin, anything else to abort.
```

5. Press ⓒ . The light on the disk drive will glow for more than one minute while the disk is being formatted. Then FORMAT will display:

```
Do you have any more disks to format? (y/n):
```

6. Type **N** at this prompt. CP/M will display:

```
A>
```

With Backup Disk I in drive A and the Working Disk in drive B, proceed to SYSGEN.

## SYSGEN

Use the SYSGEN utility to copy the CP/M Operating System to the Working Disk from the Backup Disk.

1. At the A> prompt, type **SYSGEN** and press ⓒ. This entry invokes SYSGEN, which will display the following message:

```
CP/M-85 SYSGEN VER 2.2.100
SOURCE DRIVE NAME (OR RETURN TO SKIP):
```

2. Type **A.** SYSGEN will display:

```
SOURCE ON A, THEN TYPE RETURN
```

3. Press ⓒ. SYSGEN will display:

```
FUNCTION COMPLETE
COPY BIOS88.SYS & BIOS85.SYS (Y/N):
```

4. Type **Y.** SYSGEN will display:

```
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT):
```

5. Type **B.** SYSGEN will display:

```
DESTINATION ON B, THEN TYPE RETURN
```

6. Press ⓒ. SYSGEN will display:

```
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT):
```

7. Press ⓒ. CP/M will display:

```
A>
```

With Backup Disk I in drive A and the Working Disk in drive B, proceed to PIP.

**PIP**

This PIP activity will help you to copy application program files from Application Program Disks to your Working Disk.

1. Type **PIP B:=A:PIP.COM[V]** ㊞ .

2. When the system prompt appears, reset the computer holding down the **CTRL** key while pressing the **RESET** key.

3. Remove both disks. Then insert the Working Disk in the left or upper drive, and the Application Program Disk in the remaining drive.

4. Boot up with the Working Disk.

5. Type **PIP** ㊞ at the "A>" system prompt. This entry form causes PIP to display the "*" prompt.

6. Type **A:=B:{filename.ext}[V]** ㊞ at the "*" prompt; where {filename.ext} is the name of a file you wish to copy from the Application Program Disk to the Working Disk. When finished copying the file, PIP will redisplay the "*" prompt.

7. For each application program you wish to copy from the **same** Application Program Disk, repeat Step 6.

   For each application program you wish to copy from a **different** Application Program Disk, press ㊞ at the "*" prompt, and repeat Steps 5 and 6.

You now have a Working Disk that contains a fully customized CP/M Operating System and some useful programs.

# CP/M-85

# REFERENCE GUIDE

## Volume II

HEATH | ZENITH data systems

First Printing
Printed in the
United States of America

II

# ABOUT THIS VOLUME

This Reference Guide contains detailed descriptions of each of the commands accompanying Heath/Zenith CP/M-85. The guide deals with all 23 resident commands and transient commands, presenting each command in alphabetical order. The text explains the following aspects of commands (where applicable):

- Function
- Invocation
- Options or Parameters
- Common applications
- Error message explanations

The text on each command is divided into numbered sections and subsections concerning specific aspects of the command. The numbers heading important sections are often indicated within a summary paragraph at the beginning of the text.

The Reference Guide features several examples of user interaction with the microcomputer. In these examples, displays presented by the microcomputer console will be represented by the following typestyle:

```
THIS TYPESTYLE represents console displays

(0123456789#$*.' .A
```

User input will be represented by boldfaced type, as shown:

**BOLDFACED TYPE represents the things the user types**

**(0123456789#$*?:=[.])**

When you should enter a carriage return by pressing the key marked "RETURN", the example will show the symbol ⓒⓡ.

An example of user interaction with the terminal might appear as follows:

A>**STAT {drive}:{file name}** ⓒⓡ

Where "A>" is a console display (displayed by CP/M);

where **STAT** is a command entry typed through the console by the user;

**where {drive}, {file name}, or any other description enclosed in curved braces is an entry that the user should supply; and**

where ⓒⓡ is a user-entered carriage return.

The "where" statements that immediately follow a sample screen display or sample entry describe the important components of the display or entry.

# ASM

*The Utility that Creates an Intel Hexadecimal File and a Print-Out File from an Assembly Language Program File*

The ASM assembler is designed to use the 8080 instruction set. The Z-100 and H-100 microcomputers contain an 8085 central processing unit, which accommodates 8080 instructions. This section of the Reference Guide assumes that the user is familiar with assembly language programming, and with the use of 8080 instructions. A brief synopsis of the function of 8080 op-codes is provided in "Appendix D: Assembler Operation Codes." For individuals with no background related to assembly language programming, the Heathkit Continuing Education Course in Assembly Language Programming is recommended.

The ASM utility reads assembly language source files (files with the "ASM" extension) from a disk and translates them into two output files (1):

- An output file with the "HEX" extension, containing 8080 machine code in Intel hexadecimal form (2). This file can be loaded into the computer (using the LOAD utility) and executed under the CP/M Operating System.

- An output file with the "PRN" extension, containing ASCII characters. This file can be printed out, displayed on a console screen, or not produced (1.1). It includes the same mnemonics as the "ASM" file (3, 4, and 5), plus the hexadecimal address of each source line and signal characters for errors that occurred during the assembly (6.2).

# 1. ASM INVOCATION

The ASM utility is invoked in response to the system prompt by entering a command in the following form:

A>**ASM {file name}.{asm}{hex}{prn}** ⏎

Where {**file name**} is the primary name of an assembly language source file whose extension is assumed to be "ASM". This file will be translated into the two output files by ASM;

where {**asm**} represents a one-letter parameter entered to specify the drive in which the assembly language source file is located;

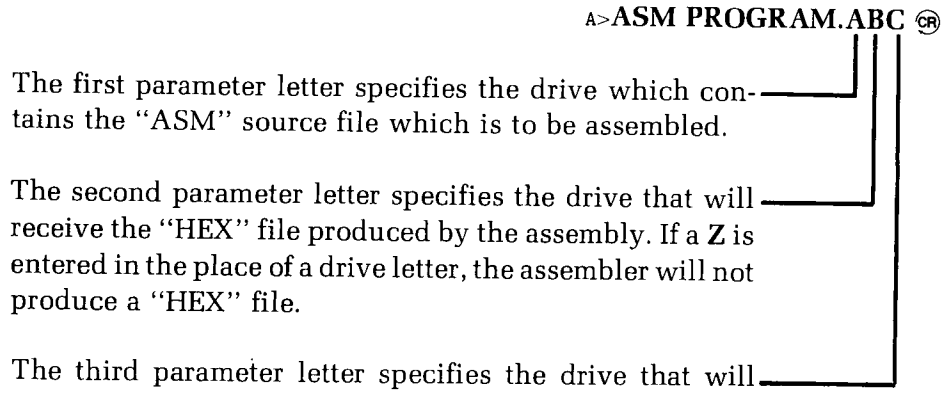
where {**hex**} represents a one-letter parameter entered to specify the drive to which the "HEX" output file will be written; and

where {**prn**} represents a one-letter parameter entered to specify the drive to which the "PRN" output file will be written.

CP/M assumes that the source file has the extension "ASM". Therefore, the "ASM" extension does not have to be entered in the invocation command.

## 1.1 Assembler Parameters

Three single-letter parameters can be entered immediately after the period (.) in the invocation command line, as if they were a file extension. The following example gives a detailed explanation of each parameter:




If you omit any of these parameter letters from the invocation command line, the ASM utility will try to read from, or write to, the default drive.

## 1.2 Example ASM Command Lines

The following examples and explanations illustrate the different forms of the ASM command line and the results they produce.

**A>ASM PROGRAM**↵

Source file PROGRAM.ASM is read from the default drive and assembled. Hex file PROGRAM.HEX and print file PROGRAM.PRN are written to the default drive.

**A>ASM PROGRAM.ABB**↵

Source file PROGRAM.ASM is read from drive A and assembled. Output files PROGRAM.HEX and PROGRAM.PRN are both written to drive B.

**A>ASM PROGRAM.BAX**↵

Source file PROGRAM.ASM is read from drive B and assembled. Hex file PROGRAM.HEX is written to disk A, and print file PROGRAM.PRN is displayed on the console.

**A>ASM PROGRAM.BZZ**↵

Source file PROGRAM.ASM is read from drive B and assembled. Neither output file is created. Such an entry is used to check for source file errors.

# 2. ASM EXECUTION

When invoked, ASM identifies itself with the following display:

```
CP/M ASSEMBLER - VER 2.0
```

Where "2.0" indicates the version number of the ASM utility.

If you specified (in a parameter) that the PRN file be displayed rather than stored, then this display will now appear on the console. This display (and the assembly operation) can be suspended by entering a **CTRL-S**, and resumed by entering any character. However, if you enter a character other than CTRL-S before entering CTRL-S, the CTRL-S entry will not suspend the display.

The display and assembly operation can be aborted by resetting the computer.

After an assembly operation is completed, ASM displays a message in the form of the following example:

```
0FD4
02AH USE FACTOR
END OF ASSEMBLY

A>
```

Where "0FD4" (or any number appearing in this position) is the last address (in hexadecimal) of the HEX file produced in the assembly; and

where "02AH" represents the percentage of free system memory used for ASM's symbol table. This percentage is expressed in hexadecimal values, with 000H being 0 percent, and 0FFH being 100 percent.

# 3. FORM FOR SOURCE FILE STATEMENTS

Assembly language source files must be composed of program statements in the following form:

**line# label operator operand ;comment**

Where the **line#** field is an optional integer value representing the source program line number. These numbers are only for the convenience of the user and are ignored by the assembler when present;

where the **label** field is optional, except when required by particular statement types. It consists of a maximum of 16 alphanumeric characters (letters and numbers). The first character must be a letter. Labels can be freely used by the programmer to identify elements such as program steps.

A single label should not contain any spaces, because ASM interprets a space as the end of the label field. However, the dollar sign ($) can be used to improve the readability of labels that contain more than one word. ASM will ignore the dollar sign, and consider the other characteristics in the label as if they were in a continuous string. The label "DATES$OF$BIRTH" is somewhat easier to read than the label "DATESOFBIRTH". ASM will treat the two labels as being identical.

Labels may also be followed by a colon (:) to maintain compatibility with some other assemblers. The colon will not become part of the label and be ignored by ASM.

Where **operator** is either a pseudo operation, or an assembler directive (see 5, Page 2-15), or an assembler opcode;

where the **operand**, in general, contains an expression formed from constants and labels together with arithmetic and logical operations on these elements (see 4, Page 2-9); and

where a **comment** consists of any characters following a semicolon (;) until an end-of-line is encountered. An asterisk (*) used as the first character on a line will also indicate a comment. All comments are read and listed in the print file but otherwise ignored by ASM.

Any or all of these fields may be present in a statement, each separated by a space or a tab. Each assembly language statement is terminated by a carriage return and a line feed (both of which are inserted when the RETURN key is pressed with many text editors), or with the exclamation point character (which is recognized by the ASM utility as an "end-of-line" character). If the exclamation point (!) is used to signify the end of an assembly language statement, other statements can be entered on the same physical line.

# 4. FORMING THE OPERAND

In order to completely describe the operation codes and pseudo operation, it is first necessary to present the form of the operand since it is used in nearly all statements.

Expressions consist of simple operands (labels, constants, and reserved words) combined in properly formed subexpressions by arithmetic and logical operators. Expression computation is carried out by the assembler as the assembly proceeds. Each expression must produce a 16-bit value during the assembly.

The number of significant digits in the result must not exceed the intended use. If an expression is to be used in a byte move-immediate instruction, then the most significant 8 bits of the expression must be zero. The restrictions on the expression significance are given with the individual instructions.

## 4.1 Labels

A label is an identifier which occurs on a particular statement. Generally, the label is given a value determined by the type of statement which it precedes.

If the label occurs on a statement which generates machine code or reserves memory space (such as a MOV instruction, or a DS pseudo operation), the label is given the value of the program address that it labels. If the label precedes an EQU or SET instruction, the label is given the value resulting from evaluating the operand. Except for the SET statement, an identifier can only label one statement. Unless a SET instruction follows a particular label, this label can occur only once in a program.

When a label appears in the operand, its value is substituted by the assembler. This value can then be combined with other operands and operators to form an operand for a particular instruction.

## 4.2 Numeric Constants

A numeric constant is a 16-bit value in one of several bases. The base, or radix, of the constant is denoted by a trailing radix indicator. The radix indicators are:

B. Binary constant (base 2)
O Octal constant (base 8)
Q Octal constant (base 8)
D Decimal constant (base 10)
H Hexadecimal constant (base 16)

"Q" is an alternate radix indicator for octal numbers since the letter O is easily confused with the digit zero (0).

A numeric constant which does not terminate with a radix indicator is assumed to be a decimal constant.

A constant is composed of a sequence of digits, sometimes followed by a radix indicator. The digits are in the appropriate range for the radix. Binary constants must be composed of the digits 0 and 1, octal constants can contain digits in the range 0-7, while decimal constants contain decimal digits 0-9. Hexadecimal constants contain decimal digits and hexadecimal digits A (10D), B (11D), C (12D), D (13D), E (14D), and F (15D). The leading digit of a hexadecimal constant must be a decimal digit in order to avoid confusing a hexadecimal constant with an identifier. (Preceding the hexadecimal number with a zero will always suffice).

A constant composed in this manner must correspond to a binary number that can be contained within a 16-bit counter, otherwise it is truncated on the right by the assembler. As with labels, dollar signs ($) can be imbedded in constants to improve readability. Finally, the radix indicator is translated to upper case if a lower case letter is encountered. The following are valid examples of numeric constants.

```
1234     1234D    1100B    1111$0000$1111$0000B
1234H    0FFEH    33770    33$77$22Q
3377o    0fe3h    1234d    0ffffh
```

## 4.3 Reserved Words

There are several reserved character sequences with pre-defined meanings in the operand field of a statement. Names of 8080 registers are given below. When ASM encounters one of these register names, the numeric value shown in the following table is produced:

| REGISTER NAME | REGISTER VALUE |
|---|---|
| A | 7 |
| B | 0 |
| C | 1 |
| D | 2 |
| E | 3 |
| H | 4 |
| L | 5 |
| M | 6 |
| SP | 6 |
| PSW | 6 |

Lower case names have the same value as their upper case equivalents.

When the symbol "$" occurs in the operand field (not imbedded within identifiers and numeric constants) its value becomes the address of the next instruction to generate, not including the instruction contained within the current logical line.

## 4.4 String Constants

String constants, which are sequences of ASCII characters, are represented by enclosing the characters within apostrophe symbols ('). All strings must be fully contained within the current physical line (thus allowing "!" symbols within strings) and may not exceed 64 characters in length. The apostrophe character can be included within a string by representing it as a double apostrophe (press the apostrophe key two consecutive times), which ASM interprets as a single apostrophe.

In most cases, the string length is restricted to either one or two characters (the DB pseudo operation is an exception). If the string consists of one character, it becomes an 8-bit value. If the string consists of two characters, it becomes a 16-bit value. Two character strings become a 16-bit constant, with the second character as the low order byte, and the first character as the high order byte.

The value of a character is its corresponding ASCII code. There is no case translation within strings. Therefore, both upper and lower case characters can be represented. Only ASCII characters that print are allowed within strings. Valid strings are:

```
'A'    'AB'    'ab'    'c'
'She said "Hello" to me.'
'I said "Hello" to her.'
```

## 4.5 Arithmetic and Logical Operators

The operands described previously can be combined in normal algebraic notation using any combination of properly formed operands, operators, and parenthesized expressions. The operators recognized in the operand field are shown by the following list:

| | |
|---|---|
| a + b | Unsigned arithmetic sum of a and b |
| a − b | Unsigned arithmetic difference between a and b |
| + b | Unary plus (produces b) |
| − b) | Unary minus (identical to 0 − b) |
| a * b | Unsigned magnitude multiplication of a and b |
| a / b | Unsigned magnitude division of a by b |
| a MOD b | Remainder after a / b |
| NOT b | Logical inverse of b (0's become 1's, 1's become 0's), where b is considered a 16-bit value |
| a AND b | Bit-by-bit logical and of a and b |
| a OR b | Bit-by-bit logical or of a and b |
| a XOR b | Bit-by-bit logical exclusive or of a and b |
| a SHL b | The value which results from shifting a to the left by an amount b, with zero fill |
| a SHR b | The value which results from shifting a to the right by an amount b, with zero fill |

In each case, a and b represent simple operands (labels, numeric constants, reserved words, and one or two character strings), or fully enclosed parenthesized subexpressions such as:

```
10+20      10h+37Q      L1/3      (L2+4)      SHR 3
('a' and 5fh) + '0'          ('B' +B) OR (PSW+M)
(1+(2+c)) shr (A−(B+1))
```

All computations are performed at assembly time as 16-bit unsigned operations. Thus, −1 is computed as 0−1 which results in the value 0ffffh (i.e., all 1s). The resulting expression must fit the operation code in which it is used. If the expression is used in an ADI (add immediate) instruction, then the high order eight bits of the expression must be zero. As a result, the operation "ADI −1" produces an error message (−1 becomes 0ffffh, which cannot be represented as an 8-bit value), while ADI (−1) AND 0FFH" is accepted by the assembler since the "AND" operation zeroes the high order bits of the expression.

## 4.6 Precedence of Operators

ASM assumes that operators have a relative precedence of application which allows the programmer to write expressions without nested levels of parentheses. The resulting expression has assumed parentheses which are defined by the relative precedence.

The order of application of operators in unparenthesized expressions is listed below. Operators listed first have highest precedence (they are applied first in an unparenthesized expression), while operators listed last have lowest precedence. Operators listed on the same line have equal precedence, and are applied from left to right as they are encountered in an expression.

```
* / MOD SHL SHR
- +
NOT
AND
OR XOR
```

In the following examples, the expressions shown to the left are interpreted by ASM as the fully parenthesized expressions shown to the right:

| APPEARANCE IN PROGRAM | AS INTERPRETED BY ASM UTILITY |
|---|---|
| a * b + c | (a * b) + c |
| a + b * c | a + (b * c) |
| a MOD b * c SHL d | ((a MOD b) * c) SHL d |
| a OR b AND NOT c + d SHL e | a OR (b AND (NOT(c + (d SHL e)))) |

Balanced parenthesized subexpressions always can be used to override the assumed parentheses. The last expression in the preceding example could be rewritten to force application of operators in a different order, as shown:

```
(a OR b) AND (NOT c) + d SHL e
```

This expression is interpreted by ASM as the following expression with assumed parentheses:

```
(a OR b) AND ((NOT c) + (d SHL e))
```

An unparenthesized expression is well-formed only if the expression resulting from inserting the assumed parentheses is well-formed.

# 5. ASSEMBLER DIRECTIVES

Assembler directives are used to set labels to specific values during the assembly, perform conditional assembly, define storage areas, and specify starting addresses in the program. Each assembler directive is denoted by a "pseudo operation" that appears in the operation field of the line. These pseudo operations are acceptable to ASM:

| Pseudo Operation | Function | Text Subsection |
|---|---|---|
| ORG | Set the program or data origin | (5.1) |
| END | End program, optional start address | (5.2) |
| EQU | Numeric "equate" | (5.3) |
| SET | Numeric "set" | (5.4) |
| IF | Begin conditional assembly | (5.5) |
| ENDIF | End of conditional assembly | (5.6) |
| DB | Define data bytes | (5.7) |
| DW | Define data words | (5.8) |
| DS | Define data storage area | (5.9) |

## 5.1 The ORG Directive

The ORG Directive takes the following form:

Label ORG expression

Where "label" is an optional program label; and

where "expression" is a 16-bit expression, consisting of operands which are defined previous to the ORG statement.

The assembler begins machine code generation at the location specified in the expression. There can be any number of ORG statements within a particular program. There are no checks to ensure that the programmer is not defining overlapping memory areas. Most programs written for the CP/M operating system begin with an ORG statement of:

ORG 100H

Machine code generation begins at the base of the CP/M transient program area (hexadecimal address 100H). If a label is specified in the ORG statement, then the label is given the value of the expression. (This label can then be used in the operand field of other statements to represent this expression.)

## 5.2 The END Directive

The END Directive is optional in an assembly language program. If it is present, it should be the last statement. All subsequent statements will be ignored by ASM.

The two forms of the END directive are:

label END

label END expression

Where the "label" field is optional; and

where the "expression" field is the program starting address.

If the first form is used, the assembly process stops and the default starting address of the program is taken as 0000.

If the second form is used, the expression in the statement becomes the program starting address. (This starting address is included in the last record of the Intel formatted machine code "HEX" file that results from the assembly.)

Most CP/M assembly language programs should end with the statement:

END 100H

resulting in the default starting address of 100H (which is the beginning of the Transient Program Area).

NOTE: When an assembled .HEX file is loaded using the CP/M LOAD utility (included with your CP/M-85 distribution software), use of this directive is not necessary. LOAD produces .COM files that automatically execute from address 100H regardless of any END directive that might exist at the end of a program.

## 5.3 The EQU Directive

The EQU Directive is used to set up synonyms for particular numeric values. It takes the following form:

label EQU expression

Where the "label" field must be present, and must not label any other statement; and

where the "expression" field is assigned to an identifier given in the label field. The identifier is usually a name which describes the value in a more human-oriented manner. This name is used throughout the program to "parameterize" certain functions.

Suppose, for example, that data received from a Teletype appears on a particular input port, and that data is sent to the Teletype through the next output port in sequence. The following series of equate statements could be used to define these ports for a particular hardware environment:

```
TTYBASE  EQU  10H          ;BASE PORT NUMBER FOR TTY
TTYIN    EQU  TTYBASE      ;TTY DATA IN
TTYOUT   EQU  TTYBASE+1    ;TTY DATA OUT
```

At a later point in the program, the statements which access the Teletype could appear as shown:

```
IN       TTYIN      ;READ TTY DATA TO REG-A
. . .
OUT      TTYOUT     ;WRITE DATA TO TTY FROM REG-A
```

This directive makes the program more readable than if the absolute I/O ports had been used.

If the hardware environment is redefined to start the Teletype communications ports at 7FH instead of 10H, the first statement need only be changed to:

```
TTYBASE    EQU  7FH   ;BASE PORT NUMBER FOR TTY
```

Then, the program can be reassembled without changing any other statements.

## 5.4 The SET Directive

The SET Directive takes the following form:

label SET expression

Where the "label" field must be present, and may appear in other SET statements throughout the program; and

where the "expression" field is evaluated by ASM, and becomes the current value associated with the label.

The EQU Directive defines a label with a single value while the SET Directive defines a value which is valid from the current SET statement to the point where the label appears in the next SET statement. The use of the SET is similar to the EQU statement, but SET is most often used in controlling conditional assembly.

## 5.5 The IF Directive

## 5.6 The ENDIF Directive

The IF and ENDIF Directives define a range of assembly language statements which are included or excluded during the assembly process. The form is:

```
If expression
statement#1
statement#2
. . .
statement#n
ENDIF
```

When ASM encounters the IF statement; it evaluates the expression following the IF. (All operands in the expression must be defined ahead of the IF statement). If the expression evaluates to a value where the low order bit (bit 0) is one, then statement #1 through statement #n are assembled. If the expression evaluates to a value where the low order bit (bit 0) of the expression is zero, then the statements are listed but not assembled.

Conditional assembly is used to write a single "generic" program that includes a number of possible run-time environments, with only a few specificportions of the program selected for any particular assembly. The following program segments, might be part of the program that communices with either a Teletype console or CRT console by selecting a particular value for TTY before the assembly begins:

```
TRUE    EQU  0FFFFH          ;DEFINE VALUE OF TRUE
FALSE   EQU  NOT TRUE        ;DEFINE VALUE OF FALSE
;
TTY     EQU  TRUE            ;TRUE IF TTY, FALSE IF CRT
;
TTYBASE EQU  10H             ;BASE OF TTY I/O PORTS
CRTBASE EQU  20H             ;BASE OF CRT I/O PORTS
        IF   TTY             ;ASSEMBLE RELATIVE TO TTYBASE
CONTIN  EQU  TTYBASE         ;CONSOLE INPUT
CONOUT  EQU  TTYBASE+1       ;CONSOLE OUTPUT
        ENDIF
;
        IF   NOT TTY         ;ASSEMBLE RELATIVE TO CRTBASE
CONTIN  EQU  CRTBASE         ;CONSOLE INPUT
CONOUT  EQU  CRTBASE+1       ;CONSOLE OUTPUT
        ENDIF
        . . .
        IN   CONTIN          ;READ CONSOLE DATA
        . . .
        OUT  CONOUT          ;WRITE CONSOLE DATA
```

In this case, the program would assemble for an environment where a Teletype is connected, based at port 10H. The statement defining TTY could be changed to:

```
TTY     EQU  FALSE
```

and, in this case, the program would assemble for a CRT based at port 20H.

## 5.7 The DB Directive

The DB Directive allows the programmer to define initial storage areas in single-byte format. The statement appears in the form:

label DB e#1,e#2, ...,e#n

Where "e#1" through "e#n" are either:

- Expressions which evaluate to 8-bit values (the high order eight bits must be zero), or
- ASCII strings of fewer than 65 characters.

There is no practical restriction on the number of expressions that can be included on a single source line. The expressions are evaluated and placed sequentially into the machine code file following the last program address generated by ASM. String characters are similarly placed into memory, starting with the first character and ending with the last character.

Strings containing more than two characters cannot be used as operands in more complicated expressions. They must stand alone between the commas. ASCII characters are always placed in memory with the parity bit reset (to 0). There is no translation from lower to upper case with strings. The optional label can be used to reference the data area throughout the remainder of the program. Examples of valid DB statements are:

```
data:   DB      0,1,2,3,4,5
        DB      data and 0ffh,5,377Q,1+2+3+4
signon: DB      'please type your name',CR1f,0
        DB      'AB' SHR 8, 'C', 'DE' and 7FH
```

## 5.8 The DW Directive

The DW Directive lets you define initial storage areas in double-byte words. The statement appears in the form:

```
label    DW    e#1,  e#2,  ..., e#n
```

Where "e#1" through "e#n" are expressions which ASM evaluates in 16-bit results.

ASCII strings are limited to one or two characters. Data storage is consistent with the 8080 processor. The least significant byte of the expression is stored first in memory, followed by the most significant byte. The following are examples of DW usage:

```
doub:   DW      Offefh,doub+4,signon-$,255+255
        DW      'a', 5,  'ab',  'CD',  6 sh1  8 or 11b
```

## 5.9 The DS Directive

The DS Directive is used to reserve an area of uninitialized memory. This directive takes the following form:

```
label    DS  expression
```

Where the label is optional.

The assembler begins subsequent code generation after the area reserved by the DS. The DS statement given previously has exactly the same effect as the following statement:

```
label:  EQU   $  ;LABEL VALUE IS CURRENT CODE LOCATION
        ORG   $+expression   ;MOVE PAST RESERVED AREA
```

# 6. ASM ERROR MESSAGES

The ASM utility has two kinds of error messages. One signals problems that occur during ASM file manipulation (6.1), and another signals errors in the assembly language source file (6.2).

## 6.1 File Manipulation Error Messages

The following error messages are displayed when ASM encounters difficulty in manipulating the source and/or output files involved in operation of the ASM utility:

```
NO SOURCE FILE PRESENT
```

EXPLANATION: The file specified in the ASM command does not exist on the accessed disk.

```
NO DIRECTORY SPACE
```

EXPLANATION: The disk directory is full. Unnecessary files should be erased, and the operation attempted again.

```
SOURCE FILE NAME ERROR
```

EXPLANATION: Improperly formed ASM file name (such as a name specified with "?" fields) was entered.

```
SOURCE FILE READ ERROR
```

EXPLANATION: The source file cannot be read properly by the assembler. The TYPE command, a text editor, or DDT can be used to determine the location of the unreadable code.

```
OUTPUT FILE WRITE ERROR
```

EXPLANATION: Output files cannot be written properly. The most likely cause is a full disk. Unnecessary files should be erased, and the operation attempted again.

```
CANNOT CLOSE FILE
```

EXPLANATION: Output file cannot be closed. Disk should be checked for write protection.

```
SYMBOL TABLE OVERFLOW
```

EXPLANATION: The symbol table has exceeded memory capacity.

## 6.2 Assembly Program Error Messages

When errors occur within the assembly language program, they are listed as single character flags in the left-most position of the source listing. The line in error is also echoed in a terminal display so that the source listing need not be examined to determine if errors are present. The error codes are:

D Data error: Element in data statement cannot be placed in the specified data area.

E Expression error: Expression is ill-formed and cannot be computed at assembly time.

L Label error: Label cannot appear in this context (may be duplicated label).

N Not implemented: Features which will appear in future ASM versions (e.g., macros) are recognized, but flagged in this version.

O Overflow: Expression is too complicated (i.e., too many pending operators) to compute, simplify it.

P Phase error: Label does not have the same value on two subquent passes through the program.

R Register error: The value specified as a register is not compatible with the operation code.

V Value error: Operand encountered in expression is improperly formed.

# BSYSGEN

*The Utility that Copies CP/M Between Disks*

The BSYSGEN utility is used to transfer either part or all of the CP/M operating system to a disk, depending on the circumstances. Unlike the SYSGEN utility (see Page 2-193) the BSYSGEN utility can **not** be used to copy the system kernel directly from memory to a disk after running the MVCPM207 utility, although it can copy a file that was recorded on a disk by the SAVE command after a run of MVCPM207.

NOTE: This release of the CP/M Operating System consists of a system kernel and the BIOS files (BIOS85.SYS and BIOS88.SYS). To make a disk bootable, you must put the system kernel on the disk's system tracks **and** the BIOS85.SYS and BIOS88.SYS files on the disk's file area.

BSYSGEN can be used by two methods: the Utility Prompt Method or the System Prompt Method.

## 1. UTILITY PROMPT METHOD

Under the Utility Prompt Method, you first load the BSYSGEN utility into computer memory, and then respond to BSYSGEN prompts that define the operation.

### 1.1 Utility Prompt Command Entry

To begin under the Utility Prompt Method, type the following command at the system prompt:

```
A>BSYSGEN (CR)
```

The following display will appear:

```
BSYSGEN VER 2.2.100
SOURCE DRIVE NAME :
```

## 1.2 Specifying the Source

At the "SOURCE DRIVE NAME:" prompt, you can specify the drive containing the disk from which the system will be copied. Enter the letter that stands for that drive.

The following example shows how you would answer this prompt if the source of the system was to be the disk in drive A:

```
SOURCE DRIVE NAME: A
```

NOTE: BSYSGEN can only copy the system between disks of the same type. Therefore, you can **not** enter a carriage return at this BSYSGEN prompt to copy a system that has been moved into computer memory by the MVCPM207 utility. (If you do wish to copy the system from memory immediately after a MVCPM207 activity, use the SYSGEN utility.)

BSYSGEN will now prompt you to confirm your selection of the source drive, with a prompt in the following form:

```
SOURCE ON A, THEN TYPE RETURN:
```

You can confirm your specification of the source drive name by entering a carriage return at this prompt. (You can also abort the BSYSGEN operation and return control to the operating system by simultaneously pressing the **CTRL** and **C** keys at this prompt.)

If you confirm the "SOURCE ON" prompt with a carriage return, BSYSGEN will then display the message:

```
COPY BIOS88.SYS & BIOS85.SYS (Y/N):
```

## 1.3 Copying BIOS88.SYS and BIOS85.SYS with BSYSGEN

To instruct BSYSGEN to copy the files BIOS88.SYS and BIOS85.SYS from the source disk to the destination disk, press **Y** at the "COPY BIOS88.SYS & BIOS85.SYS (Y/N):" prompt. If you do **not** wish to copy BIOS88.SYS and BIOS85.SYS, press **N**.

NOTE: If you decline to copy the BIOS files using the BSYSGEN utility, you can copy them using the PIP utility.

If you pressed Y to copy the BIOS files, BSYSGEN will display the message "FUNCTION COMPLETE", and then prompt for destination. If you pressed N to forgo the copying of the BIOS files, BSYSGEN will immediately prompt for destination.

## 1.4 Specifying the Destination

A few seconds after you have made an entry at the "COPY BIOS88.SYS & BIOS85.SYS" prompt, BSYSGEN will prompt for destination as shown:

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

The first time this prompt appears, you should type the drive letter for the disk that you wish to receive the system. For instance, type **B**. BSYSGEN would then display a prompt in the following form:

```
DESTINATION ON B, THEN TYPE RETURN
```

Confirm your destination specification by entering a carriage return at such a prompt.

BSYSGEN will again display the "DESTINATION DRIVE NAME (OR RETURN TO REBOOT):" prompt. This time, you can specify a different drive name, insert a new disk into the former destination drive and specify this drive again as the destination, or enter a carriage return to cause a warm boot. (A warm boot will exit you from the BSYSGEN utility to the CP/M Operating System. Then a system prompt will be displayed.)

# 2. SYSTEM PROMPT METHOD

The System Prompt Method, enables you to enter all of the specifications necessary for a BSYSGEN operation in a single command line entered at the CP/M system prompt.

## 2.1 System Prompt Command Entry

System Prompt Method BSYSGEN commands are entered in the following form:

A>**BSYSGEN {destination}={source}{[option,option]}**CR

Where **BSYSGEN** is the command line function, stored in the file BSYSGEN.COM on the logged disk;

where {**destination**} is the name of the drive (A:, B:, C:, or D:) containing the formatted disk that you wish to receive the copy of the system;

where {**source**} can be either a drive name, a file name, or both; and

where {**[option,option]**} represents letters enclosed in square brackets [ ] and separated by a comma , to specify how the BSYSGEN operation should be conducted.

NOTE: In a CP/M command line "equation", the data source is always on the right and the data destination is always on the left.

## 2.2 BSYSGEN Data Sources

The source of the transferred data in a System Prompt Method command can be one of the following four types:

- Drive Name, including a letter for a drive within your hardware environment and a colon, as with **A:**, **B:**, **C:**, or **D:**;
- File Name, which specifies a file that was created and stored by consecutive MVCPM207 and SAVE commands, as with **CPM48.SYS** or **CPM64.SYS**; or
- Drive Name and File Name, where the file desired for the system kernel source resides on a disk in a non-default drive and the drive name must specify that drive, as with **B:CPM48.SYS** or **C:CPM64.SYS**.

## 2.3 BSYSGEN Options and Defaults

BSYSGEN command lines entered by the System Prompt Method can include the following options (enclosed in square brackets [ ] ):

**B** BIOS88.SYS and BIOS85.SYS files will be copied from the specified source to the specified destination. If files named BIOS88.SYS and BIOS85.SYS already exist on the destination disk, they will be overwritten.

**N** No prompts will be displayed during this operation.

When you enter a BSYSGEN command line with source and destination specifications, and neglect to specify options, BSYSGEN will perform the operation according to these default criteria:

- BIOS88.SYS and BIOS85.SYS files will **not** be copied (as if the B option was not specified);
- Prompt **will** be displayed to confirm which drive will receive the copy of the system (as if option N was not entered). The BSYSGEN prompt displays in the following form:

```
BSYSGEN VER 2.2.100

DESTINATION ON B, THEN TYPE RETURN
```

## 2.4 System Prompt Method Examples

A>**BSYSGEN B:=A:** ⓒⓡ

BSYSGEN will copy the system core from the disk in drive A to the disk in drive B. The BIOS files from A will **not** be copied and a prompt **will** appear before the copying, by default.

A>**BSYSGEN B:=D:[N]** ⓒⓡ

BSYSGEN will copy the system core from the disk in drive D to the system tracks of the disk in drive B. The BIOS files will **not** be copied, by default. A prompt will **not** appear before the copying, as specified by the N option.

A>**D:BSYSGEN B:=C:CPM48.SYS[B,N]** ⓒⓡ

The BSYSGEN utility in this case, is stored on the disk in non-default drive D. It will copy the system core from the file named "CPM48.SYS" (recorded onto the disk in drive C by the SAVE command after creation in memory by the MVCPM207 command), and put it on the system tracks of the disk in drive B. It will also copy the BIOS files from drive C to drive B, and display no prompts during the operation, as specified by options.

# 3. BSYSGEN ERROR MESSAGES

```
INVALID DRIVE NAME
```

EXPLANATION: User must specify drive names using the names of drives that exist in the hardware environment, and are recognized by the operating system that was loaded at bootstrap.

```
NO SOURCE FILE ON DISK
```

EXPLANATION: The drive specified as "SOURCE DRIVE" did not contain one of the BIOS files (BIOS88.SYS or BIOS85.SYS). Use a different disk in the source drive, or copy working BIOS files to the source disk, or rename existing BIOS files to "BIOS88.SYS" and "BIOS85.SYS".

```
SOURCE FILE INCOMPLETE
```

EXPLANATION: BSYSGEN failed in an attempt to copy the BIOS files from the disk in the source drive. This file might have been damaged by disk media flaws or partially overwritten. The user should reset, perform bootstrap, and reenter the BSYSGEN command using a different disk in the source drive.

```
WRITE ERROR DURING BIOS.SYS
```

EXPLANATION: The user should try BSYSGEN again with a destination disk that is write-enabled, formatted, and has at least 6 kilobytes of free space.

```
ERROR READING BIOS.SYS
```

EXPLANATION: BSYSGEN failed in an attempt to copy the file BIOS.SYS from the disk in the source drive. This file might have been damaged by disk media flaws or partially overwritten. The user should reset, perform bootstrap, and reenter the BSYSGEN command using a different disk in the source drive or using a different disk to perform bootstrap.

```
PERMANENT ERROR, TYPE RETURN TO IGNORE
```

EXPLANATION: The system kernel or BIOS files are either incompatible with the destination disk type or otherwise flawed. The user should reset, reboot, and reenter the BSYSGEN command using a different disk in the source drive or using a different disk to boot up. Under some circumstances, the user must use the MVCPM207 utility before BSYSGEN.

```
UNABLE TO SELECT DRIVE
```

EXPLANATION: Specify the name of a drive that can be accessed by BSYSGEN. Such a drive must be a valid drive that is recognized by the operating system.

```
COMMAND SYNTAX ERROR
```

EXPLANATION: System Prompt Method command line was entered without following the entry form explained in "2.1 Command Line Entry". Enter command again after reviewing this entry form.

```
ILLEGAL OPTION
```

EXPLANATION: System Prompt Method command line was entered with an option other than a B or an N. Reenter command with either, none, or all of the BSYSGEN options B and N. Enclose the option(s) in square brackets and separate them with a comma if both are used.

# CONFIGUR

*The Utility that Customizes CP/M for Your Hardware and/or Preferences*

The CONFIGUR utility helps you to change the CP/M Operating System so that it will accommodate a particular printer (3) or modem (4). CONFIGUR also enables you to set the system to automatically invoke commands (5) upon cold boots and/or warm boots, and to assign physical devices to logical devices (6). After you have specified these changes to the system, CONFIGUR enables you to apply them to the system in memory and/or the system on disk, or to cancel them completely (7).

CONFIGUR is usually run during the first session of CP/M use in a particular hardware environment. But it should also be run whenever a hardware component is added or changed, or whenever you wish to change an automatic command line.

You do **not** need to use the CONFIGUR activity if you do not have a printer or modem, or if you have one of the following:

- A serial printer (such as the Z-25 or the H-25) that runs at 4800 baud, accepts 8 bits per character with no parity bit, handshakes with RTS pin number 4, is ready when handshaking signal is High, and uses no protocol.

- A modem (such as the WH-13, the Lexicon WH-23, the UDS WH-33, or the Hayes WH-43) that runs at 300 baud, accepts 8 bits per character, with no parity bit, and uses no handshaking.

These hardware settings match the default settings of the CP/M-85 system when it is shipped.

Since CONFIGUR adjusts the system for hardware characteristics, you must sometimes answer prompts that ask about these characteristics. Always consult your hardware manuals and check the settings of your hardware devices when answering these prompts.

NOTE: Changes specified through the CONFIGUR utility can only be recorded on a disk if the disk is write-enabled. If you are performing a CONFIGUR operation with a write-protected disk, you can only apply changes to the system in memory.

# 1. CONFIGUR INVOCATION

You can invoke CONFIGUR by responding to the system prompt with a command line in the following form:

A>**CONFIGUR** ⏎

This entry causes the display of CONFIGUR's identification message, copyright notice,version number, and "MAIN MENU."

# 2. CONFIGUR MAIN MENU

CONFIGUR is a menu-driven utility. It first displays a main menu, which looks like this:

```
CP/M-85 System Configuration Utility version 2.2.100
      Copyright (C) 1982 by Zenith Data Systems

                *** MAIN MENU ***


           P - Printer Configuration
           M - Modem   Configuration
           C - Command Configuration
           I - I/O map Configuration
           ? - Brief Help message

           X - Exit

Selection [P,M,C,I,X or ?] :
```

This menu enables you to access any of four sub-menus so that you can adjust the CP/M Operating System to accommodate your printer, modem, or automatic command preference. In addition, you can obtain screen displays of helpful comments about the menu and its use. This menu also enables you to exit the CONFIGUR utility, and thus gain access to the CP/M Operating System.

To enter a particular sub-menu or exit selection, type the character listed to the left of that selection and press ⏎.

# 3. PRINTER SPECIFICATION

In order to adjust the system to accommodate your printer, you must select "Printer Configuration" at the main menu, and specify your printer either by its name or by its characteristics.

When you enter **P** and ⓒR at the main menu, CONFIGUR displays the following sub-menu:

```
                *** Printer Configuration ***

 1-   MX-80 or other PARALLEL Centronics-interface printer
 2-   H/Z-25
 3-   H-14 or TI-810(WH-24)
 4-   Dec LA-34 or LA-36
 5-   Diablo 620
 6-   Diablo 630,1610,1620,1630 or 1640(WH-44)
 7-   MX-80 Serial
 8-   Votrax Type 'n Talk
 9-   User-defined SERIAL Printer

Choose the number that corresponds to your printer :
```

- If your printer is listed by name in this menu, and you have not changed the switch settings since it was shipped, then see "3.1 Specifying Printer Name".
- If you have a printer that is not listed by name on this menu or a listed printer on which the switch settings have been changed since shipping, then see "3.2 Specifying Printer Characteristics".

## 3.1 Specifying Printer Name

If your printer is listed by name on the "Printer Configuration" menu, and you have not changed the switch settings since it was shipped, then type the number listed to the left of the printer name and a carriage return.

This entry will adjust the CP/M Operating System to accommodate the characteristics that usually apply to your type of printer when it is shipped.

CONFIGUR then displays a message describing the characteristics of the printer you selected. The printer characteristics descriptions for each menu-listed printer are shown beginning on the following page.

NOTE: If you have a printer that is not listed by name on this menu, or a listed printer on which the switch settings have been changed since shipping, then type **9** and carriage return, and see "3.2 Specifying Printer Characteristics".

Each printer you select in response to the "Printer Configuration" menu has different characteristics, which are described in the message CONFIGUR displays after your selection.

### MX-80 or Centronics

```
You have selected a MX-80 Parallel, and Centronics

   Assure Centronics style parallel operation
Press RETURN to access Main menu:
```

### H/Z-25

```
You have selected a H/Z-25


Standard settings:

   4800   baud
  Handshake on RTS
  Printer ready on high

Press RETURN to access Main menu:
```

### H-14 or TI-810

```
You have selected a H-14, TI-810(WH-24)


Standard settings:

   4800   baud
  Handshake on RTS
  Printer ready on low

Press RETURN to access Main menu:
```

### Dec LA-34 or LA-36

```
You have selected a DEC LA-34 or an LA-36


Standard settings:

    300   baud

Press RETURN to access Main menu:
```

### Diablo 620

```
You have selected a Diablo 620


Standard settings:

    300   baud
  ETX/ACK protocol

Press RETURN to access Main menu:
```

### Diablo 630 or 1640

```
You have selected a Diablo 630,1610,1620,1630 and 1640


Standard settings:

   1200   baud
  ETX/ACK protocol

Press RETURN to access Main menu:
```

### MX-80 Serial

```
You have selected a MX-80 with serial interface


Standard settings:

   4800   baud
  Handshake on DTR
  Printer ready on low

Press RETURN to access Main menu:
```

### Votrax Type 'n Talk

```
You have selected a Votrax Type 'n Talk


Standard settings:

   4800   baud
  Handshake on RTS
  Printer ready on high

Press RETURN to access Main menu:
```

After viewing the message that lists your printer's characteristics, press ⓒ. The "MAIN MENU" will be redisplayed.

- If the settings of your printer match those listed in the message for your printer, then enter the letter for another selection.
- If you specified your printer by name at the "Printer Configuration" menu, but want this printer set with characteristics that differ from those listed for your printer by CONFIGUR, type **P** and ⓒ again at the "MAIN MENU". Then type **9** and ⓒ at the "Printer Configuration" menu and see "3.2 Specifying Printer Characteristics".

NOTE: Printers selected by name at the "Printer Configuration" menu have their appropriate physical device assigned automatically to the LST: logical device.

# 3.2 Specifying Printer Characteristics

To adjust the system for a printer that is not listed by name on this menu, or a listed printer on which the switch settings have been changed since shipping, type **9** and ⓒⓡ at the "Printer Configuration" menu.

CONFIGUR will then prompt you to specify characteristics of your printer such as: baud rate, bits per character, parity, handshaking pin, handshake polarity and protocol.

NOTE: If you respond to one of these prompts by pressing ⓒⓡ without first typing one of the values listed in the prompt, the prompt will be redisplayed.

### BAUD RATE

When you enter **9** and ⓒⓡ at the "Printer Configuration" menu, CONFIGUR will first display the following message:

```
                User Defined Printer

 Baud rate selection
(Space=next possibility,
 BACK SPACE=last selection,
 RETURN=select this one)
Baud Rate :    45.5
```

### Explanation

"Baud rate" is the speed at which data is transmitted to and from your printer, roughly corresponding to the number of bits per second transmitted. Since the CP/M Operating System coordinates the transmission of data to and from your printer, CP/M must know how fast your printer is set to send and receive data.

### Specification Technique

To the right of the words "Baud Rate :" in this prompt, you will see a number that corresponds to one of the 16 baud rates possible on the Z-100 computer (45.5, 50, 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2000, 2400, 4800, 9600, 19200, 38400).

Each of these baud rate values is stored in sequence by CONFIGUR, although only one value is visible at one time. You can view a different value by pressing the **space bar** for a greater value, or by pressing the **BACK SPACE** key for a lesser value.

When the correct baud rate value is displayed in the "Baud Rate :" prompt, press ⏎ to specify this value. CONFIGUR will get ready to set the CP/M Operating System to accommodate this baud rate value for printer data transmission.

## BITS PER CHARACTER

After you have specified a baud rate, CONFIGUR will display the following prompt:

```
Bits per character: [5,6,7 or 8] :
```

### Explanation

"Bits per character" is the number of significant data bits your printer expects to receive in order to decipher one character (byte) from the stream of bits that are transmitted. Start bits, stop bits, and parity bits are not included in this number. (Most printers accept 7 or 8 bits per character.)

### Specification Technique

To specify the number of bits per character your printer accepts, type **5, 6, 7,** or **8** at the "Bits per character" prompt. Then press ⏎. CONFIGUR will get ready to set the CP/M Operating System to transmit this many bits within every character of data it sends.

## PARITY

After you have specified bits per character, CONFIGUR will display the following prompt:

```
Parity [O(dd),E(ven),N(one)] :
```

### Explanation

"Parity" is a method by which data is checked to make sure it hasn't changed during transmission.

When odd parity is used, a parity bit is sent along with each character that is sent to the printer. Before transmission, this bit is set to either one or zero to ensure that the sum of all of the transmitted bits is an odd number. If the printer receives a byte of data bits and a parity bit that do not all add up to an odd number, then an error must have occurred during transmission.

Even parity works the same way, except that the parity bit is set to either one or zero to ensure that the sum of the transmitted bits is an even number.

When no parity is used, no parity bit accompanies each transmitted character.

### Specification Technique

To specify the kind of parity (if any) that is used to check errors in the data sent to your printer, type **O** for odd parity, **E** for even parity, and **N** for no parity. After typing one of these specifications, press ⏎. CONFIGUR will get ready to set the CP/M Operating System so that it transmits parity bits as required by your printer.

## HANDSHAKE PIN

After you have specified the nature of your printer's parity, CONFIGUR will display the following prompt:

```
Handshake Pin [N(one), D(tr/pin 20), R(ts/pin 4)] :
```

### Explanation

The "handshake pin" is the circuit through which the printer and the computer signal each other to determine when data should be transmitted. This circuit is one of many bundled together in the RS-232C cable that connects the computer with the printer. The printer uses this cable circuit to signal that the printer is ready to receive more data.

The DTR (Data Terminal Ready) handshake pin is labelled with the number 20 on the receptacle ends of the cable; and the RTS (Request To Send) handshake pin is labelled with the number 4 on the receptacle ends of the cable.

### Specification Technique

To specify the particular handshake pin (if any) that is used to signal the printer's readiness to receive data, type **N** for no pin, type **D** for the DTR (number 20) pin, or type **R** for the RTS (number 4) pin. After typing one of these specifications, press ⏎. CONFIGUR will get ready to set the CP/M Operating System so that it signals the printer's readiness as required by your printer.

## POLARITY

If you specified a handshake pin (by typing D or R at the previous prompt), then a prompt will appear in the following form:

```
Polarity [N(one),H(ready when High),L(ready when Low) ] :
```

NOTE: If you specified no handshake pin (by typing N at the previous prompt), then the "Polarity" prompt will not appear. Instead, the handshake pin prompt will be followed by a protocol prompt.

### Explanation

"Polarity" is the voltage level at which the printer signals the computer that the printer is ready to receive more data. This signal can be sent at either a high voltage level or a low voltage level. It travels through the handshake pin that was specified at the previous prompt.

### Specification Technique

To specify a polarity, type **H** if the printer signals its readiness with a high voltage signal, and type **L** if the printer signals its readiness with a low voltage signal. To specify no polarity, type **N**.

## PROTOCOL

After you have answered the handshake pin prompt (and possibly the polarity prompt), CONFIGUR will display the following prompt:

```
Protocol [X(on/Xoff),E(tx/Ack),N(one)] :
```

### Explanation

"Protocol" is a method of coordinating the transmission of data between your computer and your printer by assigning one device or both devices to transmit specific characters as a signal to the other device.

With XON/XOFF protocol, the printer transmits the DC3 character (Device Control 3: turns transmitter off) when the printer must momentarily stop receiving data, either because the buffer is full or because the printer is "off line". The printer sends the DC1 character (Device Control 1: turns transmitter on) when the printer is ready to receive more data, as when the buffer has empty space and the printer is "on line".

With ETX/ACK protocol, the computer transmits the ETX character at the end of the transmission of a unit of data (to signify the End of TeXt). When the printer processes the unit of data (buffer size) with the ETX character, the printer transmits the ACK character back to the computer (to ACKnowledge that the data unit was received). If this option is selected, you will be asked to enter the device buffer size. Enter a number between 10 and 255 to reflect the largest amount of data that your printer can handle.

### Specification Technique

To specify the particular protocol characters (if any) that are transmitted to signal the printer's readiness to receive more data, type **X** for XON/XOFF protocol, type **E** for ETX/ACK protocol, and type **N** for no protocol. After typing one of these letters, press ⏎. CONFIGUR will get ready to set the CP/M Operating System so that it allows transmission of the proper signal characters, if any.

After you have specified a protocol, CONFIGUR will display:

```
Press RETURN to access Main Menu:
```

Press ⏎ and CONFIGUR will redisplay the "MAIN MENU", at which you can enter another selection.

# 4. MODEM SPECIFICATION

In order to adjust the system to accommodate your modem, enter **M** and ⓒ at the "Configur Main Menu". Then CONFIGUR will display one or more prompts that help you to specify either the model number or characteristics of your modem. The first modem prompt looks like this:

```
*** Modem Configuration ***
Standard Heath/Zenith Modem? (WH-13,WH-23,WH-33,WH43) [Y/N] :
```

Your response to this prompt depends on whether your modem is listed in the prompt by its model number. If not, you must specify individual characteristics of your modem.

## 4.1 Specifying Modem Model Number

If the model number of your modem (as shown in the Heath/Zenith catalog) is listed in this prompt, and you have not changed the switch settings since it was shipped, then type **Y** and press ⓒ. This entry gets CONFIGUR ready to adjust the CP/M Operating System to accommodate the characteristics that usually apply to these modems when they are shipped. After this entry, CONFIGUR will redisplay the "MAIN MENU". Then you can enter the letter for another selection.

## 4.2 Specifying Modem Characteristics

If you have a modem that is not listed in this prompt, or if you have a listed modem on which the switch settings have been changed since shipping, then type **N** and ⓒ at this prompt. CONFIGUR will then enable you to specify characteristics of your modem such as: baud rate, bits per character, parity, and handshaking pin. First CONFIGUR displays a prompt for baud rate selection.

NOTE: If you respond to one of these prompts by pressing ⓒ without first typing one of the values listed in the prompt, the prompt will be redisplayed.

## BAUD RATE

When you enter the letter **N** at the "Modem Port Selection" menu, CONFIGUR will first display the following message:

```
  Baud rate selection
(Space=next possibility,BS=last,CR=select this one)
   Baud Rate :    45.5
```

### Explanation

Baud rate is the speed with which data is transmitted to and from your modem, roughly corresponding to the number of bits per second transmitted. Since the CP/M Operating System coordiantes the transmission of data to and from your modem, CP/M must know how fast your modem is set to send and receive data.

### Specification Technique

This prompt enables you to specify the baud rate of your modem. To the right of the words "Baud Rate :", you will see a number that corresponds to one of the 16 baud rates possible on the Z-100 computer (45.5, 50, 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2000, 2400, 4800, 9600, 19200, 38400).

CONFIGUR displays these values, one-at-a-time, in sequence. You can view a different value by pressing the **space bar** for a greater value, or by pressing the **BACK SPACE** key for a lesser value.

When the correct baud rate value is displayed in the "Baud Rate :" prompt, press ⓒⓡ to specify this value. CONFIGUR will get ready to set the CP/M Operating System to accommodate this baud rate value for modem data transmission.

## BITS PER CHARACTER

After you have specified a baud rate, CONFIGUR will display the following prompt:

```
Bits per character: [5,6,7 or 8] :
```

### Explanation

"Bits per character" is the number of significant data bits your modem expects to receive in order to decipher one character (byte) from the stream of bits that are transmitted. Start bits, stop bits, and parity bits (if used) are not included in this number.

### Specification Technique

To specify the number of bits per character your modem accepts, type **5**, **6**, **7**, or **8** at the "Bits per character" prompt. Then press ⏎. CONFIGUR will get ready to set the CP/M Operating System to transmit this many bits within every character of data it sends.

## PARITY

After you have specified bits per character, CONFIGUR will display the following prompt:

```
Parity [O(dd),E(ven),N(one)] :
```

### Explanation

"Parity" is a method by which data is checked to make sure it hasn't changed during transmission.

When odd parity is used, a parity bit is transmitted along with each character that is transmitted to the modem. Before transmission, this bit is set to either one or zero to ensure that the sum of all of the transmitted bits is an odd number. If the modem receives a byte of data bits and a parity bit that do not all add up to an odd number, then an error must have occurred during transmission.

Even parity works the same way, except that the parity bit is set to either one or zero to ensure that the sum of all of the transmitted bits is an even number.

When no parity is used, no parity bit accompanies each transmitted character.

### Specification Technique

To specify the kind of parity (if any) that is used to check errors in the data sent to your modem, type **O** for odd parity, **E** for even parity, and **N** for no parity. After typing one of these specifications, press ⓒⓡ. CONFIGUR will get ready to set the CP/M Operating System so that it transmits parity bits as required by your modem.

## HANDSHAKE PIN

After you have specified the nature of your modem's parity, CONFIGUR will display the following prompt:

```
Handshake Pin [N(one), D(tr/pin 20), R(ts/pin 4(] :
```

### Explanation

The "handshake pin" is the circuit through which the modem and the computer signal each other to determine when data should be transmitted. This circuit is one of many bundled together in the RS-232C cable that connects the computer with the modem. The modem uses this circuit to signal the computer that it is ready to receive more data.

The DTR (Data Terminal Ready) handshake pin is labelled with the number 20 on the receptacle ends of the cable; and the RTS (Request To Send) handshake pin is labelled with the number 4 on the receptacle ends of the cable.

### Specification Technique

To specify the particular handshake pin (if any) that is used to signal the modem's readiness to receive data, type **N** for no pin, type **D** for the DTR (number 20) pin, or type **R** for the RTS (number 4) pin. After typing one of these specifications, press ⏎. CONFIGUR will get ready to set the CP/M Operating System so that it signals the modem's readiness as required by your modem.

Then CONFIGUR will redisplay the "MAIN MENU", at which you can enter the letter for another selection.

# 5. AUTOMATIC COMMAND SPECIFICATION

In order to adjust the system to accommodate your preference in automatic commands, you must type **C** and ⓒ to select "Command Configuration" at the main menu. CONFIGUR will display the following menu:

```
                *** Command Line Configuration ***

      C - Cold Boot Command Line =
      W - Warm Boot Command Line =
      ? - Brief Help Message

      X - Exit

    Selection [C,W, ? or X]:
```

This menu enables you to specify that a command be automatically invoked upon a cold boot, or that a command be automatically invoked upon a warm boot. You can specify up to two different commands: one for execution on cold boot, and one for execution on warm boot. In addition, this menu enables you to display a screen of helpful comments about this menu and its use, if you enter a **?**.

## 5.1 Cold Boot Command Line

To specify a command for automatic cold boot invocation, type **C** and ⓒ at the "Command Line Menu". CONFIGUR will display the prompt:

```
Cold Boot Command Line :
```

At this prompt type a valid CP/M command line, just as you would type it at a CP/M system prompt. When you enter the ⓒ to end the command line, CONFIGUR will display the command line on the "Command Line Menu" and get ready to set the CP/M Operating System to invoke this command immediately after every cold boot.

If the menu already shows a cold boot command line and you wish to remove it, type **C** at the "Command Line Menu" and immediately press ⓒ.

NOTE: Your Z-100 is capable of performing an automatic cold boot when powered up or reset, if a switch within the Z-100 is set properly. If this automatic booting feature is not switched on, you must power up or reset the computer and then enter a bootstrap command to perform a cold boot. In either case, the cold boot will be immediately followed by invocation of a command specified through CONFIGUR.

## 5.2 Warm Boot Command Line

To specify a command for automatic warm boot invocation, type **W** and ⓒⓡ at the "Command Line Menu". CONFIGUR will display the prompt:

```
Warm Boot Command Line :
```

At this prompt, type a valid CP/M command line, just as you would type it at a CP/M system prompt. When you enter the ⓒⓡ to end the command line, CONFIGUR will display the command line on the "Command Line Menu" and get ready to set the CP/M Operating System to invoke this command immediately after every warm boot.

If the menu already shows a warm boot command line and you wish to remove it, type **W** at the "Command Line Menu" and immediately enter a carriage return.

NOTE: A warm boot can occur when you press the **CTRL** and **C** keys simultaneously, or when some application programs exit to the operating system. In either case, the warm boot will be immediately followed by invocation of a command specified through CONFIGUR.

## 5.3 Acceptable Commands

Any valid resident command, transient command (utility), or application program is acceptable in the automatic command line. However, any file that the command line refers to must reside on the disk specified in the command line. For example, if the line reads:

```
C - Cold Boot Command Line = B:SC
```

Then the file "SC.COM" (part of the SuperCalc application program) must reside on the disk in drive "B:" for the command to work. If any data referenced in the command line cannot be found in the specified drives, then command execution will be aborted and an error message will be displayed.

NOTE: Some transient commands (utilities) and application programs finish their execution by performing a warm boot. Therefore we recommend that you do not enter both an automatic cold and warm boot command line for a program that performs a warm boot after execution, because endless execution of the same program might result.

## 5.4 Returning to the Main Menu

To finish your command line menu activities and regain access to the "Configur Main Menu", type **X** at the "Command Line Menu". CONFIGUR will redisplay the "MAIN MENU", which enables you to enter the letter for another selection.

# 6. I/O DEVICE SPECIFICATION

In order to adjust the system to send certain types of data through different physical devices, enter **I** and ㊮ at the "Configur Main Menu". Then CONFIGUR will display a menu in the following form:

```
            ***I/O Map Configuration***

    C - Console     Currently: CRT
    R   Reader      Currently: PTR
    P - Punch       Currently  PTP
    L - List        Currently: TTY
    ? - Brief Help Message

    X    Exit

  Selection [C R P L ? or X] :
```

(The current assignments of your physical devices may differ from those in this sample menu.)

This menu shows the physical device names that are currently assigned to each logical device category.

NOTE: If you have already chosen a printer (by name or by characteristics) during this CONFIGUR run, then List device will have been automatically assigned to the proper physical device. Therefore, the letter "L" will not **not** be listed in the I/O map selection prompt. Furthermore, you will not be able to select the List device during this CONFIGUR run.

**Explanation**

Data can be sent to or from a wide variety of peripheral hardware devices. CP/M groups these devices under the logical device categories "Console", "Reader", "Punch", and "List" (as shown on the left side of the menu).

However, these categories provide CP/M with only a general reference to the actual device being used. CP/M is capable of controlling data transfer through many different devices that are included in these categories. Therefore, a more specific name must be assigned to each logical device category to inform CP/M of the kind of devices it must control. The specific name assigned to each logical device category is the "physical device name" Each physical device name helps CP/M to recognize and control a particular kind of peripheral hardware.

The physical device name currently assigned to each logical device category is listed on the right side of the menu.

Table 2-1 will help you determine which physical device names can be matched with which logical device names.

NOTE: By entering a ? at this menu, you will cause a screen display of helpful comments about this menu and its use.

This table names and describes the physical devices that can be assigned to each logical device category.

| SELECTION LETTER | LOGICAL DEVICE CATEGORY | PHYSICAL DEVICE NAME | DESCRIPTION AND/OR CATALOG NAME OF THE ACTUAL HARDWARE DEVICE USED |
|---|---|---|---|
| C | Console | TTY: | A printing terminal attached to serial port outlet A on the Z-100 (e.g. Decwriter). |
| | | CRT: | A video display terminal and keyboard. |
| | | BAT: | A batch pseudo-device using RDR: for input and LST: for output. |
| | | UC1: | A modem attached to serial port B on the Z-100. |
| R | Reader | TTY: | A printing terminal attached to serial port outlet A on the Z-100 (e.g. Decwriter). |
| | | PTR: | Not implemented. |
| | | UR1: | A modem attached to serial port B on the Z-100. |
| | | UR2: | A video display terminal and keyboard. |
| P | Punch | TTY: | A serial printer attached to serial port outlet A on the Z-100. |
| | | PTP: | Not implemented. |
| | | UP1: | A modem attached to serial port B on the Z-100. |
| | | UP2: | A video display terminal and keyboard. |
| L | List | TTY: | A serial printer attached to serial port outlet A on the Z-100 (e.g. H/Z-25, H-14, TI-810, Diablo, Epson MX-80 serial, Decwriter). |
| | | LPT: | A parallel printer attached to the parallel port on the Z-100 (e.g. Epson MX-80 parallel). |
| | | CRT: | A video display terminal and keyboard. |
| | | UL1: | A modem attached to serial port B on the Z-100. |

**Table 2-1**

### Specification Technique

To select a logical device category to be changed, you must first type the letter (**C, R, P,** or **L**) listed to the left of the category in the menu and ⓒ. CONFIGUR will display one of the following prompts:

```
    Console:  [T(ty), C(rt), B(at) or 1(UC1)]:
or
    Reader:   [T(ty),P(tr),1(UR1),2(UR2)] :
or
    Punch:    [T(ty),P(tp),1(UP1),2(UP2)] :
or
    List:     [ T(ty),C(rt),L(pt),1(UL1)] :
```

Each of these prompts show the logical device category on the left, and all of the physical device names that could be assigned to this category on the right.

To assign a physical device name to a logical device category, type the single character of the name that is displayed outside the parentheses.

CONFIGUR will respond by instantly displaying the name of the specified physical device in the "I/O Map Configuration" menu.

You can access the CONFIGUR MAIN MENU by entering **X** and ⓒ at the "Selection" prompt beneath the "I/O Map Configuration."

NOTE: These logical/physical device assignments can be temporarily changed outside of CONFIGUR by using the STAT utility (see Page 2-174).

### Specification Examples

For instance, if you want to change the assignment of physical device name "TTY" to "LPT" in the "List" category, type **L** (for "List") and ⓒ at the menu's selection prompt. Then type **L** (for "LPT") and ⓒ at the "List" category prompt. CONFIGUR will redisplay the "I/O Map Configuration" menu with the "LPT" named as "Currently" the physical device assigned to "List".

As a further example, if you wish to assign the physical device name "UR1" in place of the name "PTR" for the "Reader" category, type **R** (for "Reader") and ⓒ at the menu's selection prompt. Then type **1** (for "UR1") and ⓒ at the "Reader" category prompt. CONFIGUR will redisplay the "I/O Map Configuration" menu with the "UR1" named as "Currently" the physical device assigned to "Reader".

# 7. EXITING FROM CONFIGUR

The "MAIN MENU" enables you to exit from the CONFIGUR utility, and eventually gain access to the CP/M Operating System.

```
CP/M-85 System Configuration Utility version 2.2.100
      Copyright (C) 1982 by Zenith Data Systems

                  *** MAIN MENU ***


            P - Printer Configuration
            M - Modem   Configuration
            C - Command Configuration
            I - I/O map Configuration
            ? - Brief Help message

            X - Exit

Selection [P,M,C,I X or ?]:
```

## 7.1 The Exit Selection

To begin to exit from CONFIGUR to CP/M, press **X** and ⓒ.

If you have **not** specified any changes to the system during this CONFIGUR activity, CONFIGUR will immediately relinquish control to the CP/M, which will display the system prompt.

If you **have** specified any changes to the system during this CONFIGUR activity, CONFIGUR will display the following menu:

```
      *** EXIT OPTIONS ***


T - Make changes temporary (to memory only)
P - Make changes permanent (to memory and disk)
Q - Make no changes

? - Brief Help Message

Choice [T,P,Q or ?]:
```

- If you wish to cancel all of the changes that you have specified without having them applied to the system, then press **Q** or ⓒ. The unchanged CP/M Operating System will then display the system prompt.
- If you wish to have CONFIGUR apply all of the changes you have specified only to the CP/M system now in active computer memory, then press **T** (for Temporary) at this prompt. These changes will remain in effect until the computer is reset. If you had specified changes pertaining to printers and/or modems, CONFIGUR will now display a diagram to assist you in connecting your hardware devices.

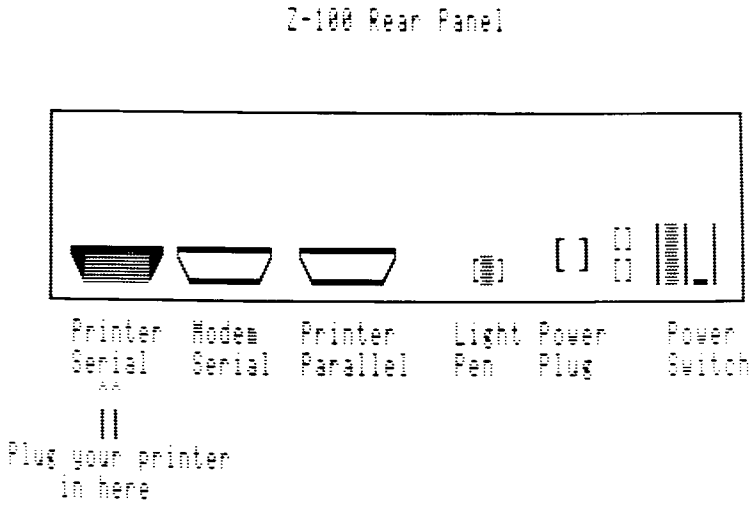
(See "7.2 Connecting Your Hardware".) However if you had specified no printer or modem changes, CONFIGUR will now relinquish control to CP/M, which will display the system prompt.

- If you wish to have CONFIGUR apply all of the changes you have specified both to the CP/M system in active computer memory and the system on the disk, then press **P** (for Permanent) at this prompt. These changes will go in effect as soon as CP/M regains control. These changes will be recorded on the disk in drive A until the next CONFIGUR session conducted with the same disk in drive A. If you specified changes pertaining to printers and/or modems, CONFIGUR will now display a diagram to assist you in connecting your hardware devices. (See "7.2 Connecting Your Hardware".) However if you specified no printer or modem changes, CONFIGUR will now relinquish control to CP/M, which will display a system prompt.

NOTE: If the disk in drive A was write-protected when you invoked CONFIGUR, the "P- Make changes permanent" choice will not appear in the exit menu, and entering the letter P at this menu will have no effect.

## 7.2 Connecting Your Hardware

Before returning control to the CP/M Operating System, CONFIGUR will display a diagram in the following form:



This diagram shows the location of the outlet on your Z-100 rear panel, onto which you should attach the RS232 cable that leads to your printer and/or modem. Attach the applicable cable(s) to the indicated outlet(s) before trying to use the printer and/or modem.

NOTE: This example diagram shows the message that will appear beneath the diagram if you had specified a serial printer during this CONFIGUR activity. The message appears beneath a different peripheral outlet if you had specified a parallel printer or a modem.

When your hardware devices are all properly attached, you can press ⓒⓡ to exit to the CP/M Operating System.

# 8. CONFIGUR ERROR MESSAGES

```
Incompatible Configur & CPM-85
```

**Explanation**

You are running a CONFIGUR utility under a CP/M Operating System of a different version. The version number of your CONFIGUR and your CP/M must match. Copy CP/M (using SYSGEN) and CONFIGUR (using PIP) from CP/M-85 Distribution Disk I, boot up, and invoke CONFIGUR again.

```
Bad choice.   Choice [1..9]:
```

**Explanation**

Your entry at the "Printer Selection" menu was not a number between 1 and 9. Enter the number between 1 and 9 that corresponds to your printer.

# DDT

*The Dynamic Debugging Utility*

> This text assumes that the user is familiar with assembly language programming, the hexadecimal number system, and 8080 CPU registers. Caution is advised during DDT use, because DDT has the ability to alter executable programs.

The Dynamic Debugging Tool (DDT) utility enables the user to debug machine language programs (files with the extension "HEX" or "COM"). DDT loads a program into the Transient Program Area of the computer's memory (1). During a debugging session, the utility exposes and manipulates the hexadecimal, assembly language, and ASCII forms of the loaded program when the user implements the special DDT commands (3). The user can copy the results of the debugging session to a disk file by exiting from DDT and implementing the SAVE resident command (2).

# 1. DDT INVOCATION

The DDT utility can be invoked using two methods: the System Prompt Method or the Utility Prompt Method.

## 1.1 System Prompt Invocation Method

You respond to the system prompt with a command line in the form:

A>**DDT {file name}**CR

Where {**file name**} is the complete name of a program file residing on the default disk with a "HEX" or "COM" extension. This file is automatically loaded into the computer's memory beginning at address 100H (the Transient Program Area start.)

DDT will identify itself with the message:

```
DDT VERS 2.2
NEXT   PC
aaaa  pppp
-
```

Where "2.2" is the utility's version number;

where "aaaa" is the next available memory address after the program is loaded (or the address after the last address occupied by the loaded program);

where "pppp" is the current position of the program counter within the Transient Program Area (this value is usually 0100 when the program is first loaded into memory); and

where "-" (the hyphen character) is the DDT prompt, at which the user can enter special DDT commands.

Because DDT loads programs into the Transient Program Area, usually beginning at address 0100, the user can approximate a program's size by subtracting 0100 from the "aaaa" value in the display.

## 1.2 Utility Prompt Invocation Method

You respond to the system prompt with the following command:

```
A>DDT<CR>
```

The DDT utility will identify itself with the following display:

```
DDT VERS 2.2
-
```

Where "2.2" is the version number; and

where "-" (the hyphen character) is the DDT prompt, at which you can enter special DDT commands.

To load a file from the default disk into memory under this invocation method, you must now make the following entries, in order: the letter **I,** (no space), the complete file name of the program to be loaded, a carriage return, the letter **R,** and another carriage return.

The DDT utility will **I**nsert the name of the program into memory, **R**ead the named program into memory, and display values for both the "NEXT" available address and the "PC" program counter. For example, if you desire to debug the program file named "PROGRAM.HEX", the console display of these transactions might appear as follows:

```
A>DDT<CR>
DDT VERS 2.2
-IPROGRAM.HEX<CR>
-R<CR>
NEXT  PC
3E80 0100
-
```

Where "3E80" is the next available memory address after the end of the program (or the address after the last address occupied by the loaded program); and

where "0100" is the current value of the program counter (or the first memory address occupied by the program); and

where "-" (the hyphen character) is the DDT prompt, at which you can enter special DDT commands.

## LOADING A PROGRAM FROM A NON-DEFAULT DRIVE

Using the second DDT invocation method, you can access a file from a non-default disk by doing the following, in order:

- Enter the letter **I,** the complete file name of the program to be loaded, and a carriage return; and
- Enter the character string **S5C** and a carriage return. This entry will produce a six-digit hexadecimal display.
- Enter the two-digit drive number for the drive containing the program that is to be debugged, and a carriage return. (See the table below to find the drive number that corresponds to the appropriate drive letter. This table will not be displayed on the console.)

| Drive Number | Drive Letter |
|---|---|
| **00** | DEFAULT |
| **01** | A |
| **02** | B |
| **03** | C |
| **04** | D |

NOTE: "DEFAULT" is the drive logged before DDT invocation.

The drive number you entered will be displayed on the right of the six-digit display, and a second six-digit display will appear beneath;

- Enter a . (period) and a carriage return. This entry will be displayed on the right of the second six-digit display; and finally
- Enter the letter **R** and a carriage return. DDT will read the program file into memory, and display values for both the "NEXT" available address and the "PC" program counter.

For instance, to access the file named "PROGRAM.HEX" from the disk within non-default drive B (02), you must interact with DDT as shown in the following display:

```
A>DDT<CR>
DDT VERS 2.2
-IPROGRAM.HEX<CR>
-S5C<CR>
005C 00 02<CR>
005D 57 .<CR>
-R<CR>
NEXT  PC
3880 0100
-
```

# 2. SAVING A DEBUGGED PROGRAM

You can preserve the results of a debugging session by copying a block of data from the Transient Program Area (TPA) to a disk file. The SAVE resident command will assist in this task. SAVE copies a user-specified number of pages (256-byte blocks) from the TPA to a user-specified file name on a disk.

To SAVE a program that has been loaded into memory and debugged by DDT, first exit from the DDT utility to the operating system. If the exit is performed properly, the image of the program in the TPA will remain undisturbed as SAVE copies from it to a disk file.

You can exit from the DDT utility by either of the following two methods:

- **Performing a Warm Boot** — Any time the DDT hyphen (-) prompt is displayed, enter a **CTRL-C** (by pressing the **C** key while holding down the **CTRL** key). The operating system will display a system prompt (A>).

- **Jumping to the Operating System's Entry Address** — Any time the DDT hyphen (-) prompt is displayed, you can trigger execution of the operating system, by entering the DDT command **G0** (where "**0**" is zero). This command sends the program counter to computer memory location 0000H (zero), where the system parameters area begins. The operating system responds by displaying a system prompt (A>).

You must next enter the SAVE command. The SAVE command is entered in the form:

A>**SAVE {pages} {filename.ext}**↵

Where {pages} are 256-byte units of data that are expressed in decimal (not hexadecimal) numerals, and

where **{filename.ext}** specifies the file name under which you wish to store the results of the debugging session.

You can SAVE the entire program by determining the (decimal) number of pages between the "pppp" value and the "aaaa" value displayed when DDT initially loads the program into the TPA, as shown:

```
NEXT  PC
aaaa pppp
```

Unless the program has been moved from the beginning of the TPA, the two left-hand digits in the "aaaa" value will be the hexadecimal number of pages the program occupies, as long as the two right-hand digits are not zeros. If the two right-hand digits are zeros, one should be subtracted from this total to determine the number of hexadecimal pages. In either case, the hexadecimal number of pages should be converted to a decimal number suitable for the SAVE command.

# 3. DDT COMMANDS

The DDT utility has its own assortment of commands. They are entered in response to the DDT hyphen (-) prompt. DDT command lines begin with a single command letter. These letters are listed on the left side of the following list:

- **A** Assembly: Assembly language mnemonics are inserted
- **D** Display: Display memory contents in hexadecimal and ASCII form
- **F** Fill: Fill a block of memory with a specified data constant
- **G** Go: Go to specified address to run a program
- **H** Hex: Hexadecimal computation of sums and differences
- **I** Insert: Insert file name into file control block
- **L** List: List assembler mnemonics of a program
- **M** Move: Move a data block to a different memory area
- **R** Read: Read a program file into memory from disk
- **S** Substitute: Substitute hexadecimal values at an address
- **T** Trace: Trace program execution
- **U** Untrace: Untrace program execution
- **X** eXamine: Examine or change registers or flags

Most of these command letters can be followed by parameters such as hexadecimal values or a file name. All DDT command lines must end with a carriage return.

When hexadecimal parameters are used, these values consist of one to four digits. (Longer numbers are automatically truncated on the right.) One, two, or three such values can be entered in some command lines. Values are separated by commas or single blank spaces.

Only one DDT command can be entered in response to a single DDT hyphen (-) prompt. Each DDT command line, however, can be composed using most of the same line editing keys and techniques as are used for commands entered at the CP/M system prompt.

No DDT command line can exceed 32 characters in length. If a thirty-third character is entered, it is interpreted as a carriage return and execution begins based on the first 32 characters in the command line.

Many DDT commands operate under a "CPU state" which corresponds to the program being tested. The CPU state holds the program's registers. Initially, all registers and flags contain zeroes — except for the program counter (P) and the stack pointer (S), which default to the value 100H.

The program counter is a CPU register that is used as a moveable reference point for DDT commands. It keeps track of the last hexadecimal address that was displayed and/or altered by a DDT command. The address immediately after this last address will be the starting address for the next DDT command the user enters (unless the next command specifies a different starting address).

## 3.1 A Assembly Language Mnemonics are Inserted

The A (assemble) command enables you to insert assembly language instructions into the program being tested. The command is entered in the form:

**As**

Where "s" is the memory address at which the user desires to start inserting assembly language instruction statements.

DDT responds to such an entry by echoing the value ("s") entered. You can then enter an assembly language statement to the right of the echoed "s" value. The statement must end with a carriage return.

DDT will display the next available memory address after the new statement is appended to the program. You can enter another statment to the right of the displayed address, or enter a carriage return alone to end A command operations and retrieve the DDT prompt.

For example, if you want to insert a "MOVE IMMEDIATE to register C" statement into a program at memory address 0104H, the following entry should be made in response to the DDT prompt:

**A104**CR

DDT will echo the address with the display:

```
0104
```

To the right of the address display, you can insert the statement:

**MVI C,{data}**CR

Where {data} is the data to be moved into register C.

DDT will then display the value for the next available memory location. Since the "MVI" statement took up two locations, DDT displays:

```
0106
```

You can enter another statement at location 0106H or end the operations of the A command by entering a carriage return alone.

NOTE: When the A command inserts a statement at a particular memory address, the statement(s) that formerly occupied that part of memory will be overwritten, and therefore destroyed. If you insert a statement that does not occupy the same number of locations as the statement(s) being replaced, the meaning of subsequent statements might be changed. You should use the "L" command immediately after finishing A command operations, to verify that the desired results were achieved during use of the "A" command. The following example demonstrates this problem.

Suppose you want to replace a "jump" instruction (JMP) with a "return from subroutine" statement (RET). The JMP statement occupies three locations, and the RET instruction will occupy one. Inserting the one-byte RET into the first location of the three-byte JMP will leave the last two-thirds of the JMP statement in the program. This partial statement could cause problems when the program is run.

## 3.2 D Display Memory Contents in Hexadecimal and ASCII Form

The D (display) command allows the user to view the contents of memory in both hexadecimal and ASCII formats. The display appears in the following form:

```
aaaa bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb cccccccccccccccc
```

Where "aaaa" is the address of the first memory location displayed in this line;

where "bb" represents the hexadecimal contents of a memory location; and

where "cccccccccccccccc" represents the ASCII translation of the contents of each memory location.

If the contents of a memory location cannot be displayed as an ASCII character, a period (.) will be displayed instead.

The display address acts as a pointer in memory which is initially set to 100H. As each memory location is displayed, the pointer is incremented by one so that at the end of a D command, the pointer is positioned ready to display the next 256 memory locations.

The three forms of the command are:

| | |
|---|---|
| **D** | Display memory from the current display address. DDT will display 12 lines, each representing 16 bytes of data. |
| **Ds** | Change the starting display address to "s". Then display memory beginning with address "s" and continuing for 192 more bytes. |
| **Ds,f** | Change the starting display address to "s". Then display memory beginning with address "s" and continuing until memory address "f" is reached. |
| **D,f** | Display memory from the current display address until address "f" is reached. |

Displays triggered by any of these commands can be suspended if a **CTRL-S** is entered during the display. The display will resume if any character is entered.

A display will be aborted if any character other than CTRL-S is entered. However, it is recommended that the display be intentionally aborted by pressing the DELETE key, because other characters will appear at the next DDT hyphen (-) prompt if they are used to abort the display.

## EXAMPLE DISPLAY

For example, if the user wants to see a hexadecimal and ASCII display of the file "SYSGEN.COM", **D** should be entered at the DDT prompt. This entry would produce a display like the following:

```
0100 C3 87 02 43 4F 50 59 52 49 47 48 54 20 28 43 29 ...COPYRIGHT (C)
0110 20 31 39 37 38 2C 20 44 49 47 49 54 41 4C 20 52  1978, DIGITAL R
0120 45 53 45 41 52 43 48 20 6F 26 00 29 29 29 29 29 ESEARCH o&.)))))
0130 29 29 C9 0E 01 CD 05 00 FE 61 D8 FE 7B D0 E6 5F ))........a..{..
0140 C9 5F 0E 02 CD 05 00 C9 3E 0D CD 41 01 3E 0A CD . ........>..A.>..
0150 41 01 C9 E5 CD 48 01 E1 7E B7 C8 E5 CD 41 01 E1 A....H..~....A..
0160 23 C3 58 01 D5 4F 2A 01 00 11 18 00 19 D1 E9 2A #.X..O*........*
0170 01 00 11 1B 00 19 E9 2A 01 00 11 1E 00 19 E9 2A ........*........*
0180 01 00 11 21 00 19 E9 2A 01 00 11 24 00 19 E9 2A ...!...*...$...*
0190 01 00 11 27 00 19 E5 21 9A 07 3A 98 07 BE F5 35 ...'...!..:.....5
01A0 CA AE 01 F1 0E 00 C2 B7 01 0E 02 C3 B7 01 F1 3A ...............:
01B0 98 07 32 9A 07 0E 01 E1 E9 0E 14 C3 05 00 0E 0F ..2............
```

NOTE: To use this example, you must first load SYSGEN.COM into memory.

## 3.3 F Fill A Block of Memory with a Specified Data Constant

The F (fill) command allows the user to fill a block of memory with a specific constant. The form of the command is:

Fs,f,c

Where "s" is the address at which the filling should begin;

where "f" is the address at which the filling should end; and

where "c" is the data constant that should occupy each memory address in between.

Any data that resided between addresses "s" and "f" prior to the entry of the command will be overwritten by the constant, and therefore destroyed.

Only hexadecimal values should be entered in such a command, and value "f" must be greater than value "s". If "s" is greater than "f", the operation will not be executed and the DDT prompt will reappear.

For example, the following command:

- **F9200,9400,E5** 

would fill every memory location from address 9200H through address 9400H with the hexadecimal value "E5".

## 3.4 G Go to Specified Address to Run a Program

The G (Go to) command enables you to begin execution of the program from any address, and to specify one or two execution breakpoints if desired. (A breakpoint is the address of an instruction which, when reached, will stop the execution of the program and redisplay the DDT prompt.) Execution begins with the instruction at the memory address immediately following the one specified in the command. The instruction at the specified address is not executed.

If no breakpoint is entered, the only other way in which the control of the program may be returned to the user is if an "RST 7" instruction is encountered within the program. This instruction will immediately stop program execution and redisplay the DDT prompt to allow further DDT commands from you.

The G command can be entered in the following forms:

**G** Begins execution of the program at the current value of the program counter, with no breakpoints set. The program will run to completion.

**Gs** Sets the program counter to address "s" and begins execution of the program from that address, with no breakpoints set. The program will run to completion.

**G,b** Begins execution of the program at the current value of the program counter and continues until the instruction at address "b" (the breakpoint) is reached. Then program execution stops.

**Gs,b** Sets the program counter to "s" and begins execution of the program at address "s". When the instruction at address "b" (the breakpoint) is reached, program execution stops.

**G,b,c** Begins execution of the program at the current value of the program counter and continues until either address "b" or address "c" is reached. When either of these breakpoint addresses is reached, program execution stops.

**Gs,b,c** Sets the program counter to address "s" and begins execution of the program at this point. When either address "b" or address "c" is reached, program execution stops.

At a breakpoint, program execution stops and DDT displays:

```
*bbbb
-
```

Where "bbbb" is the address at which program execution stopped; and

where "-" is the DDT prompt

For example, you could "go to" the very beginning of computer memory (address 0000H) and trigger execution of the program that is situated there. This program is, of course, the CP/M operating system. Its execution can be triggered by entry of a G and a zero, as shown:

```
-G0<CR>
```

The operating system would respond by displaying the system prompt, as shown:

```
A>
```

This command has the same effect as a warm boot.

## 3.5 H Hexadecimal Computation of Sums and Differences

The H (hexadecimal value) command simultaneously adds and subtracts two hexadecimal values. This command is entered in the following form:

**Ha,b**

Where "a" is a hexadecimal value; and

where "b" is another hexadecimal value.

The resulting display appears in the form:

```
ssss  dddd
```

Where "ssss" represents the sum of two values; and

where "dddd" represents the difference between the two values.

This command is helpful in determining addresses to which programs will be relocated with DDT command "M".

For example, if you have a program that begins at address 0311H, and wish to move this program 0126H bytes higher in memory, then the H command could be used to calculate the new starting address, as shown:

```
-H311,126<CR>
0437 01EB
```

0437H would be the new starting address for the program.

However, if you enter an "a" value that is smaller than the "b" value, the sum ("ssss") will be the same, but the difference ("dddd") will be equal to 10000H minus the amount that "b" is greater than "a".

Such a case is demonstrated by the following entry:

```
-H1,2<CR>
```

which will produce this displayed solution:

```
0003 FFFF
```

## 3.6 I Insert File Name Into File Control Block

The I (input) command allows you to insert a file name into the area of memory that is used to store the names of files to be read from the disk. This area of memory begins at address 5CH. This is one of the memory areas from which the Console Command Processor (a functioning part of the CP/M operating system) distributes control to utilities and resident commands. This DDT command is entered in one of the following forms:

-I{**primary file name**}<CR>

-I{**primary file name**}.{**extension**}<CR>

If the second form of the command is used and the {**extension**} entered is either "HEX" or "COM", then subsequent R commands can be used to read the pure binary or hexadecimal machine code.

The I command will not read the file from the disk and store it into memory. It will only insert the file name into the File Control Block portion of the Console Command Processor, so that a subsequent R command can read the named file into memory.

## 3.7 L List Assembly Language Mnemonics of a Program

The L (list) DDT command enables you to disassemble the instructions within a span of memory, and to display the assembly language mnemonics of the disassembled code on the console. The command can be entered in any of the following three forms:

**L** Lists 11 lines of disassembled machine code, beginning at the current list address. The list address acts as a pointer in memory which is initially set to 100H. As each memory location is disassembled, the pointer is incremented by one.

**Ls** Changes the list address to "s", and then lists 11 lines of disassembled machine code beginning at address "s".

**Ls,f** Lists disassembled code from starting address "s" to the final address "f".

The list appears in the form:

```
aaaa  mmmm  oooo
```

Where "aaaa" is the address of the instruction,

where "mmmm" is the mnemonic of the operator, and

where "oooo" is the operand.

Listings triggered by any of these commands can be suspended if a **CTRL-S** is entered during the listing. The listing will resume if any character is entered.

A listing will be aborted if any character other than CTRL-S is entered. It is recommended that the listing be intentionally aborted by pressing the DELETE key, because other characters will appear at the next DDT hyphen (-) prompt if they are used to abort the listing.

If an invalid mnemonic is encountered in a statement of a disassembled program, question marks ("??") will be used to represent it in the listing.

The disassembled mnemonics between address 0919 and address 091C of the program in memory can be listed with the entry of the following command:

**-L0919,091C**CR

Such a listing might appear as:

```
0919 OUT  F2
091A INX  H
091B MOV  A,M
091C ORA  A
```

## 3.8 M Move a Data Block

The M (move) command allows you to move a block of data from one area of memory to another. This command is entered as:

**Ms,f,d**

Where "s" is the starting address;

where "f" is the final address of the block of data to be moved; and

where "d" is the starting point of the memory area to which the data is moved.

The data is moved to the area of memory beginning at the address "d". An example of the command is:

**M100,200,1000**CR

This command would take the contents of the block of memory starting at address 0100H and running through address 0200H, and move these contents to the area of memory beginning with the address 1000H.

## 3.9 R Read a Program File Into Memory From Disk

The R (read) command is used after the I command to read COM and HEX files from the diskette into the transient program area in preparation for a debug operation. The R command requires a previous I command, specifying the name of the HEX or COM file to be read. The command can be entered in either of two forms:

**R** Reads the file whose name is in the file control block at address 5CH from disk and places it in the Transient Program Area. (The file name was placed in this location with the I command.)

**Rb** Reads the file whose name is in the file control block at address 5CH from disk and places it in the Transient Program Area with the addition of a bias factor, "b", which is a hexadecimal number added to each program instruction address or data address as it is read. This factor allows you to locate the program at any location in memory. When the bias is omitted, then b=0000 is assumed.

The read operation must not place the file in the first page of memory (0-0FFH) because this would write over the system parameters stored in this area. If the file specified in the preceding I command is a HEX file, the load address is derived from each individual HEX record. If the file to be loaded is a COM file, a load address of 100H is assumed. Any number of R commands may be issued following an I command to reread the program under test.

The Read command reads the file from the default drive. If another drive is desired, enter the command "S5C" to substitute the desired number, such that the number of the drive is specified by a number in the following table:

| Drive Number | Drive Letter |
|---|---|
| 00 | DEFAULT |
| 01 | A |
| 02 | B |
| 03 | C |
| 04 | D |

This substitution should be performed between the I command and the R command.

When the R command loads a named file into the Transient Program Area, a message in this form is displayed:

```
NEXT   PC
nnnn  pppp
```

Where "nnnn" is the address immediately following the loaded program; and

where "pppp" is the current value of the program counter (100H for COM files, or is taken from the last record if a HEX file is specified.)

The next address "nnnn" can be used to determine the size of the file which was loaded. If the beginning address is 100H, then subtracting 100H from "nnnn" will give the user the size of the program in bytes. The size derived in this manner is in hexadecimal units, and may have to be converted to decimal units before it is used.

## 3.10 S Substitute Hexadecimal Values

The S (set) command enables you to examine — and optionally alter — the contents of specified memory locations. This command is entered as:

**Sb**

Where "b" is the hexadecimal address of the first memory location to examine.

DDT responds with a display of addresses and bytes:

```
aaaa  cc
```

Where "aaaa" is the hexadecimal address, and

where "cc" is the hexadecimal contents of the memory location.

You may substitute a new value for "cc" by entering the new value (in one or two hexadecimal digits) and a carriage return when DDT displays "aaaa" and "cc". Your entry will appear on the right side of this display, and replace "cc" in the memory image of the program.

The next address "aaaa" and its contents "cc" are displayed, inviting you to substitute a new value for this "cc". When you are finished altering address contents in this sequence of addresses, a period (.) and a carriage return should be entered, rather than a new value. Your alterations will be retained in memory, and the DDT hyphen (-) prompt will reappear.

If you wish to skip an address without changing it, a carriage return (without a period) should be entered in response to one of the "aaaa" "cc" displays.

For example, if you enter the following command:

**-S100**CR

then memory addresses and their contents will be displayed on the screen, starting with address 0100H, as shown. Your substituted values for the address contents are in bold-faced print on the right side of the following example display:

```
0100 C3 3CCR
0101 C0 CCR
0102 01 10CR
0103 20 CR
0104 43 CR
0105 4F F4CR
0106 50 .CR
-
```

## 3.11 T Trace Program Execution

The T (trace) command allows the user to trace the execution of one to 65,535 (0FFFFH) program steps. During the trace, the contents of all registers and the status of all flags within the central processing unit (CPU) are displayed. This command can be entered in either of these forms:

**T** Displays the contents of the CPU registers and the status of the flags; then executes one program instruction. The DDT hyphen prompt (-) reappears.

**Tn** Displays the contents of the CPU registers and the status of the flags; then executes "n" program instructions, and stops. The DDT hyphen prompt (-) reappears.

Displays caused by the T command take this form:

```
CfZfMfEfIf  A=bb B=dddd D=dddd H=dddd S=dddd P=dddd inst *hhhh
```

Where "f" is a 0 or 1 flag value;

where "bb" is a byte value;

where "dddd" is a double byte quantity corresponding to a register pair;

where the "inst" field contains the dissassembled instruction which occurs at the location addressed by the program counter; and

where "hhhh" is the next address available for execution.

The display address (used in the D command) is set to the value of the H and L registers. The list address (used in the L command) is set to the value of "hhhh" so it will be ready to list the next program steps to be executed if desired. Since the state of the flags and registers of the CPU displayed by the T command occur before each instruction is executed, it may be helpful to use an X command to view the state of the CPU after the trace command.

The second form of the T command will trace the execution for n steps (n is a hexadecimal value) before a program breakpoint occurs. A breakpoint can be forced during long trace displays by using the DELETE key. The state of the CPU is displayed before each program step is executed in the trace mode.

If the program being tested must access the disk or Input/Output (I/O) devices through the CP/M system, the program tracing is discontinued at the interface to CP/M, and resumes after return from CP/M to the program being tested. CP/M functions which access I/O devices, such as the disk drive, operate at the proper speed (real-time), avoiding I/O timing problems. Programs running in the trace mode execute approximately 500 times slower than real-time because DDT takes control after each user instruction is executed. In programs which use interrupt instructions, the interrupts are always enabled during the trace mode.

## 3.12 U Untrace Program Execution

The U (untrace) command allows you to trace the execution of one to 65,535 (0FFFH) program steps. During the untrace, the contents of all registers and the status of all flags within the central processing unit (CPU) are displayed. Intermediate program steps are not displayed. This command can be entered in either of these forms:

**U** Displays the contents of the CPU registers and the status of the flags; then executes one program instruction. The DDT hyphen prompt (-) reappears.

**Un** Displays the contents of the CPU registers and the status of the flags; then executes "n" program instructions, and stops. The DDT hyphen prompt (-) reappears.

Displays caused by the U command take this form:

```
CfZfMfEfIf  A=bb B=dddd D=dddd H=dddd S=dddd P=dddd inst=*hhhh
```

Where "f" is a 0 or 1 flag value;

where "bb" is a byte value;

where "dddd" is a double byte quantity corresponding to a register pair;

where the "inst" field contains the dissassembled instruction which occurs at the location addressed by the program counter; and

where "hhhh" is the next address available for execution.

The display address (used in the D command) is set to the value of the H and L registers. The list address (used in the L command) is set to the value of "hhhh" so it will be ready to list the next program steps to be executed if desired. Since the state of the flags and registers of the CPU is displayed by the U command before each instruction is executed, it may be helpful to use an X command to view the state of the CPU after the untrace command.

The second form of the U command will untrace the execution for n steps (n is a hexadecimal value) before a program breakpoint occurs. A breakpoint can be forced during long untrace displays by using the DELETE key. The state of the CPU is displayed before each program step is executed in the untrace mode.

If the program being tested must access the disk or Input/Output (I/O) devices through the CP/M system, the program untracing discontinues at the interface to CP/M, and resumes after return from CP/M to the program being tested. CP/M functions which access I/O devices operate at the proper speed (real-time), avoiding I/O timing problems. Programs running in the untrace mode execute approximately 500 times slower than real-time because DDT takes control after each user instruction is executed. In programs which use interrupt instructions, the interrupts are always enabled during the untrace mode.

## 3.13 X eXamine or Change Registers or Flags

The X (examine) command enables you to display and alter the state of the registers and flags of the CPU at any time during the debugging process. This command can be entered in either of these forms:

**X**

**Xr**

Where "r" is one of the 8080 CPU registers in the following table:

| 8080 CPU REGISTER SYMBOL | REGISTER NAME | RANGE OF REGISTER CONTENTS |
|---|---|---|
| C | Carry Flag | (0/1) |
| Z | Zero Flag | (0/1) |
| M | Minus Flag | (0/1) |
| E | Even Parity Flag | (0/1) |
| I | Interdigit Carry | (0/1) |
| A | Accumulator | (0-FF) |
| B | BC Register Pair | (0-FFFF) |
| D | DE Register Pair | (0-FFFF) |
| H | HL Register Pair | (0-FFFF) |
| S | Stack Pointer | (0-FFFF) |
| P | Program Counter | (0-FFFF) |

The first form of the command displays the state of the CPU in this form:

```
CfZfMfEfIf  A=bb B=dddd D=dddd H=dddd S=dddd P=dddd inst
```

Where "f" is a 0 or 1 flag value,

where "bb" is a byte value,

where "dddd" is a double byte quantity corresponding to a register pair, and

where the "inst" field contains the dissassembled instruction which occurs at the location addressed by the program counter.

The second form of the command displays the flag or register value of the specified register, and allows alteration of the hexadecimal value within this flag or register. The user can substitute a new value for the value held in the register. You make this substitution by entering the new value and a carriage return to the right of the existing value.

This example demonstrates how the value in a register can be altered. Your entries are in bold faced type:

```
-XS
S=00FE EF00<CR>
-
```

When you substitute a value into a flag or register, ending the substitution with a carriage return, the DDT hyphen prompt (-) reappears. If you wish to make no changes to the values, a carriage return alone should be entered.

BC, DE, and HL are displayed as register pairs. You must enter the entire register pair when either B or C or the BC pair is altered.

# 4. DDT ERROR SIGNALS

DDT does not display entire messages when you make an erroneous entry. However, DDT will display a question mark (?) if your entry does not conform to valid entry syntax restrictions.

Additionally, DDT will display question marks (??) in place of invalid mnemonics that it encounters when dealing with the assembly language form of a program.

# DIR

*The Resident Command that Displays Disk File Directories*

The DIR resident command is issued to determine the presence of: (1) all of the files on a disk; (2) a specified file; or (3) a group of specified files. After command entry, DIR displays file names to the console (4). Some file names cannot be accessed by DIR (5).

## 1. DIRECTORY OF ALL FILES ON A DISK

DIR can be used to determine the names of all files on a disk by answering the system prompt with this entry:

A>**DIR**<CR>

If you desire a DIRectory of the files on a disk that does not reside in the default drive (drive B for instance) the DIR command should be entered with a drive specification, as:

A>**DIR B:**<CR>

NOTE: Because DIR is a resident command, it is automatically loaded into the computer with the rest of the operating system. It is never necessary (or valid) to specify a drive at the **beginning** of a DIR command line. For example, the command A>**B:DIR** <CR> is invalid.

## 2. DIRECTORY OF SPECIFIED FILE

To find out if one particular file resides on a disk, the complete name of that file is entered one space after the resident command specification "DIR". For example, the entry of the following command line will check the disk in default drive A for the file named "THISFILE.DOC":

A>**DIR THISFILE.DOC**CR

The presence of a specific file on a disk in a non-default drive can be determined by entering the appropriate drive name and a colon immediately before the name of the specified file.

## 3. DIRECTORY OF A GROUP OF FILES

To inquire about several files belonging to a group with similar names, you can enter an ambiguous file name (a file name with wildcard characters "*" and "?"). For example, to check the default drive disk for all of the files with the extension "BAK", you would enter this command line:

A>**DIR *.BAK**CR

As another example, the command line:

A>**DIR PROGRAM?.HEX**CR

will check the disk for files such as "PROGRAM1.HEX", "PROGRAM2.HEX", "PROGRAM3.HEX", etc. In addition, the command:

A>**DIR S*.COM**CR

will check the disk for files such as "SC.COM", "STAT.COM", and "SYSGEN.COM".

# 4. CONSOLE DISPLAY OF FILE DIRECTORY

Entering a DIR command line will produce a console display showing up to four file names in a horizontal line, with the name of the logged drive preceding each line. This example shows a DIRectory display of all of the files on some disk in drive A:

```
A: MVCPM207 COM : LIST     COM : PIP      COM : SUBMIT   COM
A: STAT     COM : XSUB     COM : ED       COM : ASM      COM
A: DDT      COM : LOAD     COM : CONFIGUR COM : SYSGEN   COM
A: DUMP     COM : DUMP     ASM : DUP      COM : FORMAT   COM
A: CPM48    COM : BSYSGEN  COM : DUMP     PRN : DUMP     HEX
```

If a specified file or group of files is not found on the disk being investigated, the console displays the message:

```
NO FILE
```

This message will also appear if a DIR command line references files on an empty disk.

# 5. FILES NOT ACCESSIBLE BY DIR

DIR commands will not produce a display indicating the presence of files that maintain the "SYS" status. (See "Changing a File's System Status" on Page 2-181".)

Files assigned to an unlogged user area are also inaccessible to a DIR command unless a USER command is issued before the DIR command. (See "USER" on Page 2-201.)

# DUMP

*The Utility that Displays a File in Hexadecimal Form*

The DUMP utility is invoked with the name of a file (1) to display the hexadecimal contents of each address in a file (2). The file contents appear in lines containing 16 bytes of data each. To the left of each line is the address of the first byte in each line. This display will continue to the end of the program, unless suspended or aborted (3).

## 1. DUMP INVOCATION

The hexadecimal contents of a file can be displayed on the console by responding to the system prompt with the command of:

A>**DUMP {filename.ext}**<CR>

Where {**filename.ext**} is the complete file name of the disk file that you wish to examine in a hexadecimal display.

## 2. EXAMPLE DUMP DISPLAY

The command **DUMP THISFILE.HEX** ⏎ might produce this display:

```
0000 C3 C0 01 20 43 4F 50 59 52 49 47 48 54 20 28 43
0010 29 20 31 39 37 39 2C 20 44 49 47 49 54 41 4C 20
0020 52 45 53 45 41 52 43 48 20 44 49 53 4B 20 4F 52
0030 20 44 49 52 45 43 54 4F 52 59 20 46 55 4C 4C 24
0040 46 49 4C 45 20 45 58 49 53 54 53 2C 20 45 52 41
0050 53 45 20 49 54 24 4E 45 57 20 46 49 4C 45 24 2A
0060 2A 20 46 49 4C 45 20 49 53 20 52 45 41 44 2F 4F
etc. . .
```

## 3. SUSPENDING LONG SCROLLING DISPLAYS

The displays produced by the DUMP utility often scroll by quickly on the console. They can be halted temporarily by entering **CTRL-S** at any time during the scroll. The display can be resumed by entering any keyboard character except CTRL-C (which executes a warm boot and aborts the program). A run of the DUMP utility can be aborted altogether by entering any keyboard character (other than CTRL-S) during the display.

## 4. DUMP ERROR MESSAGE

```
NO INPUT FILE PRESENT ON DISK
```

EXPLANATION: The file specified to be DUMPed does not exist on that disk. Command should be reentered by specifying the proper file name or drive name, or by inserting the proper disk.

# DUP

*The Utility that Duplicates and/or Verifies Entire Disks*

The DUP utility can be used to duplicate **all** of the data from one disk to another disk. It can also compare the two disks to verify whether the data recorded on one disk correspond exactly to the data recorded on another disk. If desired, DUP will even perform both operations consecutively, to ensure accurate duplication of a disk.

NOTE: Both of the disks involved in a DUP operation must be prepared in the exact same fashion by the FORMAT utility. The density and number of sides used for data storage on each disk must be identical. You can not duplicate disks that were initialized through the Z-DOS Operating System when using the CP/M DUP utility.

You can use the DUP utility through either of two methods: the Utility Prompt Method or the System Prompt Method.

## 1. UTILITY PROMPT METHOD

With this DUP method, you invoke the DUP utility from a disk by entering a command at the system prompt. Then you answer a series of DUP prompts to define the duplication operation.

## 1.1 Utility Prompt Command Entry

Invoke DUP by typing a command at the system prompt in the following form:

```
A>DUP ↵
```

The DUP utility will present this display:

```
Disk Utility Program
Version 2.2.100

Do you want to:

     A  copy and verify
     B  copy only
     C  verify only

     Z  exit to operating system

Selection:
```

This display includes a menu listing the four operations DUP offers. You can begin execution of an operation by typing the letter listed to the left of that operation. Each operation is explained in the following sections.

## 1.2 A copy and verify

The "copy and verify" operation makes an exact duplicate of a disk, and verifies that the operation was performed flawlessly, by comparing the two disks.

To begin this operation, type **A** at the "Selection:" prompt. (No carriage return is necessary.)

DUP will ask, in a series of consecutive prompts, for the letter of the drive that contains the disk to be copied **from** ("Source unit:") and for the letter of the drive that contains the disk to be copied **to** ("Destination unit:"). Enter the appropriate drive letter for each prompt. (No carriage return is necessary.)

When you have specified both a source and destination, DUP will instruct you to put the appropriate disks in the specified drives (even if you have already done so). Then the screen display will look something like:

```
Source unit:C
Destination unit:D

Put source disk in drive C.
Put destination disk in drive D.

Press RETURN to begin:
```

The last three lines in this display give you a final opportunity to make certain that you have specified the proper drives and inserted the proper disks. (It is important to make sure of these factors, to avoid accidentally duplicating the contents of a blank disk to a disk that has useful data.)

When both disks are seated in the proper drives, enter a carriage return to start the copying process or anything else to abort the operation. The lights on the specified drives will glow alternately to signify DUP activity. (The duration of the copy and verify operation varies depending on the density and number of sides used on the disks.) When finished, DUP will display:

```
Copy finished.
```

The verification process will begin automatically. DUP will compare the source and destination disks to verify the accuracy of the copy. Then DUP will display:

```
Verification finished.
```

and redisplay the DUP selection menu.

## 1.3 B copy only

The "copy only" operation makes an exact, duplicate of a disk.

To begin this operation, type **B** at the "Selection:" prompt. (No carriage return is necessary.)

DUP will ask, in a series of consecutive prompts, for the letter of the drive that contains the disk to be copied **from** ("Source unit:") and for the letter of the drive that contains the disk to be copied **to** ("Destination unit:"). The screen display for this operation may appear as:

```
Source unit:C
Destination unit:D

Put source disk in drive C.
Put destination disk in drive D.

Press RETURN to begin:
```

The last three lines in this display give you a final opportunity to make certain that you have specified the proper drives and inserted the proper disks. (It is important to make certain of these factors, to avoid accidentally duplicating the contents of a blank disk to a disk that has useful data.)

When both disks are seated in the proper drives, enter a carriage return to start the copying process or anything else to abort the operation. The lights on the specified drives will glow alternately to signify DUP activity. (The duration of the copy and verify operation varies depending on the density and number of sides used on the disks.) When finished, DUP will display:

```
Copy finished.
```

Then DUP will redisplay the selection menu.

## 1.4 C verify only

The "verify only" operation helps you determine whether two disks are identical in media and data contents.

To begin this operation, type **C** at the DUP selection menu. (No carriage return is necessary.)

DUP will ask, in two consecutive prompts, for the letter of the drives that contain the disks to be compared. (These prompts will ask you to specify "Source unit:" and "Destination unit:", although "source" and "destination" are not pertinent for this operation.)

Answer each prompt with the name of one of the drives containing a disk to be compared. The screen display for this operation might appear as:

```
Source unit:C
Destination unit:D

Put source disk in drive C.
Put destination disk in drive D.

Press RETURN to begin:
```

Enter a carriage return to start the verification process. The lights on the specified drives will glow alternately to signify DUP activity. (The duration of the verification operation varies depending on the density and number of sides used by the disks.)

If DUP finishes comparing the two disks and finds that they are absolutely identical, DUP will display:

```
Verification finished
```

Then DUP will redisplay the selection menu.

If the two disks do not correspond exactly, in data content and data position, then DUP displays:

```
Verification error
```

Then DUP will redisplay the selection menu.

NOTE: Conceivably, two disks could contain the exact same data, but in different positions on the disk surface. DUP verification regards such disks as **different**.

## 1.5 Z exit to operating system

When the DUP selection menu is redisplayed, you should insert a bootable disk in drive A. Then you can end the program and return to the operating system by typing the **Z** alternative at the "Selection:" prompt. Such an entry produces the system prompt:

```
A>
```

## 1.6 Invalid Entries During a DUP Prompt Operation

If you answer a DUP prompt with an invalid character, either the prompt or the menu will be repeated in most cases. If you specify a drive that does not exist in the hardware environment (when responding to the "source unit:" prompt or the "Destination unit:" prompt) the terminal may "hang up", freezing the keyboard. In such case, reset the computer and perform bootstrap to proceed with any CP/M operation.

# 2. SYSTEM PROMPT METHOD

The System Prompt Method enables you to include all of the specifications necessary for the DUP operation in a single command line entered at the CP/M system prompt.

## 2.1 Command Line Entry

System Prompt Method DUP commands are entered in this form:

A >**DUP {destination}:={source}:{[option]}** ⏎

Where **DUP** is the command line function, stored in the file DUP.COM on the logged disk;

where **{destination}** is the letter of the drive containing the blank disk that you wish to receive the copied data;

where **{source}** is the letter of the drive containing the data disk that you wish to duplicate; and

where **{[option]}** represents letters enclosed in square brackets [ ] and separated by a comma , to specify how the DUP operation should be conducted. One or two of the letters **C, V,** or **N** can be used, although none of these options are mandatory.

NOTE: In a CP/M command line "equation", the data source is always on the right and the data destination is always on the left.

## 2.2 DUP Options

Through the System Prompt Method, you can enter the following options to perform the DUP operation as indicated. These option letters should always be enclosed in square brackets. If two option letters are used, they should be separated by a comma.

**C** — Copy only: DUP will copy all of the data from the source disk to the destination disk, without comparing them to verify the accuracy of the copy.

**V** — Verify only: DUP will compare the source and destination disks to verify that they contain the exact same data in the exact same locations on the disk surface. When this option is specified, **either** disk can be the "source" or "destination".

**N** — No inquiry prompt: DUP will perform the operation you have specified without displaying a prompt to confirm whether the disks are in the appropriate drives.

**Without the C or V Options** — Copy and Verify: DUP will copy all of the data from the source disk to the destination disk, and then automatically compare the two disks to verify that the copying was accurate. This operation will be performed by default if you enter a DUP command with source drive and destination drive specifications but without specifying a C or V option.

## 2.3 DUP Defaults

If you enter a System Prompt Method DUP command line (specifying destination and source drives) and abstain from specifying certain options, the following default conditions will be in effect during the DUP operation:

- Copy and Verify operation will be performed. The entire contents of the source disk will be copied to the destination disk and then the two disks will be compared to verify whether they are exact duplicates. This operation occurs if neither the C option nor the V option is specified.
- Prompts are displayed to encourage you to insert the source and destination disks in the specified drives, and to trigger the start of the DUP operation with a carriage return. These prompts appear as:

```
Disk Utility Program
Version 2.2.100


Put source disk in drive A.
Put destination disk in drive B.

Press RETURN to begin:
```

## 2.4 System Prompt Examples

| | |
|---|---|
| A>**DUP B:=A:** ⏎ | DUP will prompt you to insert the proper disks into drives B and A, then copy all of the data from drive A to drive B, and then verify whether the copy was performed accurately. |
| A>**B:DUP C:=D:[C,N]** ⏎ | The DUP utility, in this case, is stored on the disk in non-default drive B. It will copy all of the data from the disk in drive D to the disk in drive C. As specified in the options, DUP will **not** prompt you to insert the disks in the appropriate drive and DUP will **not** verify the accuracy of the copy. |
| A>**DUP D:=C:[V,C]** ⏎ | If your command line options are contradictory (both V and C used in same line), DUP will acknowledge only the last one. In this case, DUP will prompt you to insert the proper disks into drives D and C, and then **copy** all of the data from drive C to drive D. DUP will **not** verify that the copy was performed accurately. |
| A>**dup c:=d:[v,n]** ⏎ | DUP will verify that the data stored on the disks in drives C and D is exactly the same and arranged in the same locations on the surfaces of these disks. As specified in the options, DUP will **not** copy any data and it will **not** prompt you to insert your disks. Notice that the letters in a DUP command line do not have to be in upper case (capitalized). |

NOTE: Conceivably, two disks could contain the exact same data, but in different positions on the disk surface. A DUP verification operation would regard such disks as **different**, and display a "Verification Error" message.

# 3. DUP ERROR MESSAGES

```
Media incompatible on diskettes.
```

EXPLANATION: Disks used for DUPping must be identical in size, density, number of sides, and tracks per inch.

```
Drives incompatible for copy operation.
```

EXPLANATION: DUP operations can only be performed between two drives that write the same type of disk media. Specify drives that write to identical disk types for your "Source unit" and "Destination unit".

```
Invalid Source Unit or Invalid Destination Unit.
```

EXPLANATION: Drives specified as source or destination must be drives that are connected in the hardware environment, turned on and recognized by the operating system. Specify such drives.

```
Hard read error on source disk. Copy/Verify aborted.
```

EXPLANATION: DUP failed in an attempt to read data from a source disk. Try the operation again. If DUP failures persist, use the PIP utility to copy files from the source disk, and the SYSGEN utility to copy the operating system from the source disk.

```
Hard read error on destination disk. Copy/Verify aborted.
```

EXPLANATION: DUP failed in an attempt to write data to the destination disk. Use the FORMAT utility to prepare the destination disk before using DUP. If DUP failures persist, use a different disk as the data destination. Make certain that the destination disk media is the same as the source disk media.

```
Verification error.
```

EXPLANATION: DUP's comparison of two disks found them to be different. Determine which disk contains the most desirable data. Call this disk the "source disk", use the FORMAT utility to erase and prepare the inferior disk. Perform DUP's "copy and verify" operation. If a second verification of these two disks produces this error, use a new disk of the same media type for the destination. Repeat the "copy and verify" operation.

```
Source and Destination cannot be the same drive.
```

EXPLANATION: Different drive units must be specified as source and destination unit.

```
Destination diskette is write protected.
```

EXPLANATION: Disk should be write-enabled by removing the adhesive tab from the write-enable notch on a 5.25-inch disk jacket, or by attaching an adhesive tab to the write-protect on an 8-inch disk.

```
Command line syntax error
```

EXPLANATION: System Prompt Method command line was entered incorrectly. Reenter using the entry form explained in "2.1 Command Line Entry".

```
Unknown command line option
```

EXPLANATION: System Prompt Method command line was entered with invalid characters used as options. Reenter using only the options listed in "2.2 DUP Options".

```
Error encountered during copy operation
```

EXPLANATION: DUP was unable to copy one disk's data to another disk. Retry the copy operation. If the error occurs again, use a different destination disk.

```
Operation aborted.
```

EXPLANATION: This message occurs when you press a character other than carriage return at the "Press RETURN to begin" prompt. Below it, the DUP menu will be redisplayed.

# ED

*The Line Editing Utility that Creates and Edits Text Files*

The ED utility allows you to compose, alter, and manipulate files containing ASCII characters. The files composed by ED are often referenced or manipulated by other system utilities and commands.

When invoking ED, you either create a new file or summon an old file into the computer (1). ED works on files using an area of computer memory known as the memory buffer (2). Files being EDited can be moved, altered, or displayed when you enter various ED commands (3).

# 1. ED INVOCATION

The ED utility is invoked by the entry of a command line in the following form:

A>**ED {file name}**<CR>

Where {**file name**} is the complete name of a file that you wish to compose or edit. You must specify the name of the text file here.

If the file resides in a non-default drive, this drive should be specified immediately before the file name in the command line. If the file does not yet exist, ED will create it on the disk in the default drive, or on a disk in a specified drive.

The following entry, for example, would cause ED to open the file named "THISFILE.TXT", that resides on the disk in non-default drive B:

A>**ED B:THISFILE.TXT**<CR>

When ED "opens" a file, it checks the disk directory for the name of the file. Then, ED reserves the computer's Transient Program Area (TPA) as a memory buffer to be used for text editing and file manipulation.

If the name of the file is not in the disk directory, ED creates the empty file and displays both the "NEW FILE" message and the ": *" prompt. You can begin inserting text into the empty file by using the "I" command at the prompt.

If the name of the file does exist in the disk directory, then ED locates it, and displays the ": *" prompt on the console. You can then move the existing file into the memory buffer by using the "A" command at the prompt.

# 2. ED STRUCTURE AND FEATURES

Text sections 2.1, 2.2, and 2.3 explain the structure and features of the ED utility. It is recommended that you fully understand these concepts before trying to implement ED commands.

## 2.1 Text Files in the CP/M Environment

To be properly read from, written to, and transferred, CP/M text files must be composed entirely of American Standard Code for Information Interchange (ASCII) characters. Text files must end with the entry of the CTRL-Z "end-of-file" character. Text can be moved to and from the disk in units of "lines". A line is defined as a string of ASCII characters that ends with the carriage return and line feed characters. (The "carriage return and line feed" combination can be entered by pressing the RETURN key.)

## 2.2 The Memory Buffer

The memory buffer is an area in the computer's memory that the ED utility uses as a "scratch pad" on which to compose and edit text before it is transferred to a disk for storage.

You can send a specified number of lines of text to the buffer from the disk or send text characters into the buffer by entering them directly through the keyboard. The memory buffer in your computer can hold more than 36,000 text characters at one time. When full, you can purge the buffer of text by writing a specified number of text lines from the full buffer to the disk.

## 2.3 The Character Pointer

Text in the buffer is usually arranged on numbered lines. To help you access locations within the text, the buffer contains an invisible character pointer, that can be moved to specific locations within the text by various user commands.

The character pointer resides either before the first character in the text, after the last character, or between any two text characters. When it is moved to a position in the text, a specified number of characters or lines can be inserted or deleted starting at the pointer's current position. Text is inserted through the use of special commands, sometimes followed by direct keyboard typing.

The character pointer is positioned by user commands that move it up or down to different lines, left or right along a line, or to the top or bottom of the file.The character pointer can also search through the text to locate a user-specified text string.

All ED commands are executed starting at the current position of the character pointer.

# 3. ED COMMANDS

Commands used within the CP/M EDitor are entered in response to the ":*" prompt, or to a prompt in the form "n:*" where "n" is the number of the text line upon which the character pointer resides. All ED commands (except those which end an ED session) can be entered in a series on the same command line. The last command entered on any line must be followed by a carriage return to initiate command processing.

The ED utility uses four commands (A, W, X, and R) to transfer text lines between the disk and the memory buffer (3.1).

When a file is in the memory buffer, four commands (B, C, L, and n) can be used to move the character pointer to a specified line or character (3.2).

Once the character pointer is in position, three commands (I, D, and K) are used to insert text into or delete text from the buffer (3.3)

Whenever text is within the memory buffer, two commands (T and P) can be used to display it to the screen. Three other commands (Z, V, and U) are used to alter characteristics of screen display (3.4).

When a file is in the memory buffer, four commands (F, S, N, and J) can be used to move the character pointer to an occurrence of a specified text string (3.5).

To remove an entire file from the memory buffer, four commands (E, Q, H, and O) can be used to send it from the buffer to the disk, or to dispose of it in some manner. These commands must be entered alone and followed immediately by a carriage return (3.6).

One command (M) is used to trigger multiple execution of other ED commands (3.7).

## 3.1 Moving Text to or from Memory Buffer

**nA** **Append lines from disk to buffer**

This command will copy "n" lines of text from the disk file specified in the invocation command to the memory buffer, where the text image can be edited.

The nA command must be implemented when you wish to edit text from an existing file. This command is entered in response to the ": *" prompt, and causes the "1: *" prompt to be displayed. The counterpart of the nA command, the nW command, transfers edited text lines back to the disk.

If you do not specify the number (n) of lines to be appended, ED will append one line from the disk. If the "#" character is entered in place of "n" (#A), then all of the text lines in the disk file (up to the capacity of the memory buffer) will be copied to the buffer.

**nW** **Write lines from buffer to disk**

This command transfers "n" lines of edited text from the memory buffer to the disk. Text that is written to the disk in this manner will no longer exist in the memory buffer.

The W command is implemented when the memory buffer becomes full during text editing. It is entered in response to the ": *" or "n: *" prompt, and causes the ": *" prompt to be displayed.

The W command always starts at the top of the buffer, transferring the first line in the buffer through the "nth" line in the buffer. The buffer text line that occurs after the "nth" line then becomes the first line, moving up to the top of the buffer.

If you do not specify the number of lines to be transferred to the disk, one line will be transferred. If the "#" character is entered in place of "n" (#W), then all of the text lines in the memory buffer will be transferred to the disk, and the buffer will be empty.

As the edited text is written back to the disk, a few lines at a time, it accumulates in a temporary file that has a "$$$" extension.

**nX** **eXtradite text block from buffer to temporary library file**

This command transfers a block of text from the memory buffer to a temporary disk file, so that it can be transferred back to the buffer (by the R command) at a desired location.

The block of text begins with the line containing the character pointer, and ends "n" lines later. The block is stored on the disk in a temporary file which is automatically named $$$.LIB (CP/M's standard name for a temporary library file). After this file is created on the disk, you should implement other ED commands to move the character pointer to the text location at which the temporary file should be inserted.

When the character pointer is at the desired location, the R command is used to transfer the $$$.LIB file back to the buffer at the desired location. The letter "R" and a carriage return should be entered to insert the text at the current location. The same text can then be inserted at another text location by moving the character pointer and, again, entering "R" and a carriage return.

Before a different block of text is transferred to the $$$.LIB file, the "0X" (a zero followed by an X) command should be entered to clear the old text from the file.

If no number is specified in the space preceding the "X", then one line of text (starting from the current character pointer location) is transferred to the temporary file. If the "#" symbol is specified, then all of the text lines within the buffer which follow the character pointer are transferred.

**Rf** **Read library file f from disk to buffer**

This command copies the text from a disk library file to the memory buffer, inserting this text at the current location of the character pointer.

The file being read into the buffer should usually be specified in place of the "f" in the command "Rf". However, these files are always assumed to have the "LIB" extension. Therefore, you only need to enter the primary file name in such a command. For example, the text from library file "ROUTINEX.LIB" could be read into the buffer with the entry of the following command:

**RROUTINEX**

If you wish to read a temporary library file into the buffer text, then no part of the file name need be entered with the R command. CP/M's standard temporary library file name "$$$.LIB" will be assumed.

## 3.2 Positioning the Character Pointer

**+/−B** **Beginning/Bottom of text character pointer movement**
This command will move the character pointer to the beginning of the first line of the text in the buffer (if entered in the form **B**), or to the end of the last line in the buffer (if entered in the form **−B**).

**+/−nL** **Line down/up character pointer movement**
This command will move the character pointer from its current line within the memory buffer text to another line a specified number (**n**) of lines away.

When the command is entered in the form **nL,** the character pointer will move ahead (down) the specified number of lines to the beginning of a text line.

When the command is entered in the form **−nL,** the character pointer will move backward (up) the specified number of lines to the beginning of a text line.

If the number of lines to move is not specified when the **L** command is entered, the pointer will move one line ahead (down).

In order to move the pointer to the beginning of the line upon which it currently resides, enter the "0L" command (with a zero preceeding the "L").

**+/−n** **Move to nth line and display it**
This command moves the character pointer a specified number of lines (**n**) and displays the text of the line on which it lands. This command produces the same results as the simultaneous entry of both the "L" and "T" commands. If no number is entered before the carriage return, ED assumes the number one.

**+/−nC Character pointer movement to right/left**

This command moves the character pointer a specified number (**n**) of character spaces, usually toward the right or left edge of a text line. (When a command-driven character pointer reaches the edge of a text line, the carriage return and line feed characters will cause it to change its direction momentarily.)

When entered with a plus sign ("+") or with no sign, this command will move the pointer the specified number of spaces to the right, and/or down to successive lines. When entered with a minus sign ("−"), this command will move the pointer the specified number of spaces to the left, and/or up to preceding lines. If you wish to move the character pointer past the edge of a text line using the "C" command, the number specified in the command will have to include two character spaces to get past the carriage return and line feed characters at the end of the line.

## 3.3 Inserting and Deleting Text

**I Insert characters from keyboard to buffer**

The "I" command enables you to insert characters directly into the text at the current position of the character pointer. This command is entered in response to the ": *" or "n: *" prompt.

If you enter a carriage return immediately after the "I" command, then the "n:" prompt appears on the next line and text characters can be inserted on successive lines until you enter a CTRL-Z "end-of-file" character. The entry of a CTRL-Z causes the ": *" prompt to be displayed. After this insertion operation, the character pointer will be positioned at the end of the last inserted text line.

If text is inserted on the same line as the "I" command, then the insertion operation will end when the next carriage return is entered, and a "*"command prompt will appear at the left edge of the screen. After this insertion operation, the character pointer will be positioned at the beginning of the line following the line of inserted text.

If upper and lower case insertion text is desired, enter the "I" command with a lower case "i". Entering the command with an upper case "I" will automatically translate all inserted text to upper case.

**+/−nD Delete characters from buffer text**

This command will delete a specified number (**n**) of characters from memory buffer text, starting at the location of the character pointer.

Deletions will take effect to the left of the pointer if the specified number of deletions is preceded by a "−" sign. If the specified number of deletions is preceded by a "+" sign or by nothing, characters to the right of the pointer will be deleted.

If no number of characters is specified for the deletion operation, one character will be deleted. If "#" is specified as the number of characters to be deleted, all text characters before or after the pointer (depending on the sign preceding the number) will be deleted.

The carriage return and line feed characters at the end of each text line count as two separate characters, even though they are produced by pressing only the RETURN key.

**+/−nK Kill lines from buffer text**

This command will delete a specified number (**n**) of lines from the memory buffer text, starting at the position of the character pointer.

If a "+" (plus sign) precedes the number of lines to be killed, the number of lines occurring **after** the character pointer will be killed. If a "−" (minus sign) precedes the number, the number of lines occurring **before** the pointer will be killed.

If the character pointer is positioned in the middle of a line during a "K" command, the portion of the text line to the left or right of the pointer will be deleted as if it were one entire line.

If no number (**n**) is specified for the deletion operation, then **one** line will be deleted. If "#" is specified as the number of lines to be deleted, then **all** text lines before or after the pointer (depending on the sign preceding the number) will be deleted.

## 3.4 Displaying Text to Console

**+/−nT** **Type text lines on console**

This command will cause a console display of a specified number (**n**) of text lines, starting at the position of the character pointer.

If the character pointer is in the middle of a line, the portion of the line between the pointer and the end of the line will be counted as an entire line. If this command begins with a minus sign ("−"), the specified number of lines before the line containing the character pointer and the line containing the pointer are displayed. If the pointer is positioned in the middle of a line and a zero is specified in the command, only the portion of the line from its beginning to the pointer will be displayed.

If no number (**n**) of lines is specified, **one** line will be displayed. If the "**#**" symbol is specified, **all** of the lines in one direction will be displayed.

You can interrupt a console display which scrolls too quickly by entering a CTRL-S character. The scroll will resume when another CTRL-S is entered. You can abort a long scrolling screen display by entering any other keyboard character while the display is in progress.

The "T" command will not affect the position of the character pointer. At the end of a "T" operation, the pointer marks the position at which the operation began. This position will be indicated by the number in the "n: *" prompt.

The "T" command can be entered in response to an "n: *" or ": *" prompt.

**nP** **Page display on console**

This command causes text to be displayed on the video screen in units of one page (23 lines), and deposits the character pointer at the end of the display.

If a display of the first page (23 lines) beginning at the character pointer is desired, a zero should precede the **P** in such a command. Hence, a **0P** command has the same effect as a **23T** command.

If the number one or no number is specified before the "P", then one page (23 lines) of text, starting 23 lines past the character pointer, will be displayed. The command **2P** will cause the display of two pages starting 23 lines past the pointer. The command **3P** will cause the display of three pages starting 23 lines past the character pointer, and so on. Hence, a **2P** command has the same effect as a **23L46T-23L.**

**nZ** **Zone interruption of text display scroll**

This command can be entered into a command line in front of the "T" or "P" display commands to interrupt a long console display scroll at time-regulated intervals so you can view the text one zone at a time.

When "T" and/or "P" commands are entered in a series within the same command line, "Z" commands can be placed in between to interrupt their execution for time periods determined by the number preceding the "Z".

The number (**n**) preceding the **Z** in the command stands for the number of half-seconds that the display scroll will be interrupted. If a command line contains a **10Z** between two display commands, the scrolling caused by the display commands will be interrupted for five seconds.

**+/−V** **aVert or replace line numbers in console displays**

If you prefer not to use the line numbers, the **−V** command will eliminate them from the console display.

The command **V** will restore line numbers to the console display.

A special form of this command, in which a zero precedes the V (**0V**), will produce a display showing how many locations remain unused in the memory buffer (r), and the total number of buffer locations that are accessible through the ED utility (t). The display appears in the form: "r/t".

**+/−U** **Upper case/lower case text translation**

If you prefer all characters entered into text through the ED utility to be upper case, enter the "U" command.

The "−U" command can be entered to allow the inserted text to be displayed in both lower and upper case.

## 3.5 Searching for Text Strings

**nFt** **Find text string t within buffer text**

This command is used to locate the specified number of occurrences (n) of a particular string of characters (t) within the text.

The string of characters (**t**) being sought is specified immediately after the **F** in the command, and ending with the entry of a **CTRL-Z** character and a carriage return.

If you do not specify the number of occurrences (n) of the string to be found within the text, only the first occurrence is found.

If you desire to locate a string of characters (t) that contains the carriage return and line feed characters, these two characters can be specified in the command line with the **CTRL-L** character.

The string specified in the command (t) must match the actual text in spelling, spacing, capitalization, etc.

The specification of a string of characters that is not found in the text produces the error message:

```
BREAK "#" AT STRING
```

Where "string" is an unfound text string, and where the character pointer goes back to its position before the search operation failed.

**nSdt** **Search and replace text string**

This command performs the operations of the "F", "D", and "I" commands simultaneously, finding a specified string (d) within the buffer text, deleting it, and inserting a second specified string (t) at the same location.

The number (n) in the command represents the number of text string substitutions desired by the user throughout the text. The omission of this number will cause a substitution to be made only at the first occurrence of the sought-after string (d). The entry of the "#" symbol in place of this number will cause the substitution to be made at every occurrence of the specified string throughout the text.

The string of characters to be found and deleted in the buffer text (d) is specified immediately after the "S" in the command, and ended by a CTRL-Z character. Immediately after this CTRL-Z, the text string to be inserted (t) is entered, with a second CTRL-Z and a carriage return.

The specification of a string of characters that is not found in the text produces the error message:

```
BREAK "#" AT STRING
```

Where "string" is an unfound text string, and where the character pointer goes back to its position before the search and replace operation failed.

**nNt** **fiNd text string on disk**

This command performs the same search operation as the "F" command except that it can search an entire file for a text string (t).

If the specified string (t) is not found in the memory buffer, this command will automatically write the contents of the buffer to the disk (into a temporary file, as the "W" command does) and append an image of another portion of the disk file's text to the buffer (as the "A" command does) until the entire file has been searched for the string the specified number of times (n).

**nJftd** **Juxtaposition substitution and deletion**

This command finds a first string (f), inserts a second string (t) after the first, and then deletes all of the text between the end of the inserted string (t) and the beginning of the third string (d).

The "J" in the command is immediately followed by the first text string (f), a CTRL-Z, the text string to be inserted (t), a CTRL-Z, the third text string (d), a CTRL-Z, and a carriage return.

The third string (d) serves as the restraining boundary for the text deletion.

This multi-faceted operation is performed a specified number of times (n), or once, if no number is specified. If the "#" is specified, the operation will be performed for all occurrences of the first text string (f).

If the third command line string cannot be found in the buffer text, no text is deleted.

## 3.6 Closing a Text File

**E** **End session while buffer text becomes permanent disk file**
All text in memory buffer is copied to the disk, where it is combined with any text that has accumulated in a temporary file, and assigned the original file name.

At the same time, the version of the file that was copied to perform the edit is assigned the extension ".BAK" in place of its old extension.

The operating system regains control and displays the system prompt.

**Q** **Quit session by deleting edited copy of file**
All text in the memory buffer and/or any temporary file created during the session is deleted. Any existing versions of the file on the disk maintain the status and names they held prior to the editing session.

If the file being edited existed on disk before the session, then the original version remains intact, as if the editing session never took place.

Since the accidental use of this command may delete important text composed or edited during the session, its entry will produce the "Q-(Y/N)?" confirmation message. The **Y** character must be entered before the deletion will be executed. If the **N** is entered, the current editing session will continue.

**H** **Halt session temporarily to save alterations**
All alterations made to text (or any text composed) will be saved under the active file name, and the editing session will continue with an image of the currently edited file automatically appended to the memory buffer. This command has the same effect as entering a combination of both the "E" and "A" commands.

**O** **Omit recent alterations and re-start edit session**

Any text in the memory buffer or in a temporary disk file is deleted, and the editing session continues, using the same text with which it began.

In effect, this command nullifies any text alterations or composition performed in the most recent ED session and starts the session over, as if the "Q" and "A" commands had been entered consecutively.

## 3.7 Causing Multiple Command Execution

**nM** **Multiple command execution**

This command permits you to execute one or more commands a specified number of times (n) without additional command entries. Commands are entered on the same line, following the **nM** command in a string terminated by either a carriage return or a CTRL-Z.

All commands following the **nM** will be executed the number of times specified at the beginning of the entire command line (n), or until an error condition is encountered. If no number (n) is specified, then the operations invoked by the command line will be implemented from the position of the character pointer through the end of the text, or until an error condition is encountered.

This command is commonly used with the search and replace command ("nS"), to facilitate text string substitution throughout a large text area. When such a search reaches the end of the text in the memory buffer, an error condition indicates that the substitution can no longer be executed.

Multiple commands are executed from the position of the character pointer toward the end of the text. The pointer should be positioned at the beginning of the buffer text if multiple commands are to be executed throughout the text.

# 4. THE FILE EDITING CYCLE

The following sequence of diagrams shows the file named "REPORT.DOC" as it undergoes ED's file editing cycle. The left side of the diagrams represents the memory buffer, and the right side of the diagrams represents the logged disk.

In Figure 2-1, the user has opened the new file "REPORT.DOC" by entering **ED REPORT.DOC** at the "A>" system prompt. You then enter ED's **I** command, and begin typing text into the memory buffer file. No text has yet been recorded on the disk.

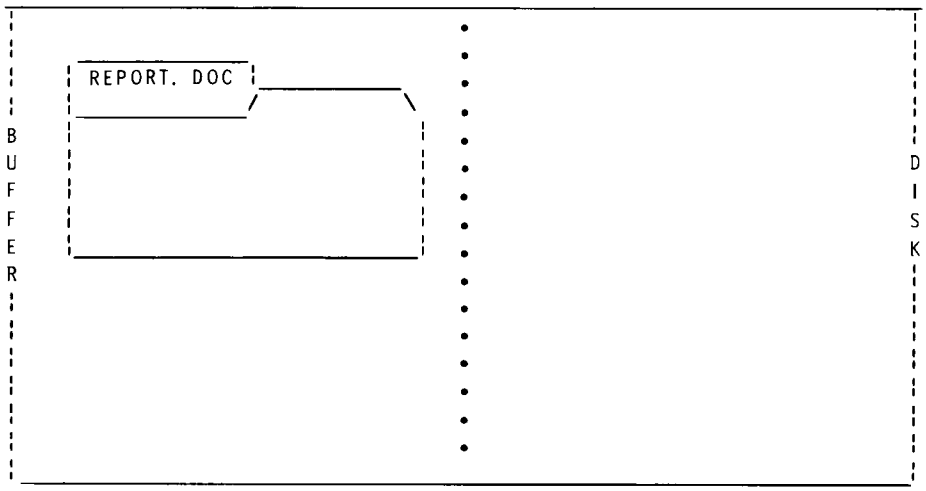



Figure 2-1

In Figure 2-2, you inserted text into the file, and want to save some number of text lines. First end the insert by entering a CTRL-Z. Then to save the text, enter ED's **W** command to send a specified number of text lines from the buffer to a temporary file on the disk. ED gives this temporary file the name "REPORT.$$$". The buffer file remains.

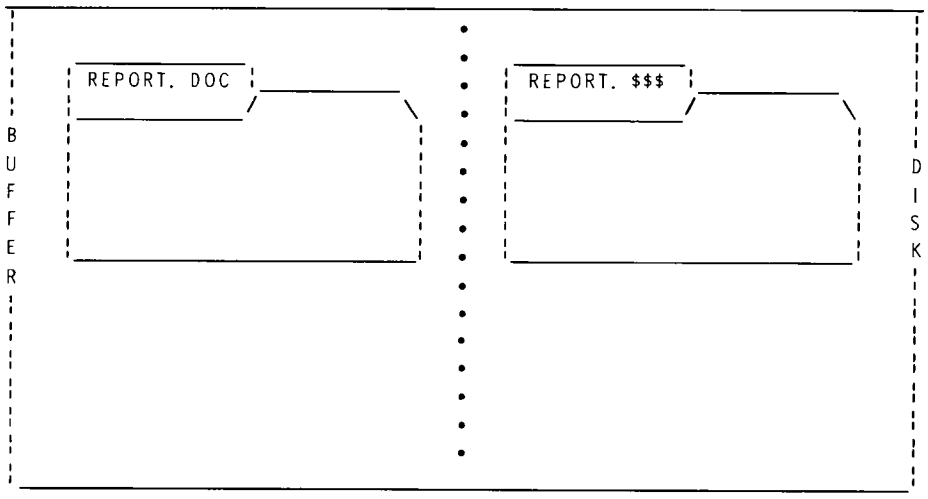



Figure 2-2

In Figure 2-3 you end the editing session and save all of the text composed for "REPORT.DOC", by entering ED's **E** command. This saved text is recorded on the disk under the file name "REPORT.DOC" Both the buffer file "REPORT.DOC" and the temporary disk file "REPORT.$$$" have disappeared.

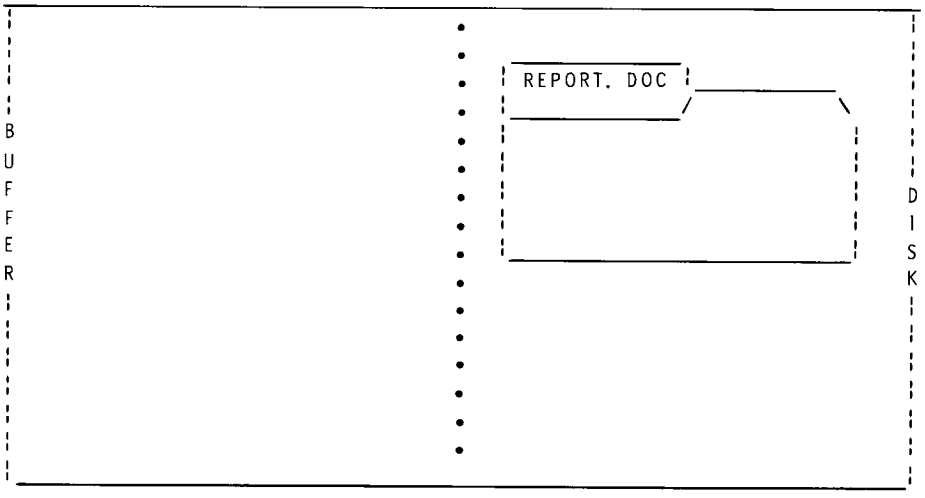



Figure 2-3

In Figure 2-4, you are reopening the file "REPORT.DOC" to edit it. First enter **ED REPORT.DOC** to invoke ED. Then enter ED's **A** command to bring a copy of a specified number of text lines from the disk into the buffer. The disk copy of "REPORT.DOC" remains on the disk.

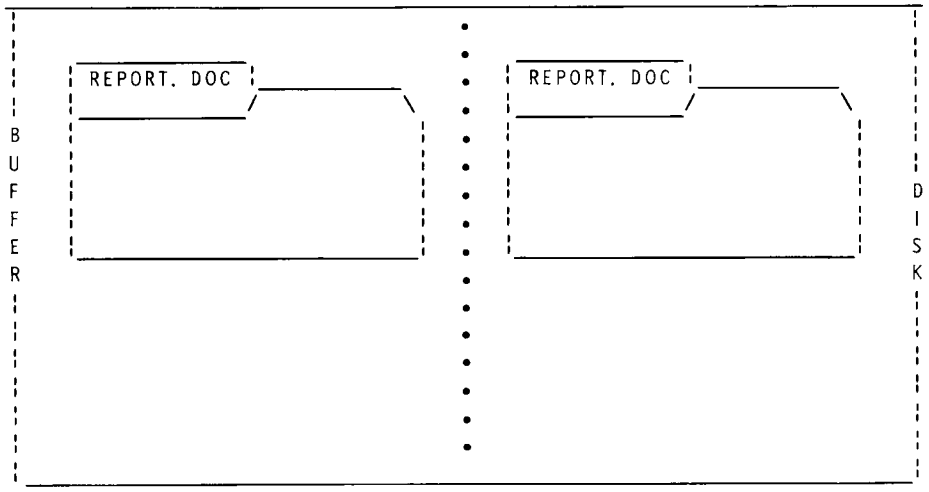



Figure 2-4

You changed and/or added to the text lines that were brought into the buffer. To save these changes and/or additons, again enter a CTRL-Z and ED's **W** command. Figure 2-5 reflects the state of the files after these activities. The original disk copy of "REPORT.DOC" remains on the disk, a copy of some of the files text remains in the buffer under the name "REPORT.DOC", and some number of edited text lines from the file have been sent to the temporary disk file named "REPORT.$$$".

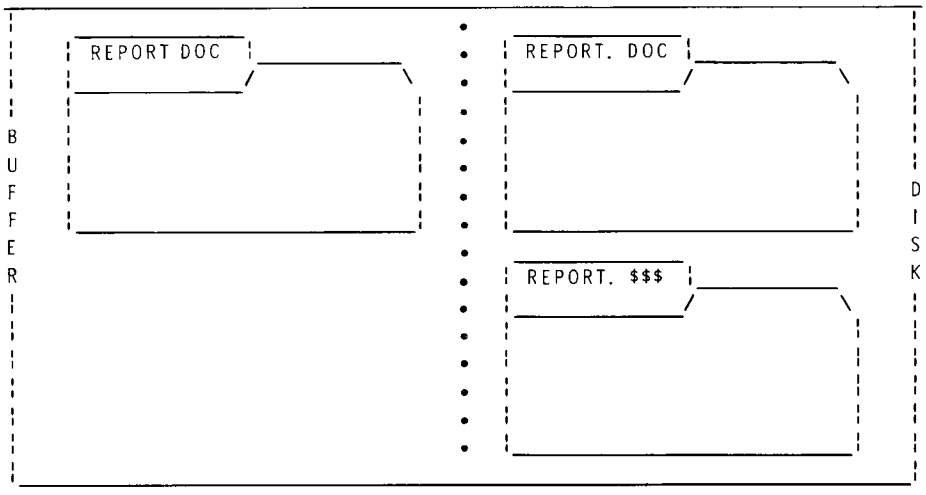



Figure 2-5

In Figure 2-6 you closed the file editing session and saved the changes and/or additions made to the text, by entering ED's **E** command. ED reacts to this entry by combining all text from buffer file "REPORT.DOC", all text from temporary disk file "REPORT.$$$", and any unchanged text from the original disk file "REPORT.DOC" into a new disk file named "REPORT.DOC". Then ED renames the original "REPORT.DOC" disk file to "REPORT.BAK". The buffer file and the temporary disk file disappear.

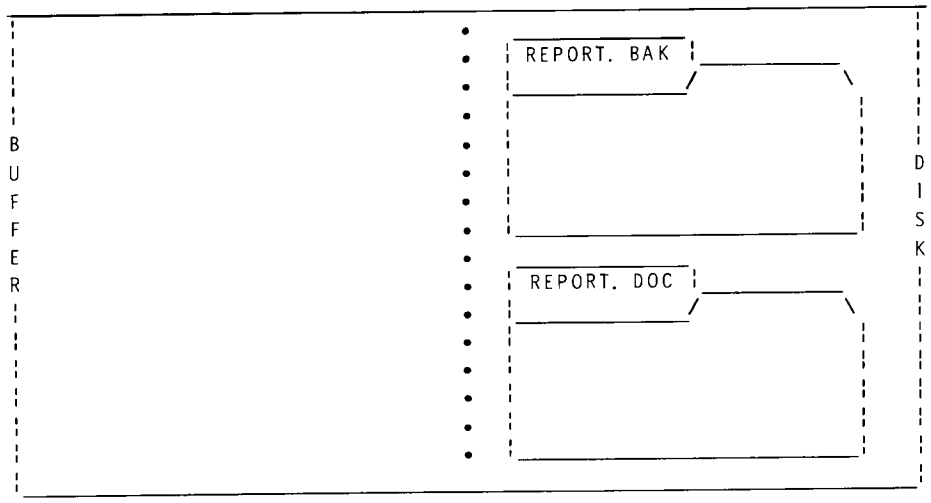



Figure 2-6

Thus a file with a file name extension assigned by the user is usually the latest version of the file, whereas a file with the extension "BAK" is usually an old copy of the file.

If you reedit the disk file "REPORT.DOC" and save the newly-edited version, the current disk file "REPORT.DOC" will become the new disk file "REPORT.BAK", as the memory file "REPORT.DOC" and the disk file "REPORT.$$$" combine to become the new disk file "REPORT.DOC". The old disk file "REPORT.BAK" will be automatically deleted after the third ED session with the file.

NOTE: These diagrams only illustrate a few of the basic options for file manipulation. The text on ED commands explains other options.

# 5. ED ERROR MESSAGES

```
BREAK "?" AT x
     : *
```

EXPLANATION: ("x" is an invalid character that the user entered.) You entered an ED command under inappropriate circumstances, at the wrong kind of prompt, or with improper syntax. Command should be reentered.

```
DISK OR DIRECTORY FULL
```

EXPLANATION: You either entered **ED** without a file name argument at the CP/M system prompt, or composed more text than the disk could hold. For the former, you should invoke the ED utility by typing a command line with a file name argument. For the latter, you cannot save the overflow of text. When composing future documents, use ED's "W" command or CP/M's STAT utility more often.

```
** FILE IS READ/ONLY **
```

EXPLANATION: You tried to save newly-edited text to a file that cannot be written to because it has read/only status. You should abandon the text in the buffer with ED's "Q" exit command, and use CP/M's STAT command to assign read/write status to the file before the next edit.

```
"SYSTEM" FILE NOT ACCESSIBLE
```

EXPLANATION: You tried to edit a file that had been given the "system" status by the STAT command. This status hides a file from commands such as ED and DIR. Assign the "directory" status to the file, using the STAT command, before trying to edit the file again.

```
NO MEMORY
```

EXPLANATION: You filled the memory buffer, and must use the "W" command to send some of the buffer text to the disk.

# ERA

*The Resident Command for Erasing Files*

The ERA resident command frees the space once occupied by disk files, permitting the storage of new files. ERA will erase a single file (1) or a group of files (2). ERA also displays a message when it cannot erase a file (3). Caution should be exercised when using ERA because erased files cannot be rescued.

## 1. ERASING A DISK FILE

To erase a file from a disk, enter a command line in the following form:

A>**ERA {filename.ext}**↵

Where {**filename.ext**} is the complete name of the file you wish to delete.

Files that do not reside on a disk in the default drive can only be erased when their file name is preceded in the command line by the appropriate drive specification.

# 2. ERASING GROUPS OF FILES

A group of files with similar names can be deleted by a single ERA command line when ambiguous file names (names with the "*" or "?" characters) are used, as:

```
A>ERA B:PROGRAM?.PRN ↵
```

which would delete files with names such as PROGRAM1.PRN, PROGRAM2.PRN, PROGRAMX.PRN, and PROGRAM/.PRN from the disk in drive B.

This command example shows how to delete even more files at once:

```
A>ERA B:*.* ↵
```

This entry would erase every file from a disk. Because of the destructive potential of this form of the command, ERASE will ask you for confirmation with this message:

```
ALL (Y/N)?
```

The command will not be executed until you confirm it by pressing the **Y** key. If the **N** key is pressed, no files will be erased and the system prompt will be displayed.

# 3. CONSOLE RESPONSE TO ERA COMMANDS

When ERA finds the specified file and erases it, the system prompt returns.

If the specified file does not reside on a disk in the specified drive, then the console will display:

```
NO FILE
```

If the file you desire to erase is write-protected by the "R/O" (read only) status, or if you switched disks between drives without performing a warm boot, this message will be displayed:

```
Bdos Err On X: File R/O
```

or

```
Bdos Err On X: R/O
```

Where "X" is the letter of the drive containing the write-protected file.

If the disk containing the file to be erased is mechanically write-protected (with adhesive tabs for 5.25-inch, or without adhesive tabs for 8-inch), then this message will be displayed:

```
Bdos Err on x: Bad Sector
```

Where "x" is the drive from which you tried to erase a file.

If you specify a 5.25-inch drive in an ERA command, and that drive contains no disk, ERA will search the drive for the disk indefinitely. Reset the computer and reboot the system to end this search.

Where "x" is the drive from which the user tried to erase a file.

# FORMAT

*The Utility that Prepares the Disk Surface*

FORMAT prepares a floppy disk for storing data by establishing storage areas on the disk surface. At the same time, FORMAT erases any data that remains on the disk from prior use, and sometimes inspects the recording surface for imperfections that could impair data storage or transmisison. FORMAT also enables you to determine how much data you will be able to store on the disk.

CAUTION: Because FORMAT erases all existing data on a disk, make certain that you only format blank disks or disks containing expendable data. You can use the DIR (Page 2-83) or STAT (Page 2-174) commands to check a disk for valuable data files before FORMATting it. However, the DIR and STAT cannot always find all of the files on a disk.

You can use the FORMAT utility through either of two methods: the Utility Prompt Method or the System Prompt Method.

## 1. UTILITY PROMPT METHOD

With this FORMAT method, you load the FORMAT utility into memory by entering a command at the system prompt. Then you answer a series of FORMAT prompts to define the formatting operation.

### 1.1 Utility Prompt Command Entry

Answer the system prompt with a command in the following form:

A>**FORMAT** ⏎

When invoked through the FORMAT prompt method, FORMAT first identifies itself with name, version number, and a caution about its capabilities. It also asks you if you wish to continue the operation, as shown:

```
CP/M-85 Format Version 2.100

This program is used to initialize a disk.
All information currently on the disk will be destroyed.
Is that what you want? (y/n):
```

Respond to this question by entering **Y**.

## 1.2. Specifying the Disk to be Formatted

After you have confirmed your intention to format a disk, FORMAT asks:

```
Which drive do you wish to use for this operation?
```

Answer this prompt by entering the letter of the drive containing the disk you wish to format. The drive you specify must be a valid disk drive in your hardware environment containing a write-enabled floppy disk of the proper disk size and sector type.

NOTE: This drive does not necessarily have to be a physical drive. For instance if you have only one physical 5.25-inch drive slot, then you can specify drive B at this prompt. You will later be prompted to put the appropriate disk in the drive.

## 1.3 Defining the FORMAT Operation

After you specify the drive containing the disk you want to format, FORMAT will enable you to define the manner in which you want the disk to be formatted. If you are formatting a 5.25-inch disk, the FORMAT utility enables you to specify the **number of sides** you want prepared for data storage. If you are formatting an 8-inch disk, the FORMAT utility enables you to specify the **density level** at which data will be stored on the disk.

## SPECIFYING NUMBER OF SIDES FOR A 5.25-INCH DISK

If you are formatting a 5.25-inch disk, the FORMAT utility enables you to specify the number of sides you want formatted by displaying the following prompt:

```
Number of Sides? (1=single, 2=double):
```

Entering the number **1** at this prompt will give the formatted disk a file capacity of 148 kilobytes. Entering the number **2** will give the formatted disk a file capacity of 304 kilobytes.

NOTE: All 5.25-inch disks formatted with this CP/M release are automatically formatted at double-density. Therefore, the FORMAT utility will not prompt you to specify the density (level of data concentration) at which you want a 5.25-inch disk formatted.

## SPECIFYING DATA DENSITY FOR AN 8-INCH DISK

If you are formatting an 8-inch disk, the FORMAT utility enables you to specify the level of density at which you wish to store data by displaying the following prompt:

```
Which density? (S=single, D=double):
```

The "density" of a disk refers to the level of concentration of the data stored on its surface. Higher density levels sometimes decrease data access reliability.

- If the disk in the specified drive is single-sided, then entering the letter **S** (for single-density) at this prompt will give the formatted disk a file capacity of 241 kilobytes. Entering the letter **D** (for double-density) will give the formatted disk a file capacity of 482 kilobytes.
- If the disk in the specified drive is double-sided, then entering the letter **S** (for single-density) at this prompt will give the formatted disk a file capacity of 490 kilobytes. Entering the letter **D** (for double-density) will give the formatted disk a file capacity of 980 kilobytes.

NOTE: Some 8-inch disks allow you to format only one side and others allow you to format both sides. These two types of disk are distinguished by the position of a small hole in the disk near the center spindle hole. Therefore, the number of sides that can be formatted on a particular 8-inch disk is an unchangeable feature of that particular disk. the FORMAT utility detects the position of this hole and automatically prepares to format the disk on the appropriate number of sides.

## 1.4 Beginning the FORMAT Operation

After you have defined the manner in which your disk will be formatted, FORMAT will display the following prompt to enable you to begin or cancel the oepration:

```
Put the disk you wish to be formatted in drive x.
Press RETURN to begin, anything else to abort.
```

(The character "x" stands for the letter of the disk drive you specified.) Entering a carriage return at this prompt will begin the actual formatting operation, while entering any other keyboard character will cause all FORMAT activities to end as CP/M displays the system prompt.

NOTE: It takes at least a minute for FORMAT to format most disks. During this time the light on the specified disk drive will glow.

## 1.5 Ending a FORMAT Operation

When FORMAT finishes preparing a disk's surface, it will display the following prompt:

```
Do you have more disks to format? (y/n):
```

If you wish to FORMAT another disk or partition without reinvoking FORMAT, press **Y** at this prompt. FORMAT will again prompt to specify the drive containing the disk you wish to format.

If you have no other disks to format, press **N**. CP/M will display the A> system prompt, at which you can enter any CP/M command.

# 2. SYSTEM PROMPT METHOD

The System Prompt Method enables you to include all of the specifications necessary for the FORMAT operation in a single command line. Enter this command line at the CP/M system prompt.

## 2.1 Command Line Entry

System Prompt Method FORMAT commands are entered in the following form:

A>**FORMAT {drive}:{[option,option]}**CR

Where **FORMAT** is the command line function, stored in the file FORMAT.COM on the logged disk;

where {**drive**} is the letter of the drive that contains the disk you wish to format (this letter must represent a valid drive in your hardware environment, such as **A**, **B**, **C**, or **D**); and

where {**[option,option]**} represents letters and/or numbers enclosed in square brackets [ ] and separated by commas , to specify how the formatting operation should be conducted.

## 2.2 FORMAT Options

FORMAT command lines entered by the System Prompt Method can include the following options:

**SD** 8-inch disk formatted to Single Density;

**DD** 8-inch disk formatted to Double Density;

**1S** 5.25-inch disk formatted on only one side;

**2S** 5.25-inch disk formatted on both sides;

**F** Fast formatting, because the routine test of disk surface media is not performed;

**N** No prompt displayed between FORMAT command entry and FORMAT execution;

Options should be enclosed in square brackets, and separated by commas (when more than one is used). Options are the last item in a FORMAT command line before the carriage return.

NOTE: All 5.25-inch disks are automatically formatted at double density by this version of FORMAT. 8-inch disks are automatically formatted on the number of sides for which the disk has been certified by the manufacturer. (FORMAT detects this certification by checking the position of the small hole in the disk cover next to the center spindle hole.)

## 2.3 System Prompt Method Defaults

When you enter a FORMAT command line with a drive specification, and decline to specify some or all of the possible options, FORMAT will prepare the disk according to the following default criteria:

- 5.25-inch, disk formatted to Double Density (regardless of any options you might specify);
- 8-inch disk formatted to Double Density (as if you specified the DD option);
- 5.25-inch disk formatted on both sides (as if you specified the 2S option);
- 8-inch, single-sided disk formatted on one side (regardless of any options you might specify);
- 8-inch, double-sided disk formatted on both sides (regardless of any options you might specify);
- Disk surface will be tested for data retention (as if you did not specify the F option); and
- Prompts will be displayed between FORMAT command entry and FORMAT execution (as if you did not specify the N option). Therefore, whenever you enter a System Prompt Method command without the N option, the following prompt will appear:

```
CP/M-85 Format Version 2.2.100


This program is used to initialize a disk.
All information currently on the disk will be destroyed.
Is that what you want? (y/n):
```

- To confirm your intention to run a FORMAT operation, enter a **Y** at this prompt. Then FORMAT will display the prompt:

```
Put the disk you wish to be formatted in drive x.
Press RETURN to begin, anything else to abort.
```

To begin execution of the formatting process, insert the appropriate disk in drive x (where x is the drive you specified in the command line) and enter a carriage return.

NOTE: You can abort the FORMAT operation by pressing any key other than Y at the first FORMAT prompt, or by pressing any key other than the RETURN key at the second FORMAT prompt. When you abort a FORMAT operation in either fashion, CP/M displays the system prompt.

## 2.4 System Prompt Method Examples

**A>FORMAT B:[2S]↵**

FORMAT will prepare the surface of the disk in drive B (a 5.25-inch disk) to double density and on both sides, as specified by options. FORMAT will display prompts before formatting and test the disk surface while formatting, by default.

**A>FORMAT B:↵**

FORMAT will prepare the surface of the disk in drive B (a double-sided 8-inch disk). Due to the disk manufacturer's recommendation, this disk will be formatted on both sides. By default, this disk will be formatted to double density. Also by default, FORMAT will display prompts before formatting and test the disk surface while formatting.

**A>FORMAT B:[2S,1S]↵**

If your command line contains contradictory options, FORMAT will acknowledge the last one. Hence, in this case, FORMAT will format the surface of the disk in drive b (a 5.25-inch disk) on only one side, as specified by the last side quantity option. FORMAT will also display prompts before formatting and test the disk surface while formatting, by default.

A>**C:format B:[Sd,f,N]**↵

The FORMAT utility, in this case, is stored on the disk in non-default drive C. It will prepare the surface of the disk in drive B (a single-sided 8-inch disk) to single density, as specified by the "Sd" option. Since the disk was manufactured for single-sided data storage, only one side will be formatted. The "f" option specifies that this formatting operation will be performed without a disk media test. The "N" option specifies that FORMAT will not prompt you to confirm your intentions before the formatting operation begins.

NOTE: As with any other command entered at a CP/M system prompt, you can edit a FORMAT command line with the **DELETE** key, or erase the entire command line by holding down the **CTRL** key while pressing the **X** key.

# 3. DISK CAPACITIES

The following three tables show the amount of file space remaining on various kinds of disk after they are formatted under various kinds of conditions. (The FORMAT utility also prepares areas of the disk for the recording of the CP/M system core and the disk file directory, although the space reserved for such software items is not included in these tables.)

The following table shows the file capacities of 5.25-inch, soft-sectored disks formatted in 48 TPI drives:

| | Single-sided | Double-sided |
|---|---|---|
| Double density | 148 kilobytes | 304 kilobytes |

This table shows the file capacities of single-sided 8-inch disks:

| | Single density | Double density |
|---|---|---|
| Single-sided | 241 kilobytes | 482 kilobytes |

This table shows the file capacities of double-sided 8-inch disks:

| | Single density | Double density |
|---|---|---|
| Double-sided | 490 kilobytes | 980 kilobytes |

NOTE: The FORMAT utility supplied with this CP/M version cannot format any disk to extended double density. However, this CP/M version can be used to read data from (but not write data to) most disks that have been formatted at any density by earlier CP/M versions.

# 4. FORMAT ERROR MESSAGES

```
Drive x: not available in current configuration
```

EXPLANATION: If you entered a drive name (x:) that does not exist in the hardware environment, enter a different drive name.

```
Unable to format this disk.  It is write protected.
Do you have any more disks to format? (y/n):
```

EXPLANATION: Disks must be write-enabled before they can be formatted. Write-enable a 5.25-inch disk by removing the adhesive tab from its write-enable notch, and write-enable an 8-inch disk by attaching a write-enable tab to its write-protect notch.

```
Unable to format this disk. Place a different disk in the drive and press any key to begin...
```

EXPLANATION: The disk to be FORMATted is damaged or improperly inserted in the drive. Try again or replace the disk.

```
Media error
```

EXPLANATION: The disk to be FORMATted is damaged or improperly inserted in the drive. The user should try the operation again and relace the disk if the message appears again.

```
Wrong type of media, or media inserted improperly,
or media damaged.
```

EXPLANATION: You may have tried to FORMAT a hard-sectored disk in a soft-sectored drive. Check that the proper type of disk is being used. If the proper disk type is being used, then it may be damaged. Use a different disk and try again.

```
ILLEGAL FORMAT OPTION
```

EXPLANATION: System Prompt Method command line was entered with undefined characters in place of options.

```
ILLEGAL COMMAND SYNTAX
```

EXPLANATION: System Prompt Method command line was entered with undefined characters in place of options.

```
DISK IS NOT OF TYPE SPECIFIED
```

EXPLANATION: A System Prompt Method command line specified a drive that contained a disk which did not match the specified disk type.

```
OPTION NOT AVAILABLE
```

EXPLANATION: A System Prompt Method command line included option characters which were not possible under the circumstances. Reenter command with pertinent options.

# LIST

*The Utility that Prints Text File Contents on Paper*

The LIST utility enables you to obtain paper copies of files by entering a command (1) for one or more files (2) to be printed. Special print-out characteristics can be set when you enter the command with LIST parameters (3). You can stop a LIST print-out in progress (4). LIST is used to print out only certain types of files (5).

## 1. METHODS OF ENTERING LIST COMMANDS

Two different methods can be used to enter LIST commands: the Utility Prompt Method and the System Prompt Method.

## 1.1 Utility Prompt Method

With the Utility Prompt Method, you enter **LIST** in response to the system prompt. This entry is sufficient to invoke LIST, which displays its own prompt — the asterisk (*). You can now enter the argument portion of the command line in response to the "*" prompt supplied by LIST. LIST prompt entries are made in the following form:

A>**LIST**↵

*{**argument**}↵

Where {argument} is the name of the file(s) to be LISTed.

After the LIST operation is finished, LIST again displays the "*" prompt. You can enter another argument or return to the operating system by entering a carriage return.

## 1.2 System Prompt Method

To invoke a LIST operation with a single command line, you must include the argument in the response to the system prompt, as in the following example:

A>**LIST {argument}**↵

Where {**argument**} is the name of the file(s) to be LISTed.

After LIST finishes the latter printing operation, it automatically returns you to the operating system, and the system prompt is displayed.

For example, the following command, entered using the Utility Prompt Method:

A>**LIST**↵

***REPORT.DOC**↵

and this command, entered using the System Prompt Method:

A>**LIST REPORT.DOC**↵

will produce the same results. Both commands will produce a paper copy of the file named "REPORT.DOC".

# 2. PRINTING CONTENTS OF MORE THAN ONE FILE

Several files can be specified in a single LIST command. To initiate a LISTing of several files, enter the names of the files in the argument, separated by single spaces.

If the files to be LISTed reside on different disks, their names should be preceded by a drive specification (drive letter and colon).

Both of the following examples demonstrate how to LIST the contents of the specified files:

A>**LIST**CR

*PRINTOUT.DOC B:PROGRAM.PRN B:REPORT.DOC C:SYSTEMX.PRNCR

or

A>**LIST PRINTOUT.DOC B:PROGRAM.PRN B:REPORT.DOC C:SYSTEMX.PRN**CR

# 3. LIST PARAMETERS

You can specify parameters in a LIST argument to alter the characteristics of a standard print-out. These parameters allow you to select print-out characteristics such as the date, the number of copies desired, the width of tabs, etc. If no parameters are specified, LIST will print a document with default value characteristics.

The parameters used to specify print-out characteristics are shown in Table 2-2.

| PARAMETER NAME | KEYBOARD ENTRY | DESCRIPTION OF PRINT-OUT CHARACTERISTIC | DEFAULT VALUE |
|---|---|---|---|
| Date | **[D xxx...x]** | First 10 characters of specified date printed on upper right corner of each document page. | no date |
| Heading | **[H xxx...x]** | First 60 characters of specified heading printed on at left of top line of each document page. | file name (first copy only) |
| No Heading | **[N]** | No heading or date printed on any document page. | file name |
| Lines per Page | **[L nn]** | Each document page has the specified number of lines. Specification of zero lines /page causes printing without page breaks. | 60 lines per page |
| Tab Stop Width | **[T nn]** | Each tab stop within text is expanded or contracted to specified number of spaces. | 8 spaces |
| Page Number | **[P nn]** | Page numbering sequence begins with specified number on first file page. | page 1 |
| Upper Case | **[U]** | All letters in document printed in upper case | upper and lower case |
| Copies Desired | **[C nn]** | Specified number of copies are printed. | 1 copy |
| Erase | **[E]** | File is erased from disk after LIST operation is completed. | file retained on disk |

**Table 2-2**
List Parameters

## USE OF LIST COMMAND LINE PARAMETERS

Parameters are entered, enclosed in square brackets, after the last file name in the LIST command. If more than one parameter is entered, each should be enclosed in a set of brackets.

A LIST command entered with parameters using the LIST Prompt Method might appear as follows:

```
A>LIST<CR>
*PRINTOUT.DOC PROGRAM.PRN [H Today's Work] [D 31-Feb-81] [P 9]<CR>
```

This command will cause the contents of the files PRINTOUT.DOC and PROGRAM.PRN to be printed, with the following heading across the top of the first page of each file:

```
Today's Work                                31-Feb-81 Page 9
```

The parameters will take effect on all of the files specified in that command line.

When the command is entered using the Utility Prompt Method, any parameters entered (except for the starting Page number parameter and Erase parameter) will remain in effect with any files specified at a successive asterisk (*) prompt, until new values are entered for the parameters, or until control is returned to the operating system.

When a LIST command line is entered using the System Prompt Method, letters in the heading, date, and page number line will be automatically translated into UPPER CASE. The following System Prompt command demonstrates this character translation:

```
A>LIST PRINTOUT.DOC PROGRAM.PRN [H Today's Work] [D 31-Feb-81] [P 9]<CR>
```

This command will cause the contents of the files PRINTOUT.DOC and PROGRAM.PRN to be printed with the following heading across the top of the first page of each file:

```
TODAY'S WORK                                31-FEB-81 PAGE 9
```

# 4. ABORTING A LIST OPERATION

After a LIST command has been entered, the print-out can be aborted by pressing any keyboard character while the print-out is in progress. (If more than one copy has been requested, you must abort each copy.) LIST will then redisplay the asterisk prompt (under the Utility Prompt Method); or CP/M will display the system prompt (under the System Prompt Method).

# 5. FILES THAT SHOULD BE LISTED

Only files containing ASCII characters should be LISTed. ASCII character files include files composed using a text editor (such as ED), or file with the "PRN" extension created by an assembler (such as ASM).

Files not composed of ASCII characters will produce meaningless print-outs if LISTed.

Files composed in a word processor will LIST, but they might possess features (such as bold face characters or page breaks) that LIST will not print in the same way as the word processor's print-out command.

# 6. LIST ERROR MESSAGES

```
File not found
```

EXPLANATION: A file specified in the LIST command argument does not exist on the disk in the logged drive. If the file does exist on a disk, that disk should be inserted in a drive and the appropriate drive should be specified before the file name in the argument.

```
Syntax error in command line
```

EXPLANATION: The LIST command was improperly entered. Often occurs if a space is not entered between file names, or when an invalid parameter is specified.

# LOAD

*The Utility that Loads a Hexadecimal File for Execution*

The LOAD utility puts an assembled hexadecimal file in the Transient Program Area (1), and translates the file into a "COM" file, which is executable under the CP/M operating system (2). Only certain types of files can be LOADed (3).

## 1. LOAD INVOCATION

To LOAD a hexadecimal file into the TPA, you should respond to the system prompt with this entry:

A>**LOAD {hex file}**CR

Where {**hex file**} is the primary file name of an Intel hexadecimal file. The file extension is omitted from this entry because LOAD always assumes the extension to be "HEX".

This command translates the hex file into machine code that can be executed under CP/M, gives it a "COM" extension, and writes it back to the disk.

## 2. EXECUTING A LOADED PROGRAM

The user can start execution of the program by entering the primary name of the file (and a carriage return) in response to the system prompt. Thus the file is executed as if it were a regular CP/M utility.

It is only necessary to LOAD a hex file once. The "COM" version of the file will be stored on the disk from which the "HEX" version came. Thereafter, CP/M treats it like another "COM" file.

The operation can take place on a non-default drive if the file name is prefixed by a drive name. The entry of the following command:

A>**LOAD B:BETA**CR

brings the LOAD program into the TPA from the default disk and then operates on the file "BETA.HEX", which resides in drive B. The file named "BETA.COM" is written to the disk in drive B. (The file "BETA.HEX" will also remain on the disk.) Now you can execute BETA.COM by responding to the system prompt as shown:

A>**B:BETA**CR

Thus, you can develop new CP/M commands by using LOAD to translate a "HEX" file produced with the ASM utility (see Page 2-3) into a CP/M-executable "COM" file.

## 3. LOAD REQUIREMENTS FOR HEXFILES

To be LOADed, a file must contain valid Intel hexadecimal format records. The ASM utility can be used to produce such a file from a file with the "ASM" extension. The hex file must begin at address 100H, which is the memory location at which the Transient Program Area (TPA) begins. In addition, the addresses in the hex records must be in ascending order. Gaps in memory regions are filled with zeroes by the LOAD command, as the hex records are read. Thus, LOAD must be used only for creating CP/M command files, which operate in the TPA. Program files that occupy memory locations other than the beginning of the TPA (address 100H) can be loaded using the DDT utility. After a program is LOADed, LOAD displays a list of hexadecimal numbers in the following form:

```
FIRST ADDRESS nnnn
LAST  ADDRESS nnnn
BYTES READ    nnnn
RECORDS WRITTEN nn
```

# 4. LOAD ERROR MESSAGES

```
INVALID HEX DIGIT
```

EXPLANATION: The hexadecimal file you tried to LOAD contains upper bits set in ASCII words.

```
ERROR ADDRESS nnnn
```

EXPLANATION: An error occurred at address "nnnn".

```
CHECK SUM ERROR
```

EXPLANATION: The hexadecimal file did not produce the correct check sum during LOAD execution.

```
CANNOT OPEN SOURCE
```

EXPLANATION: The hexadecimal source file specified in the LOAD command line is not present on the specified disk.

```
NO MORE DIRECTORY SPACE
```

EXPLANATION: The directory on the disk is full.

```
CANNOT CLOSE FILES
```

EXPLANATION: The LOADed file is not present when LOAD tries to close the "COM"file.

```
INVERTED LOAD ADDRESS, LOAD ADDRESS nnnn
```

EXPLANATION: The LOADed hexadecimal file did not start at the beginning of the TPA (0100H) as it should have.

# MVCPM207

*The Utility that Customizes a CP/M System Kernel for Memory Size*

The MVCPM207 utility will adjust the CP/M system kernel so that it has the proper memory size for your purposes. It can change the system's size within a range of 48 through 64 kilobytes of Random Access Memory (RAM). This utility should be followed immediately with another utility or command, such as SYSGEN or SAVE.

## 1. FUNCTION OF MVCPM207

MVCPM207 loads the kernel of a CP/M Operating System (the part exclusive of BIOS files BIOS85.SYS and BIOS88.SYS) into a special location in computer memory. At this location, it adjusts the system kernel to either a specified memory size or the total available memory size of the computer.

MVCPM207 must also find and measure the BIOS85.SYS file that will eventually be used with the system kernel, to allow sufficient space for this BIOS file. MVCPM207, however, always relies upon a SYSGEN or SAVE command to copy the system kernel that MVCPM207 loaded into memory.

## 2. MVCPM207 COMMAND LINE ENTRY

The MVCPM207 command line is entered in the following form, with two optional specifications separated by a space, as shown:

A>**MVCPM207** {**nn**} {**d**}:{**biosfile.ext**} ⏎

Where the {**nn**} variable represents the memory size that the transferred system kernel will occupy, in multiples of 1024 bytes (kilobytes). This is an optional value. If the "*" character or no value is entered, the system kernel will be set to occupy the entire memory capacity of the computer being used, by default. (Your Z-100 computer, as used with this version of CP/M, has a memory range of 48K through 64K.) This value can be less than or equal to the actual memory capacity of the computer. If it is larger than the computer's capacity, then the CP/M system created will be useless in the computer;

where the {**d**} variable represents the letter of the disk drive containing the BIOS files that are to be matched up with the system kernel being moved. This variable is optional. If omitted, MVCPM207 will assume that the created system kernel should be modified to be compatible with the BIOS files that are currently active in computer memory; and

where the {**biosfile.ext**} variable represents the name of the file containing the system components normally stored in the file BIOS85.SYS. This variable is optional. If omitted, MVCPM207 will assume the file name "BIOS85.SYS".

NOTE: The "*" character must be entered when you specify **no** value for the memory {"nn"} variable, and **do** specify a value for the drive name {d} and/or BIOS file name {biosfile.ext}. In this sort of command line, the "*" character acts as a "place holder" so that your drive name and/or file name parameters are not interpreted as a memory value because it was entered in the memory value space.

During execution, the MVCPM207 utility will respond with a message in the following form:

```
MVCPM207 VERSION 2.2.100

CONSTRUCTING nnk CP/M vers 2.2.100
READY FOR "SYSGEN" OR
"SAVE 38 CPMnn.COM"

A>
```

# 3. MVCPM207 EXAMPLES

The following command lines and explanations are specific examples of MVCPM207 command entry.

| | |
|---|---|
| A>**MVCPM207 48** ⏎ | The system kernel created by this command will operate with 48K of RAM. The kernel will be adjusted using the BIOS that was loaded into computer memory at bootstrap for reference. |
| A>**MVCPM207 * C:**⏎ | The system kernel created by this command will probe computer memory and operate at computer's memory capacity (64K). This kernel will be adjusted using the BIOS files stored in drive C: for reference. |

# 4. AFTER RUNNING MVCPM207 . . .

MVCPM207 should be immediately followed by either:

- The SYSGEN utility, which will transfer the adjusted CP/M system kernel to the system tracks of a specified disk; or by
- The SAVE resident command, which will transfer the adjusted CP/M system kernel to a file on a specified disk.

If the user performs any **other** CP/M activity immediately after running MVCPM207, the work of MVCPM207 will probably be destroyed.

# 5. MVCPM207 ERROR MESSAGES

```
INVALID MEMORY SIZE
```

EXPLANATION: Valid memory sizes are between 48K and 64K.

```
SYNCHRONIZATION ERROR
```

EXPLANATION: The serial number of the MVCPM207 utility used must match that of the CP/M Operating System used. Reset, cold boot the system and try again with a matching system and utility.

```
READ ERROR
```

EXPLANATION: MVCPM207 cannot read data from a file the user specified because the file and/or disk surface is flawed.

```
NO FILE
```

EXPLANATION: MVCPM207 cannot read data from a file the user specified because it cannot find the file on the specified drive.

```
NO SPACE
```

EXPLANATION: The BIOS files the user specified will not fit in memory.

```
BAD LOAD
```

EXPLANATION: A file specified by the user did not load properly. The user should try the MVCPM207 command again or specify a different file.

```
1420OCAN'T OPEN BIOS85.SYS & BIOS88.SYS
```

EXPLANATION: MVCPM207 is unable to use the specified BIOS because it is not stored in files specifically named "BIOS85.SYS" and "BIOS88.SYS", because it is not stored on the drive A disk, or because it is stored in a defective file.

```
FATAL ERR F25: NOT ENUF MEMORY
```

EXPLANATION: MVCPM207 execution was aborted. Reset, cold boot the system and try again.

```
File not found.
```

EXPLANATION: MVCPM207 could not find the file that the user specified as the BIOS on the specified disk.

```
UNABLE TO READ BIOS FILE
```

EXPLANATION: An inaccessible BIOS file was specified at the end of a MVCPM207 command line. Copy the specified file to an accessible disk and reenter the same command, or enter another MVCPM207 command specifying a different BIOS file.

# PIP

*The Utility that Copies Data between Files, Disks, and/or Hardware Devices.*

PIP stands for Peripheral Interchange Program, the CP/M utility that can be invoked (1, 2, 3) to copy (4) and link (5) files or parts of files (6), and to direct input and output between logical devices (8). PIP also allows files to be transferred between different disks when using a one-drive hardware environment (9). PIP operations can be regulated by parameters (11).

## 1. METHODS OF ENTERING PIP COMMANDS

Two different methods can be deployed to enter PIP commands: the Utility Prompt Method and the System Prompt Method.

## 1.1 Utility Prompt Method

With the Utility Prompt Method, you enter "PIP" in response to the system prompt. This entry is sufficient to invoke PIP, which displays its own prompt — the asterisk (*). You now enter the argument half of the command line in response to the "*" prompt supplied by PIP, as shown in the following example:

```
A>PIP<CR>
*{argument}<CR>
*
```

After this PIP operation is finished, PIP again displays the "*" prompt. In reply, enter another argument to command PIP to perform another operation, or enter a carriage return alone to return to the operating system and obtain a system prompt.

## 1.2 System Prompt Method

To use PIP with a single command line, the system prompt mode of command entry can be used. With this mode, you must include the argument in the response to the system prompt, as shown in the following example:

```
A>PIP {argument}<CR>
A>
```

After PIP finishes this operation, it automatically returns the user to the operating system, and the system prompt is displayed.

NOTE: PIP sends data to a destination from a source, but it does not remove the data from the source. It merely copies an image of the data, and then sends it. Therefore, any data that is PIPped will remain intact at the source after the PIP operation.

# 2. THE ARGUMENT IN A PIP COMMAND LINE

PIP commands require an argument regardless of which invocation method is used. PIP arguments must specify a "data source" (which can be either a file or a logical input device), and a "data destination" (which can be either a file, a disk, or a logical output device). The "data destination" is entered first and followed by an "=" sign, which is followed by the "data source", as shown in these examples:

```
A>PIP<CR>
*{data destination}={data source}<CR>
```

or

```
A>PIP {data destination}={data source}<CR>
```

In all cases, PIPped data is transferred in a right-to-left direction, with respect to the command line components. The file, disk, or logical output device to receive the data is always specified on the left hand side of the "=" sign; and the file or logical input device to submit the data is always specified on the right hand side of the "=" sign.

# 3. CHARACTERISTICS OF DATA DESTINATION AND SOURCE SPECIFICATIONS

A data destination can be a file, a disk, or a logical device. A data source can be a file or a logical device. Each must be specified in the command line as shown:

| DATA SPECIFICATION CHARACTERISTICS | EXAMPLES |
|---|---|
| Data can be transferred to a disk by specifying the letter of the drive in which the disk resides, and a colon. | B:=C:SENDOVER.DOC<br>B:=C:*.* |
| Files on non-default disks are identified by specifying the drive in which they reside immediately before the file name. | B:NEW.DOC=C:SEND.DOC<br>B:CREATED.DOC=CON:<br>CON:=B:SENDOVER.DOC |
| Logical devices are specified by entering the three-letter code for that particular device and a colon. | LST:=CON:<br>LST:=SENDOVER.DOC<br>CREATED.DOC=CON: |

# 4. FILE COPYING EXAMPLES

The following argument will cause PIP to transfer a copy of the file SENDOVER.DOC from drive B to the same disk, and give it the file name CREATED.DOC. In effect, this operation creates a file backup with a different name on the same disk.

**B:CREATED.DOC=B:SENDOVER.DOC**CR

The following argument will cause PIP to transfer a copy of the file SENDOVER.DOC from drive C to the disk in drive B, and give it the file name CREATED.DOC.

**B:CREATED.DOC=C:SENDOVER.DOC**CR

The following argument will cause PIP to transfer a copy of the file SENDOVER.DOC from the currently logged drive to the disk in drive B, and give it the same file name.

**B:=SENDOVER.DOC**CR

The following argument will cause PIP to transfer a copy of the file SENDOVER.DOC from drive C to the disk in drive B, and give it the same file name.

**B:=C:SENDOVER.DOC**CR

Each of the following arguments will cause PIP to transfer a copy of the file SENDOVER.DOC from the disk in drive B to the disk in the currently logged drive, and give it the file name SENDOVER.DOC. It should be noted that the data source in this argument is **not** the disk in drive B, but the file SENDOVER.DOC which resides on the disk in drive B. Because the source file is not specified in the typical data source location, PIP assumes that the source file is the same as the destination file, which is specified in its typical location.

**SENDOVER.DOC=B:**CR

or

**SENDOVER.DOC=B:SENDOVER.DOC**CR

# 5. DATA SOURCE CONCATENATION

While only one data destination can be specified in a single PIP command line, several data sources can be specified. Thus you can merge data from several locations to one location.

When more than one data source is specified, each data source specification is separated by a comma.

The length of a PIP command line cannot exceed 255 characters. (If you try to enter a 256th character into the command line, PIP will begin execution based on the first 255 characters entered.)

PIP will concatenate data sources in the order in which their specifications are entered in the command line.

The following PIP argument will cause PIP to get a copy of the files THISFILE.DOC and THATFILE.DOC from the disk in drive C and the file YOURFILE.DOC from the disk in the default drive. PIP will then combine all three files into one file (in the order they are specified), place this file on the disk in drive B, and assign the name COMBINED.DOC to the transferred combination of the three source files.

***B:COMBINED.DOC=C:THISFILE.DOC,C:THATFILE.DOC,YOURFILE.DOC** 

NOTE: Both files and devices can be specified as data sources in the same command line.

# 6. COPYING A BLOCK OF DATA

Blocks of files, as well as whole files, can be copied to another file. To do this, you enter a PIP command that specifies the source file name and the beginning and end of the block being transferred. Such an entry requires the use of PIP parameters as in the following example:

***BLOCKFIL.TXT=WHOLEFIL.TXT[Sbeginning^ZQend^Z]**CR

In this example, "BLOCKFIL.TXT" is the name of the destination file to receive a block of text from the source file named "WHOLEFIL.TXT". The block is specified by entering a bracket, the parameter "S", a unique string of text from the "beginning" of the block, a CTRL-Z character, the parameter "Q", a unique string of text from the "end" of the block, a CTRL-Z character, and another bracket.

The following example demonstrates how you can concatenate blocks from several files, by using the "S" and "Q" parameters to specify the beginning and end of each block:

***USBLOCK.TXT=GETADD.TXT[SFour^ZQearth^Z],CONSTIT.TXT[SWhen^ZQAmerica^Z]**CR

With the entry of this command line, PIP would combine the specified block from the file "GETADD.TXT" with the specified block from "CONSTIT.TXT". With this data, PIP would create the file "USBLOCK.TXT".

NOTE: The strings following the S and Q parameters will be translated to UPPER CASE if you are using PIP by the System Prompt Method. They will **not** be translated to upper case if you are using the Utility Prompt Method.

# 7. COPYING DATA TO AN INTEL HEXADECIMAL FILE

PIP performs a special function if the data destination is a file with the "HEX" extension (an Intel hex formatted machine code file) and the data source (or one of several data sources) is an input/output device, such as a paper tape reader.

In such a case, PIP checks to ensure that the source is a properly formed Intel hexadecimal file, with legal hexadecimal values and checksum records. If an invalid input record is found, PIP reports an error message at the console and waits for corrective action.

It is usually sufficient to open the reader and rerun a section of the tape (by pulling the tape back about 20 inches). When the tape is ready for the second reading, you should enter a carriage return at the console, and PIP will attempt another read. If the tape position cannot be properly read, you should continue the read (by entering a carriage return after the error message), and enter the record manually, using the ED utility after the disk file is constructed from the other data source components.

For convenience, PIP allows a CTRL-Z ("end-of-file"character) to be entered from the console if the source file is a RDR: device. In such a case, PIP reads the specified device and monitors the keyboard. If a CTRL-Z is entered at the keyboard, then the read operation is terminated normally.

For instance, the following PIP argument could be entered to create a "HEX" file:

***CREATED.HEX=CON:,B:SENDOVER.HEX,PTR:**

The preceding argument will cause PIP to create the data destination file CREATED.HEX by reading the data from source device CON: (as the user enters hexadecimal values at the keyboard and eventually enters a CTRL-Z "end-of-file" character), the data from source file SENDOVER.HEX (which includes a CTRL-Z), and finally the data from source device PTR: (until a CTRL-Z is read from the paper tape).

NOTE: Hexadecimal data can be checked for valid format by specifying the "H" or "I" format in the PIP argument (see Page 2-162).

# 8. COPYING DATA TO OR FROM LOGICAL DEVICES

The following examples, and explanations, demonstrate PIP commands that transfer data to and/or from computer-recognized logical devices:

***LST:=SENDOVER.PRN**CR

The preceding argument causes PIP to copy the source file SENDOVER.PRN to the LST:device.

***CON:=B:THISFILE.ASM,C:THATFILE.ASM,YOURFILE.ASM**CR

The preceding argument causes PIP to concatenate three ASM source files: one from the disk in drive B, one from the disk in drive C, and one from the default disk. PIP will send a concatenation of data from these sources to the CON: device. Thus the contents of the files will be displayed on your console.

***LST:=B:PROMPT.DAT,CON:** CR

The preceding argument causes PIP to send data from a drive B file to the list device until the end of the file, then from the console keyboard to the list device until you enter the CTRL-Z end-of-file character.

## 8.1 Input/Output Devices Accessible Through PIP

The PIP utility allows you to transfer data directly to or from the logical and physical devices being used. PIP supports data transfer with respect to the devices indicated in the following table:

| LOGICAL DEVICE NAME | PHYSICAL DEVICE NAME | DESCRIPTION AND/OR CATALOG NAME OF RECOMMENDED INPUT/OUTPUT MACHINE |
|---|---|---|
| CON: | TTY: | A printing terminal attached to serial port outlet A on the Z-100 (e.g. Decwriter). |
| | CRT: | A video display terminal and keyboard. |
| | BAT: | A batch pseudo-device using RDR: for input and LST: for output. |
| | UC1: | A modem attached to serial port B on the Z-100. |
| RDR: | TTY: | A printing terminal attached to serial port outlet A on the Z-100 (e.g. Decwriter). |
| | PTR: | Not implemented. |
| | UR1: | A modem attached to serial port B on the Z-100. |
| | UR2: | A video display terminal and keyboard. |
| PUN: | TTY: | A serial printer attached to serial port outlet A on the Z-100. |
| | PTP: | Not implemented. |
| | UP1: | A modem attached to serial port B on the Z-100. |
| | UP2: | A video display terminal and keyboard. |
| LST: | TTY: | A serial printer attached to serial port outlet A on the Z-100 (H/Z-25, H-14, TI-810, Diablo, Epson MX-80 serial, Decwriter). |
| | CRT: | A video display terminal and keyboard. |
| | LPT: | A parallel printer attached to the parallel printer port outlet on the Z-100 (Epson MX-80 parallel, Centronics). |
| | UL1: | A modem attached to serial port B on the Z-100. |

**Table 2-3**
Logical/Physical Devices

The user must be certain that the destination device specified is capable of receiving data, and that the specified source device is capable of sending data.

## ADDITIONAL LOGICAL DEVICES USED WITH PIP

All devices in the preceding table can also be referenced using the STAT utility, which assigns physical devices to logical devices on a temporary basis. (CONFIGUR can match the devices in Table 2-3 on a permanent basis.)

The PIP utility can also gain access to five additional devices, which are defined below:

NUL: Sends 40 "nulls" (ASCII zeroes) to the destination device. (The NUL: device is usually accessed at the beginning and/or end of the output when a paper tape punch device is used.)

EOF: Sends an "end-of-file" character (ASCII CTRL-Z) to the destination device. (A CTRL-Z is sent automatically when PIPing a file composed with ASCII characters.)

INP: Special input source which can be "patched" into the PIP program itself. Through this source, PIP accepts data input character-by-character using a system call to memory location 103H. The data returns from location 109H. The parity bit must be preset at zero.

OUT: Special output destination which can be "patched" into the PIP program. PIP transmits data from register C to this destination using a system call to memory location 106H. It should be noted that locations 109H through 1FFH of the PIP memory image are not used, and can be replaced by special purpose drivers using DDT.

PRN: This device is accessed for the same purposes as the LPT: device. In its operation, however, tabs are expanded at every eighth character position, lines are numbered, and page breaks occur every 60 lines. Output directed to the LPT: device will be treated identically if the [TNP] or [T8NP60] parameter is used in a LIST command (see Page 2-138).

## 8.2 Suspending PIP Operations

When PIPing to the CON: logical device, the copy operation can be suspended by entering the CTRL-S character at the keyboard, and resumed by entering any character other than CTRL-C. The operation can be aborted by entering any keyboard character other than CTRL-S while data is being transferred. PIP will respond to such an entry with the following message mentioning the name of the file being displayed:

```
ABORTED : {filename.ext}
```

# 9. USING PIP WITH A ONE-DRIVE HARDWARE ENVIRONMENT

When performing a PIP operation in a one-drive hardware environment, the "PIP" command should be entered alone at the system prompt (using invocation method 1.1) to produce the "*" prompt.

When using the PIP utility to copy a file to a file on another disk, the disks involved must be inserted in the drive alternately to allow PIP to read from one file and then write to another. The command line argument should be entered as follows:

***B:{destination file name}=A:{source file name}**CR

Where **B:** specifies logical drive device B;

where **A:** specifies logical drive device A; and

where both A and B are referred to as physical drive device 0 (the only physical drive device in the hardware environment).

This entry causes PIP to display the message:

```
PUT DISK B IN DRIVE A: AND PRESS RETURN
```

The disk designated to receive the file copy should be placed in the drive, and a carriage return entered. PIP will display the following message:

```
PUT DISK A IN DRIVE A: AND PRESS RETURN
```

The disk containing the data source file should be placed in the drive, and a carriage return entered. You should repeat these two steps alternately, as the PIP prompts indicate, until the entire source file has been copied to the destination file. This process will vary in length, depending on the size of the file being copied. You should keep track of which disk is A and which is B to avoid errors.

# 10. PIP'S METHOD OF OPERATION

File names and device names can both be used in a single data source argument, with PIP reading each name entered (starting with the name entered on the left-hand side of the source argument) until it reaches "end-of-file". "End-of-file" is indicated by a CTRL-Z character in ASCII files, and by the actual end of the file in non-ASCII files. Each of the data source names entered is read and then concatenated to the preceding name (to its left) until the last name has been read.

The data destination device or file receives a copy of the data from the source files and/or devices. When data from an ASCII file is written, the CTRL-Z "end-of-file" character is appended to the result.

As the PIP operation begins, a temporary file is established in the directory of the destination disk. This file has a name that consists of the primary name specified in the data destination, and a "$$$" extension. This temporary file is not changed to the actual file (with the extension specified in the command line argument) until successful completion of the PIP operation.

Files with a "COM" extension are always assumed to be non-ASCII files.

# 11. PIP PARAMETERS

The PIP utility performs enhanced data transfer operations when command line parameters are used. All PIP parameters used are to be entered after the data source and enclosed between a single set of square brackets. A carriage return must follow the closing bracket. Parameters are entered in the following form:

*{**data destination**}={**data source**}{[**parameters**]}<CR>

Where {**data destination**} is a specification of the file, disk, or device to receive the transferred data;

where {**data source**} is one or more specifications of the file(s) or device(s) from which the transferred data will be copied; and

where {[**parameters**]} is the specification of PIP parameters used to regulate the transfer operation.

NOTE: Several parameters can be enclosed between a set of brackets in a single PIP command line.

The parameters used to regulate transfer operations are as follows:

**B** Block mode transfer: PIP transfers data from a source device to a buffer (data reservoir in computer memory), which retains bits of a continuous flow of data before writing it on a disk. The buffer is emptied to a disk when the CTRL-S character is transmitted to the buffer from the source device. PIP then resumes transmitting the continuous flow of data from the source device. If the buffer overflows with data, PIP will display an error message.

**Dn** Delete characters: Causes PIP to transfer all of the characters within the source file or source device — except for the characters to the right of the nth column, which are truncated so that the file can be sent to a narrow printer or console.

**E** Echo transfers: Causes display indicating what units of data are being transferred, as they are being transferred.

**F** Filter Form Feeds From File: Removes form feeds that are embedded in a file. The "P" parameter can be entered in the same command line to insert new form feeds.

**Gn** Get file from user area "n": Enables you to access files through a different user area number "n" (in the range 0-15). Can only be implemented if the file "PIP.COM" is in the current user number.

**H** Hex data transfer: Transferred data is checked for proper Intel hexadecimal file format, and unnecessary characters between hex records are removed during the transfer operation. If errors occur, PIP will display prompts for corrective action.

**I** Ignore records: Transferred Intel hex files will not include ":00" records. The I parameter automatically sets the H parameter.

**L** Lower case letters: Translates all PIPped alphabetic characters to lower case.

**N** Number lines: Applies line numbers to the beginning of each transferred line of data. The numbers begin at one and increase sequentially by ones. The numbers are separated from data lines by a colon. If "N2" is specified, then each number is preceded by up to five zeros and followed by a tab space. This tab space is expanded if the "T' parameter is also set.

**O** Object file transfer: The normal CP/M end of file will be ignored in any data transferred with this parameter. This parameter is useful in transferring files with non-ASCII characters.

**Pn** Page ejects: Transferred data is interrupted at the beginning of the file and every "n" lines thereafter. If "n" is omitted from the parameter, the value "1" is assumed. If the value "1" is entered or assumed with the parameter, page ejects will occur every 60 lines. If the "F" parameter is entered in the same command line, then form feed suppression takes place before the new page ejects are inserted.

**Qstring^Z** Quit copying block: the text "string" between the "Q" and the "^Z" (CTRL-Z) in this parameter marks the end of a block of text to be transferred. (Block begins with text "string" in the "Sstring^Z" parameter.)

**R** Read system files: Enables transferral of files with the "SYS" status.

**Sstring^Z** Start copying block: the text "string" between the "S" and the "^Z" in this parameter marks the start of a block of text to be transferred. (Block ends with text "string" in the "Qstring^Z" parameter.)

NOTE: The strings following the S and Q parameters are translated to UPPER CASE by the system if PIP is invoked using the System Prompt Method (entire command on one line):

A>**PIP B:THIS.DOC=C:THAT.DOC[STHE BEGINNING^ZQTHE END.^Z]**

Therefore, the file's actual text must be in upper case to match the string. However, if PIP is invoked using the Utility Prompt Method (with the command line argument entered in response to PIP an "*" prompt):

A>**PIP**
***B:THIS.DOC=C:THAT.DOC[STHe beginning^ZQthe end.^Z]**

Then the string text will **not** be automatically translated into upper case.

**Tn** Tab expansion: Expands each occurrence of tab character to every "n"th column in transferred text.

**U** Upper case: Translates lower case alphabetic characters to upper case during text transferral.

**V** Verify operation: compares copy of transferred data with original to ensure exact correspondence. Might increase command execution time by as much as two times. This parameter only works if the data destination is a disk file.

**W** Write over files: Writes newly transferred data file over old data file, even if old file has been given R/O (read only) status. PIP will not prompt the user before deleting the old file.

**Z** Zero parity bit: Sets parity bit on input to 0 for each ASCII character in transferred data.

# 12. PIP ERROR MESSAGES

```
DISK READ ERROR
```

EXPLANATION: Source of data is flawed. Repeat PIP command or replace data source.

```
DISK WRITE ERROR
```

EXPLANATION: Destination disk did not have enough space for PIPped data. Clear some space on the destination and repeat PIP command.

```
VERIFY ERROR
```

EXPLANATION: Destination disk has flawed media. Replace destination disk and repeat PIP command.

```
ABORTED: {filename.ext}
```

EXPLANATION: A keyboard entry during a PIP data transfer caused this message and an end to PIP execution while {filename.ext} was being copied to the CON: device. Reenter the same command.

```
BAD PARAMETER
```

EXPLANATION: You entered an invalid parameter or an invalid bracket in a PIP command line.

```
INVALID USER NUMBER
```

EXPLANATION: You specified a user number out of the 0-15 range.

```
INVALID DIGIT
```

EXPLANATION: You entered an invalid digit for a PIP parameter. Reenter the entire command line.

```
CHECKSUM ERROR
```

EXPLANATION: PIP verification found a discrepancy between a data source and data destination file. The PIP operation should be repeated.

```
INVALID FORMAT
```

EXPLANATION: You entered PIP command line in improper form, and should repeat entry of command.

```
NO DIRECTORY SPACE
```

EXPLANATION: The directory of the destination disk is full.

```
NO FILE: =x:filename.ext
```

EXPLANATION: PIP could find no source file by the name "filename.ext" on the disk in drive "x".

```
START NOT FOUND:={filename.ext}{[param]}
```

EXPLANATION: PIP could not find the beginning of a string specified with the "S" and "Q" parameters. Reenter command specifying a string beginning that exists in the data source.

```
QUIT NOT FOUND:={filename.ext}{[param]}
```

EXPLANATION: PIP could not find the end of a string specified with the "S" and "Q"' parameters. Reenter command specifying a string end that exists in the data source.

```
DESTINATION IS R/O, DELETE (Y/N)?
```

EXPLANATION: This message prompts you that the destination already has a file by that name which has been given R/O status (see Page 2-180). You can have the old file overwritten by answering Y to this prompt.

```
NOT FOUND
```

EXPLANATION: Reenter PIP command with the name of an accessible data source.

```
UNRECOGNIZED DESTINATION
CANNOT WRITE
```

EXPLANATION: Reenter a PIP command with a valid data destination.

```
INVALID PIP FORMAT
CANNOT READ
```

EXPLANATION: You made a syntax error in command line, and PIP could not read the source file. Reenter the command after checking the invalid command for syntax errors.

```
INVALID SEPARATORS
```

EXPLANATION: Reenter PIP command with the proper square brackets around a parameter.

# PREL

*The Utility that Creates a Relocatable File from Two Hexadecimal Output Files*

The PREL utility is invoked (1) to take the hex output files of two assemblies and generates a relocatable file from them (2). The first of the assemblies should have an initial origin of 000H and the second assembly should be one page higher, or 0100H. The hex output of the first assembly should be given a file name extension of "HX0", and the hex output of the second should have "HX1" for its extension. PREL can be effectively implemented as one of the batched commands in a SUB file (3).

## 1. PREL INVOCATION

A PREL command line is entered in this form:

A>**PREL {hex file name} {relocatable file name}**CR

Where {**hex file name**} is the primary name of the two assembled hex files (These files are given the extensions HX0 and HX1); and

where {**relocatable file name**} is the primary name of the output page relocatable file. (The output file will have the extension PRE.)

For example, the command line:

A>**PREL MYPROG TESTBIOS**CR

would create the file TESTBIOS.PRE from the two hex files MYPROG.HX0 and MYPROG.HX1.

# 2. PREL FUNCTION

PREL will compare the two hex files and perform two separate functions:

- PREL converts the hex file to a binary file (similar to the function of the LOAD utility.)

- PREL appends a relocation table to the end of the file which includes a bit for each byte in the module. If a particular bit has a value of one, then the associated byte must be offset when the program is loaded into memory. An error message will result if the hex values for a given byte differ by more than 0 or 1. An error message will also result if the addresses in the hex files do not increase sequentially.

# 3. PREL OPERATION

All addresses that need to be relocated should be expressed relative to the base of the module, and not defined absolutely. A convenient way to do this is to omit "ORG" statements from the program and use the SUBMIT utility to enter batched commands from a SUB file. The commands in the SUB file should perform the required assemblies and the conversion to a relocatable format. An example of this type of batched command might be:

```
PIP TEMP0.ASM=ORG0.ASM,$1.ASM
ASM TEMP0.AAZ
REN $1.HX0=TEMP0.HEX
ERA TEMP0.ASM
PIP TEMP1.ASM=ORG1.ASM,$1.ASM
ASM TEMP1.AAZ
ERA TEMP1.ASM
REN $1.HX1=TEMP1.HEX
PREL $1 $1
ERA $1.HX0
ERA $1.HX1
```

where the files ORG0.ASM and ORG1.ASM contain nothing but "org 0000H" and "org 0100H" statements respectively. This command script could be invoked with the entry:

A>**SUBMIT MAKEPRE MYPROG** 

Where the command sequence is contained in a file called MAKEPRE.SUB, the original assembly file is called MYPROG.ASM and the final result is a page relocateable file called MYPROG.PRE.

# 4. PREL ERROR MESSAGES

```
Phase error
```

EXPLANATION: Hexadecimal files do not contain correct starting addresses.

```
Cannot open input file
```

EXPLANATION: The file with the extension "HX0" or "HX1" was not found.

```
Cannot create output file
```

EXPLANATION: The directory on the destination disk is full.

```
Hex files not monotonic
```

EXPLANATION: The hexadecimal addresses are not increasing sequentially.

```
Write error
```

EXPLANATION: The destination disk is full.

# REN

*The Resident command that Renames Files*

The REN command allows you to assign a new name to any disk file (1). If it cannot RENname a file, it displays the reason (2).

## 1. RENAMING A FILE

To RENname a file, you respond to the system prompt with a command line in the following form:

A>**REN {new name}={old name}**↵

Where **{new name}** is the name the user wishes to assign to the file; and

where **{old name}** is the name of the file before it is RENamed.

For instance, the name of the file "MYFILE.DAT" can be assigned to a file currently named "YOURFILE.DAT" by entering the following command:

A>**REN MYFILE.DAT=YOURFILE.DAT**↵

If the file receiving the new name does not reside on a disk in the default drive, then the REN command line must include a drive specification which precedes the new file name, as shown:

A>**REN C:MYFILE.DAT=YOURFILE.DAT**↵

# 2. CONSOLE RESPONSE TO RENAMING COMMANDS

If a file is successfully RENamed, the system prompt will appear.

```
A>
```

If the file being renamed does not exist on the default drive and no drive is specified (or if an incorrect drive is specified), then REN cannot find the file you want to rename and will display the error message:

```
NO FILE
```

If you try to assign a file a name that is actually the current name of an existing file, REN will not perform the change and displays the error message:

```
FILE EXISTS
```

# SAVE

*The Resident Command that Copies Data from Memory to a Disk File*

The SAVE command is used to copy the contents of a span of memory locations to a disk file (1). The amount of data that SAVE will copy is measured in pages (2). The data that SAVE copies from the computer's memory space can come from a variety of sources (3).

## 1. SAVE COMMAND LINE

To SAVE a program onto a disk and create a file for it, you respond to the system prompt with a command in the form:

A>**SAVE {pages} {file name}**↵

Where {**pages**} is the **decimal** number of pages of memory contents that SAVE should copy to the disk, and

where {**file name**} is the name of the file created to store the data.

NOTE: A page of memory in CP/M is 256 (decimal) bytes, or a span of 0100H (hexadecimal) memory locations.

For example, to copy 3 pages (the memory contents between address 100H and address 3FFH ) of data from main memory to the disk in non-default drive B and give this block of data the file name "PROGRAM.ASM", you enter the following command line:

A>**SAVE 3 B:PROGRAM.ASM**↵

## 2. SAVE DESCRIPTION

Data in the computer's Transient Program Area (TPA) memory space can be blocked off by specifying the first and last memory locations that it occupies. Programs loaded into the TPA always begin at memory location 0100H (hexadecimal). This is also the location from which SAVE starts copying blocks of data. Therefore, if you specify one page (100H bytes) in the SAVE command line, SAVE will start copying data from memory location 100H and continue through memory location 1FFH.

Only whole pages can be transferred; and SAVE recognizes only decimal integers for pages.

## 3. COMMON APPLICATIONS FOR SAVE

SAVE will copy the binary values for data that resides in memory locations 100H through FFFFH (in a 64K computer). This area of memory is known as the Transient Program Area, and it is used to accommodate data manipulated by utilities and application programs. Data often remains in these locations after being manipulated by CP/M utilities.

One useful application is to SAVE a program that has been placed in memory by the DDT utility. To produce the system prompt (A>) and allow entry of the SAVE command, you must exit from the DDT utility with a warm boot (performed by pressing CTRL-C).

Another common application is to SAVE a copy of the operating system onto a file. Part of the system always occupies the area of memory between locations 0H and 100H. However, if the system kernel is loaded into the Transient Program Area by the MVCPM207 utility, it can be SAVED onto a file. To do this, you must follow a MVCPM207 command with a command in the following form:

A>**SAVE 38 CPMkk.COM**CR

Where **kk** is the number of kilobytes of memory to which the system has been adjusted by the MVCPM207 utility.

# 4. COMPUTING PAGES TO BE SAVED

To SAVE a program from memory, you first determine how many pages this program occupies. This requires a conversion of the program's hexadecimal size to decimal pages.

The hexadecimal size of a program is displayed. when the program is loaded into the TPA under DDT. This display indicates that the program begins at address 0100H and ends at an address identified as "NEXT". You should take the two left-hand digits from the "NEXT" value, and convert them into a decimal number. This decimal number is the number of pages to be SAVEd.

For example, if you are debugging a program that occupies memory locations 0100H through 28FFH, the results of this debugging activity can be recorded by taking the hexadecimal value "28", converting them into the decimal value "40", and entering the following command:

```
A>SAVE 40 PROGRAMX.HEX<CR>
```

If the disk to which you try to SAVE a data block does not have enough space to record a file of the specified size, then the following error message will be displayed:

```
NO SPACE
```

# STAT

*The Utility that Reports Disk Statistics and Assigns Status*

The STAT utility has two functions: to provide statistical information about disk space or file size (1), and to allow you to change the assignment of various types of status (2) to files and devices.

## 1. REPORTING STATISTICS CONCERNING DISKS AND FILES

The STAT utility enables you to determine the amount of disk space that specific files occupy and/or the amount of disk space not yet occupied by files. STAT reports these statistics in a variety of units.

## 1.1 Available Disk Space

The amount of unoccupied space on the disk in the default drive that is available for use can be determined by entering the command "STAT" and a carriage return at the default drive prompt, as in the following example:

```
A>STAT<CR>
```

If drive A is the only drive that has been logged since the system was booted, then STAT will respond with a display like the following:

```
A: R/W, Space: 3k
```

Where "A:" indicates that the disk in drive A was investigated by STAT;

where "R/W," stands for Read/Write, meaning that data and files can be written to or read from the disk. If STAT encounters a disk that has been write-protected, it will report that the disk is "R/O" or read/only. Data and files cannot be recorded to or erased from a read/only diskette; and

where "3k" indicates the amount of space that is unoccupied on the disk, and thus available to record files. Memory space is expressed in units of 1024 bytes, or kilobytes (k). One byte holds one character. The disk that was investigated for the preceding example has enough remaining memory space to store 3072 characters.

If you logged other drives since the system was last booted, the preceding response to the simple "STAT" command line will also produce a listing for the disks in each of the drives that have been logged, as shown:

```
A: R/W, Space: 3k
B: R/W, Space:18k
C: R/W, Space:65k
```

Even if a non-default drive has not yet been logged, you can still ascertain the space remaining on the disk within it by specifying the drive in the STAT command line:

```
A>STAT B:<CR>
```

The entry of this command line might produce the response:

```
Bytes Remaining on B: 24k
```

## 1.2 Disk Space Consumed by a File

To find out how much space one particular file occupies, requires specification of the file name and the drive in which the file resides. An appropriate entry for such a command would be:

A>**STAT C:PRINTOUT.DOC**CR

Such an entry might produce the STAT message:

```
 Recs  Bytes  Ext Acc
  140    18k    2 R/W C:PRINTOUT.DOC
Bytes Remaining on C: 72k
```

Where "Recs" stands for the number of records within the file. The file PRINTOUT.DOC contains 140 data records. A record is the amount of data it takes to fill up 128 bytes (one-eighth of a kilobyte) of memory space. Thus, one record consists of 128 data characters. However, STAT cannot report fractions of records; and if a file contains anywhere from 1 to 128 characters, STAT will still report that the file contains one record;

where "Bytes" stands for the amount of memory space the file occupies, expressed in kilobytes (1024 bytes). In this display, STAT reports that the file PRINTOUT.DOC occupies 18 kilobytes of memory space. One kilobyte of space will hold 8 records. However, STAT cannot report fractions of kilobytes; and if a file contains anywhere from 1 to 8 records, STAT will still report that it occupies one kilobyte of space;

where "Ext" stands for the number of extents that the file occupies. In this display, STAT reports that the file PRINTOUT.DOC occupies two extents. One extent will process 16 kilobytes of data. However, STAT cannot report fractions of extents; and if a file occupies anywhere from 0 to 16 kilobytes, STAT will still report that it requires one extent to be processed;

where "Acc" indicates the file's access status. The "R/W" access status (read/write) enables the user to run commands to freely erase data from the file, or add data to the file. The "R/O" access status imposes limitations on the commands the user can run to change the data in a file. (See Page 2-180.); and

where "PRINTOUT.DOC" is the name of the file that is being statistically analyzed. Since this file name is **not** enclosed in parentheses, it has directory status. If a file name in this kind of display is enclosed in parentheses, it has system status. (See Page 2-181.)

## 1.3 Disk Space Consumed by Groups of Files

To obtain the status of all of the files on a disk without entering a command line specifying each file on the disk, the drive letter and the ambiguous filename "*.*" can be entered, as in the following command line:

A>**STAT B:*.***↵

Such an entry will produce a display like this:

```
 Recs  Bytes  Ext Acc
  175    22k    2 R/W B:DATE0927.DOC
  160    20K    2 R/W B:DATE1003.DOC
   15     2k    1 R/W B:REPORT1.TXT
    9     1K    1 R/W B:REPORT2.FIL
  119    15k    1 R/O B:THATFILE.COM
  201    25k    2 R/O B:THISFILE.COM
Bytes Remaining On B: 26k
```

(Such an operation will only reveal the files within the currently logged user area.) STAT can also convey the status of groups of files on a disk when ambiguous file names are entered with the asterisk and/or question mark representing only part of the file names, as in the following sample command lines:

A>**STAT B:*.COM**↵

A>**STAT B:DATE????.DOC**↵

A>**STAT B:REPORT?.***↵

## 1.4 Measuring the Size of a File

In addition to determining the actual amount of disk space occupied by a file, STAT can also report on the outermost boundaries of the disk space occupied by a file, or the difference between the first and last sectors occupied by the file.

In the case of sequential files, the size of the file is the same as the number of records because records are recorded on adjacent sectors. The records of random files are often scattered over a wider disk area than they actually occupy. To determine the size of a file, the normal command line for status of a particular file is entered, followed by a space and the characters "$" and "S", as shown:

```
A >STAT C:PRINTOUT.DOC $S<CR>
```

The preceding entry might produce the response:

```
 Size  Recs  Bytes  Ext Acc
  140   140    18k    1 R/W C:PRINTOUT.DOC
Bytes Remaining On C: 72k
```

NOTE: If disks are switched from one drive to another between STAT runs, the individual file statistics will be accurate, but the "Bytes Remaining on X: nnk" message will bear a kilobyte count that reflects the remaining disk space on the disk that was first logged within that particular drive since boot-up. To ensure a correct listing of remaining disk capacity, you should perform a warm boot (enter **CTRL-C**) if disks are switched between STAT runs.

In this example, the value listed for "Size" and for "Recs" are equal because the file being statistically analyzed is a sequential file.

# 2. ASSIGNING STATUS

The STAT utility also performs functions which allow disk protection, manipulation of file indicators, disk parameter statistics, user number display, and Input/Output device assignment. A list of these special STAT functions can be invoked by entering the following command:

A>**STAT VAL:** 

In response, STAT will display the following list:

```
Temp R/O Disk: d:=R/O                                    (2.1)
Set Indicator: d:filename.typ $R/O $R/W $SYS $DIR  (2.2, 2.3)
Disk Status  : DSK: d:DSK:                               (2.4)
User Status  : USR:                                      (2.5)
Iobyte Assign:                                           (2.6)
CON: = TTY: CRT: BAT: UC1:
RDR: = TTY: PTR: UR1: UR2:
PUN: = TTY: PTP: UP1: UP2:
LST: = TTY: CRT: LPT: UL1:
```

## 2.1 Temporarily Changing a Disk's Access Mode

The category title "Acc" in a STAT display stands for the access mode of a file. A file's access mode can be either R/W or R/O. The R/W (Read/Write) mode allows the user to write data to the file or erase data from it. The R/O (Read/Only) mode prevents insertion or deletion of data within the file.

To change the access mode of all files on a disk from R/W to R/O, requires the entry of "STAT", a space, the name of the drive containing the disk to be protected, a colon, an "=" sign, the new access mode "R/O", and a carriage return. The following example illustrates such a command line:

A>**STAT B:=R/O** 

After the entry of this command line, the data files on disk B cannot be erased or over-written. The protection furnished by the R/O access mode will remain in effect until a warm or cold boot is performed.

If you protect files in this manner and try to write to or delete from a file on the disk, the following error message is displayed:

```
Bdos Err On B: R/O
```

CP/M will not allow alteration to file contents, and will perform a warm boot with the next key pressed after this message is displayed. After this warm boot is performed, the disk's files are reset to the R/W access mode.

## 2.2 Changing a File's Access Mode

The R/W (Read/Write) mode allows you to write data to the file or erase data from it. The R/O (Read/Only) mode prevents insertion or deletion of file data, and it can be applied to files by the set indicator commands "$R/O" and "$R/W".

Individual files, or groups of files, can be given the R/O status with the entry of "STAT", a space, the drive specification, the file name, a space, the characters "$R/O", and a carriage return. For instance, the file named "ARCHIVAL.DOC" on the disk in drive B can be protected from alteration by the following entry:

```
A>STAT B:ARCHIVAL.DOC $R/O<CR>
```

After this command has been given, the following message is displayed:

```
ARCHIVAL.DOC set to R/O
```

Attempts to write to or erase from this file using any command will produce the message:

```
Bdos Err On B: File R/O
```

If this error message occurs, you can access the system by pressing any key.

The R/O attribute is recorded in the directory and will not change if the system is re-booted. The file can be made Read/Write again by entering a similar command with the characters "$R/W" in place of "$R/O", as in the following example:

```
A>STAT B:ARCHIVAL.DOC $R/W<CR>
```

## 2.3 Changing a File's System Status

Individual files, or groups of files, can be given the system status, which prevents their mention in DIRectory listings, prevents access by the EDitor, and prevents copying by PIP (except with the "[R]" parameter). This status is applied to a file when the entry of "STAT", the drive specification, and the file name is followed by the characters "$SYS". For instance, the file "HIDEFILE.COM" on drive B becomes ineligible for PIPing (except with [R], and DIRectory mention, EDiting, when the following command is entered:

```
A>STAT B:HIDEFILE.COM $SYS<CR>
```

After this command has been given, the message:

```
HIDEFILE.COM set to SYS
```

is displayed.

To illustrate how this status protects a file, efforts to edit the file using the ED utility will produce the message:

```
"SYSTEM" FILE NOT ACCESSIBLE
```

In addition, a check of this file using a different form of the STAT command will also produce noteworthy results, as shown. If you enter the command:

```
A>STAT B:HIDEFILE.COM<CR>
```

The following display will appear:

```
 Recs  Bytes  Ext  Acc
   38     5k    1  R/W B:(HIDEFILE.COM)
Bytes Remaining On B: 19k
```

When a file bears the SYS status, it will have parentheses around its listing in a STAT file report.

The system attribute can be removed from this file by the following entry:

```
A>STAT B:HIDEFILE.COM $DIR<CR>
```

The **$DIR** argument in a STAT command line lifts all of the restrictions imposed by the $SYS argument in a previous STAT command.

## 2.4 Listing Disk Characteristics

Characteristics of a disk can be obtained by entering "STAT", a drive specification, "DSK:", and a carriage return, as shown:

```
A>STAT B:DSK:<CR>
```

STAT will list a display similar to the following:

```
   B: Drive Characteristics
2496: 128 Byte Record Capacity
 312: Kilobyte Drive Capacity
 256: 32 Byte Directory Entries
 256: Checked Directory Entries
 128: Records/ Extent
  16: Records/ Block
  32: Sectors/ Track
   2: Reserved Tracks

A>
```

Where the letter "B" to the left of "Drive Characteristics" indicates the selected drive, which contains the disk being analyzed (a 5.25-inch soft-sectored disk in this example);

where the second and third lines indicate the disk's total "Record Capacity" and "Kilobyte Drive Capacity", respectively. These totals include file, directory, and system capacity. The amount of space accessible for storing files is a few kilobytes smaller.

where the directory can hold a maximum of "256" entries;

where the number of "Checked Directory Entries" is usually identical to the directory size for floppy disks, since this mechanism is used to detect changed disk media during CP/M operation without an intervening warm boot;

where the number of "Records/ Extent" indicates the addressing capacity of each directory entry;

where the number of "Records/ Block" indicates the basic size of a block. By knowing that a record contains 128 bytes, you can calculate the number of bytes in a block;

where the number of "Sectors/ Track" can help you determine the number of sectors on the disk (the 5.25-inch soft-sectored disk has 40 tracks on each formatted side); and

where the "Reserved Tracks" are reserved for the operating system and the disk file directory.

If no drive is specified in the command line, then a listing in the above form will be displayed for the disk in each drive that has been logged since the last warm boot.

## 2.5 Examining User Areas

User access to certain files can be controlled by assigning user areas to the files. (See text on USER resident command.) The following form of the STAT utility will display a list of the numbered user areas which have files on the currently logged disk.

```
A>STAT USR:<CR>
```

This entry might produce the response:

```
Active User : 0
Active Files: 0 1 3
```

The first line in the response lists the currently logged user area, or "Active User", which was set at zero either by the last USER command, or by a cold boot.

The second line lists the other user area numbers that have been established for files on the disk. This list is displayed as "Active Files", and indicates that additional files can be accessed by logging user area one or three.

## 2.6 Temporarily Matching Logical and Physical Devices

The STAT utility also enables you to monitor and temporarily control the assignment of physical Input/Output devices to the appropriate logical device names. In general, there are four logical peripheral devices which, at any particular instant, are each assigned to one of several physical peripheral devices. The four logical devices, and the physical devices that can be assigned to each of them, are shown in the display invoked by the **STAT VAL:** command, and in the following table:

| LOGICAL DEVICE NAME | PHYSICAL DEVICE NAME | DESCRIPTION AND/OR CATALOG NAME OF RECOMMENDED INPUT/OUTPUT MACHINE |
|---|---|---|
| CON: | TTY: | A printing terminal attached to serial port outlet A on the Z-100 (e.g. Decwriter). |
| | CRT: | A video display terminal and keyboard. |
| | BAT: | A batch pseudo-device using RDR: for input and LST: for output |
| | UC1: | A modem attached to serial port B on the Z-100. |
| RDR: | TTY: | A printing terminal attached to serial port outlet A on the Z-100 (e.g. Decwriter). |
| | PTR: | Not implemented. |
| | UR1: | A modem attached to serial port B on the Z-100. |
| | UR2: | A video display terminal and keyboard. |
| PUN: | TTY: | A serial printer attached to serial port outlet A on the Z-100. |
| | PTP: | Not implemented. |
| | UP1: | A modem attached to serial port B on the Z-100. |
| | UP2: | A video display terminal and keyboard. |
| LST: | TTY: | A serial printer attached to serial port outlet A on the Z-100 (H/Z-25, H-14, TI-810, Diablo, Epson MX-80 serial, Decwriter). |
| | CRT: | A video display terminal and keyboard. |
| | LPT: | A parallel printer attached to the parallel printer port outlet on the Z-100 (Epson MX-80 parallel, Centronics). |
| | UL1: | A modem attached to serial port B on the Z-100. |

**Table 2-4**
Possible Device Assignments

## EXAMINATION OF CURRENT DEVICE ASSIGNMENTS

The current assignments of the physical devices to logical devices are shown in the display invoked by the following command entry:

A>**STAT DEV:**↵

This command produces a display similar in form to:

```
CON: is CRT:
RDR: is UR1:
PUN: is UP1:
LST: is UL1:
```

## CHANGING DEVICE ASSIGNMENTS

The current logical to physical device assignment can be changed by entering this STAT command:

A>**STAT {logical device}:={physical device}:**↵

Where both the {**logical device**} and the {**physical device**} are specified by their four-character names.

For example, the following entry will change the physical device assigned to the list logical device (LST:) to a parallel printer (LPT:):

A>**STAT LST:=LPT:**↵

Several device assignments can be affected in one command by entering equations in series, separated by commas.

Physical to logical device pairings assigned by the STAT utility will survive a warm boot, but are replaced by default assignments when a cold boot is performed. Default assignments, that will survive a cold boot, may be changed through the CONFIGUR utility.

# 3. STAT ERROR MESSAGES

```
Bad Delimiter
```

EXPLANATION: You entered a command line with improper syntax while assigning some type of status (2). Command should be reentered in the proper form.

```
Invalid Assignment
```

EXPLANATION: You tried to make a device assignment with a device name that is not listed in Table 2-4. Reenter assignment using valid device names.

```
Invalid File Indicator
```

EXPLANATION: You tried to assign a particular indicator ($R/O, $R/W, $SYS, or $DIR) to a file but did not type the proper characters after the dollar sign. Check the invalid command for improper characters (see Page 2-179) and reenter the command.

```
File Not Found
```

EXPLANATION: STAT found none of the specified files on the logged disk. Reenter command specifying different file names, or perform a warm boot and insert a disk that has the specified file(s).

```
Invalid Disk Assignment
```

EXPLANATION: You entered a disk status assignment command that included a status other than "R/O" or "R/W". See Page 2-179 for the correct syntax of disk status command lines, and reenter the command.

```
** Aborted **
```

EXPLANATION: When you assign status ($R/O, $R/W, $SYS, or $DIR) to a file, you must wait for the system prompt to be displayed before you type the next command. If you try to enter a command while a STAT status command is executing (utilizing CP/M's "type-ahead" keyboard entry buffer), you will abort the status command.

# SUBMIT

*The Utility that Triggers Automatic Execution of CP/M Commands*

The SUBMIT utility enables you to initiate automatic execution of several sequential commands (3) by issuing only one SUBMIT command (2). The SUBMIT command accesses a file containing a pre-composed sequence of commands (1). You can also vary the execution of this command sequence with each invocation, by substituting command line parameters such as file names, drive names, device names for variables inserted into the command file.

## 1. SUB FILES

A SUB file is a file with the "SUB" extension, which is comprised of a sequence of command lines composed using a text editor (e.g. ED). The SUB file is the source of the command sequence referenced by the SUBMIT command. The command lines within a SUB file can contain prototype parameters to represent command line fields. These prototype parameters appear in the SUB file in the form:

```
$n
```

Where each numeral ("n") preceded by a dollar sign ("$") takes the place of a file name, drive name, or device name which you will substitute into the SUB file with an actual parameter entered in the SUBMIT command line.

The following example shows the text of the SUB file with the name "MULTIJOB.SUB", which contains two prototype parameters:

```
ASM $1
DIR $1
ERA *.BAK
PIP $2:=$1.PRN
ERA $1.PRN
```

## 2. SUBMIT COMMAND LINES

The SUBMIT utility can be used as in the following example:

A>**SUBMIT MULTIJOB PROGRAMX LST**CR

Where {**MULTIJOB**} is the name of the SUB file filled with batch commands,

where {**PROGRAMX**} and {**LST**} are parameters which will be substituted into one or more locations in the batch file.

An actual SUBMIT parameter is the explicit specification of a command, file, drive, or device which you want to substitute for a prototype parameter within the SUB file. The number of parameters specified in a SUBMIT command line must correspond to the number of different prototype parameters that appear within the SUB file in the form "$n".

## 3. SUBMIT EXECUTION

After the entry of a SUBMIT command line, SUBMIT will create a file named $$$.SUB, and send it to the Console Command Processor (CCP) portion of the operating system. The CCP recognizes the commands in the SUB file and executes them in sequence. The $$$.SUB file contains the actual parameters expressed in the SUBMIT command substituted for the corresponding prototype parameters which are imbedded in the original SUB file. If created in response to the preceding example command line, the $$$.SUB file would contain the following commands:

```
ASM PROGRAMX
DIR PROGRAMX.*
ERA *.BAK
PIP LST:=PROGRAMX.PRN
ERA PROGRAMX.PRN
```

# 4. SUBMITTING A SUB FILE FROM A DRIVE OTHER THAN A

The $$$.SUB file must exist on a disk in drive A for the commands to be executed. However, the SUB file used by SUBMIT to create the $$$.SUB file can reside on a disk in **any** existing drive; and the $$$.SUB file can be created on a disk in **any** existing drive.

If $$$.SUB is located on a disk in a drive other than drive A, the user must adhere to one of the following sets of stipulations to initiate command execution:

- The drive location of the disk containing the SUB file must be specified in the SUBMIT command line. $$$.SUB is created on a bootable disk, the disk is placed in drive A, and the system must be re-booted with drive A as the default drive.

- The drive location of the disk containing the SUB file must be specified in the SUBMIT command line. $$$.SUB is created on any disk and the PIP utility is used to transfer it to a disk that resides in drive A with sufficient space available to receive $$$.SUB.

# 5. ABORTING SUBMIT COMMAND EXECUTION

Execution of SUBMITted commands can be aborted by typing a DELETE, RETURN, or CTRL-C immediately after the command is displayed. In this case, the $$$.SUB file is removed, and the subsequent commands come from the console. Command processing is also aborted if the CCP detects an error in any of the commands. Programs which execute under CP/M can abort processing of command files when error conditions occur by simply erasing any existing $$$.SUB file.

# 6. UNCONVENTIONAL TEXT IN A SUB FILE

You can insert comments into a SUB file so that they will be displayed on the console with the SUB file commands during the processing of these commands. To ensure that the CCP ignores the comments and does not interpret them as commands, they must be entered on a line which begins with a semicolon (;).

To introduce literal dollar signs into a SUB file, you may enter "$$", which translates to a single "$" within the SUB file. Furthermore, a caret symbol (^) can precede the alphabetic character X, which produces a single CTRL-X character within the file.

# 7. CHAINED BATCH COMMANDS

The last command in a SUB file can be a SUBMIT command, which could either initiate execution of the commands within another SUB file, or repeat execution of the commands within the same file. You can include different parameters to be substituted for SUB file prototypes to cause varied repetition of the same set of commands.

# 8. SUBMIT ERROR MESSAGES

```
No 'SUB' File Present
```

EXPLANATION: SUBMIT command can only be issued if a file with the "SUB" extension exists on a logged disk, and if that file is specified in the SUBMIT command line. You should reenter the command with a different argument, or log the disk with the appropriate SUB file.

```
Disk Write Error
```

EXPLANATION: SUBMIT found insufficient disk space while trying to create $$$.SUB file on the disk. You should clear off some space on the disk or use another disk.

```
Error On Line n

Parameter Error
```

EXPLANATION: You entered an invalid parameter in the SUB file. Edit SUB file, with only decimal integers following the "$" sign for a parameter.

```
Directory Full
```

EXPLANATION: SUBMIT tried to create SUB file on a disk that didn't have enough directory space to accommodate the SUB file name. You should clear off some space on the disk or use another disk.

```
Cannot Close, Read/Only?
```

EXPLANATION: SUBMIT is trying to write data to a disk (as one of the activities prescribed in a SUB file command line), but the file being written to has the R/O (read/only) status. You must use the STAT utility to change the file's status to "R/W" (read/write); or change the SUB file command line that is trying to write to the file; or enter different SUBMIT parameters.

# SYSGEN

*The Utility that Puts the Operating System on a Disk*

The SYSGEN utility is used to transfer the operating system to a disk. Under some circumstances, SYSGEN does this task without the aid of other utilities. Sometimes SYSGEN needs the help of other utilities to put an entire, usable system on the disk.

## 1. SYSGEN INVOCATION

No matter which SYSGEN method is to be performed, the SYSGEN utility is invoked by entering the following command at the system prompt:

```
A>SYSGEN<CR>
```

A display in the following form will appear:

```
CP/M-85 SYSGEN VER 2.2.100

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

Your next entry depends on the SYSGEN method you are using.

# 2. SYSGEN METHODS

You must consider the circumstances before running SYSGEN, so that you use the appropriate SYSGEN method. Use of the MVCPM207 utility will often influence your choice of a SYSGEN method. Thus, you have a choice of the following two SYSGEN methods:

- If the MVCPM207 utility was **not** used immediately before SYSGEN to customize the system for memory capacity, then use the "Disk-to-Disk Method" (2.1). (The BSYSGEN utility could also be used under these circumstances.)
- If the MVCPM207 utility (which moves a system kernel into computer memory) was used immediately before SYSGEN, use the "Computer-to-Disk Method" (2.2).

NOTE: When the CP/M Operating System is copied to a disk, it is moved in two parts: the system kernel and the BIOS files. The "Computer-to-Disk System Copying" method does not copy the BIOS files. If this method is used, then the PIP utility must be used to copy the BIOS files.

## 2.1 Disk-to-Disk Method

If the operating system is being copied between two disks of the same type, you can copy both the system kernel and the BIOS files to the destination disk using this SYSGEN method. (This method is used when a MVCPM207 activity does **not** precede the SYSGEN activity.)

Under these circumstances, you should answer the "SOURCE DRIVE NAME" prompt by typing the letter for a drive name, as shown:

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

Where **x** is the letter of the source drive.

SYSGEN will respond with a prompt in the following form:

```
SOURCE ON x, THEN TYPE RETURN
```

Answer this prompt with a carriage return. SYSGEN will read the system kernel from the source disk, and signal that it has done so with the following message:

```
FUNCTION COMPLETE
```

Then SYSGEN will offer the option of copying the BIOS files (BIOS88.SYS and BIOS85.SYS) from the source disk with the prompt:

```
COPY BIOS88.SYS & BIOS85.SYS (Y/N):
```

If you wish to copy the BIOS files from the source disk, then **Y** should be pressed, and SYSGEN will again display the message:

```
FUNCTION COMPLETE
```

If you do **not** wish to copy the BIOS files ("BIOS85.SYS" and "BIOS88.SYS") with SYSGEN, press **N** at this prompt.

After either entry, SYSGEN will prompt for the drive that contains the destination disk. The user should answer the prompt as shown:

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

Where **y** is the letter of a valid, working drive.

Then SYSGEN will display the prompt:

```
DESTINATION ON y, THEN TYPE RETURN
```

Enter a carriage return at this prompt, to confirm the destination drive choice. SYSGEN will put the system kernel, and in some cases the BIOS file, onto the destination disk. (The disk in drive "y".)

Then SYSGEN will display the prompt:

```
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT):
```

If you wish to copy the same system components to a different destination disk, then type the letter of the drive containing this disk.

If you do not wish to SYSGEN other disks, then enter a carriage return. The SYSGEN activity will end, and CP/M will display the system prompt.

## 2.2 Computer-to-Disk-Method

If you have just run the MVCPM207 utility to customize the operating system for memory capacity, then a system kernel still resides in a special location of computer memory. You can copy this system kernel from the computer to the disk by using this SYSGEN method.

When SYSGEN prompts for "SOURCE DRIVE NAME", you must enter a carriage return — **not** a drive name.

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

SYSGEN will now prompt you for the drive that contains the destination disk. Answer the prompt as shown:

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

Where **y** is the letter of the drive containing the disk that is to receive the system kernel. (The BIOS files cannot be copied through this SYSGEN method.)

SYSGEN will request confirmation with the prompt:

```
DESTINATION ON y, THEN TYPE RETURN
```

Enter a carriage return at this prompt. SYSGEN will put the system core onto the destination disk (the disk in drive y).

Then SYSGEN will again display the prompt:

```
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT):
```

If you wish to copy the same system components to a different destination disk, then the letter of the drive containing this disk should be entered.

If you do not wish to SYSGEN other disks, and you have **not** specified drive A at a previous "DESTINATION DRIVE" prompt, enter a carriage return at this "DESTINATION DRIVE" prompt. The SYSGEN activity will end, and CP/M will display the system prompt.

If you do not wish to SYSGEN other disks, and you specified drive A at a previous "DESTINATION DRIVE" prompt, then you must reset the computer at the second "DESTINATION DRIVE" prompt rather than entering a carriage return. Resetting the computer at this prompt is necessary because entry of a carriage return at this prompt induces a warm boot, which would cause the new system kernel (recently recorded on the disk in drive A) to be loaded into computer memory. It is undesirable to load any part of this new system kernel into memory on a warm boot because it might have just been changed by the MVCPM207 utility. Therefore, it might be of a different size than the system kernel that was loaded into memory at bootstrap.

If you are copying the CP/M system from memory to a destination disk (as you would after using MVCPM207), SYSGEN will not be able to copy the BIOS files for you. Since the BIOS files ("BIOS85.SYS" and BIOS88.SYS") are essential to make a disk bootable, you must copy them using the PIP utility.

The PIP utility can be used to transfer the BIOS files, as shown:

A>**PIP y:=x:BIOS8?.SYS[RV]**CR

Where **y** is the destination disk receiving copies of the BIOS files;

where **x** is the source disk from which the BIOS files are copies;

where **BIOS8?.SYS** is a wild card file name that stands for both BIOS85.SYS and BIOS88.SYS; and

where **[RV]** are PIP parameters used to help you copy a file that has system status, and to help you verify the accuracy of the copy operation.

# 3. SYSGEN ERROR MESSAGES

```
INVALID DRIVE NAME
```

EXPLANATION: When you specify drive names you must use the names of drives that exist in the hardware environment, and are recognized by the operating system that was loaded at bootstrap.

```
NO SOURCE FILE ON DISK
```

EXPLANATION: The drive specified as "SOURCE DRIVE" did not contain the BIOS files. Use a different disk in the source drive, or rename BIOS files that had been given names other than "BIOS85.SYS" and "BIOS88.SYS".

```
SOURCE FILE INCOMPLETE
```

EXPLANATION: SYSGEN failed in an attempt to copy t BIOS files from the disk in the source drive. This file might have been damaged by disk media flaws or partially overwritten. Reset, perform bootstrap, and reenter the SYSGEN command using a different disk in the source drive.

```
WRITE ERROR DURING BIOSxx.SYS
```

EXPLANATION: Run SYSGEN again with a destination disk that is write-enabled, is formatted, and has at least 6 kilobytes of free space.

```
ERROR READING BIOSxx.SYS
```

EXPLANATION: SYSGEN failed in an attempt to copy the BIOS85.SYS file and/or the BIOS88.SYS file from the disk in the source drive. These files might have been damaged by disk media flaws or partially overwritten. Reset, perform bootstrap, and reenter the SYSGEN command using a different disk in the source drive or using a different disk to perform bootstrap.

```
PERMANENT ERROR, TYPE RETURN TO IGNORE
```

EXPLANATION: The system kernel or BIOS files are either incompatible with the destination disk type or otherwise flawed. Reset, perform bootstrap, and reenter the SYSGEN command using a different disk in the source drive or using a different disk to perform bootstrap. Under some circumstances, the user must use the MVCPM207 utility before SYSGEN.

```
UNABLE TO SELECT DRIVE
```

EXPLANATION: You entered the name of a drive that is not accessible. Rerun the utility entering valid drive names at the prompts.

# TYPE

*The Resident Command that Displays File Contents on the Console*

The TYPE command displays the contents of specified files on the console (1). Fast scrolling displays can be controlled to make them readable (2). TYPE is best used on ASCII files (3).

## 1. DISPLAYING A FILE

To produce a console display of the contents of a disk file, respond to the system prompt with a command line in the following form:

A>**TYPE {file name}**CR

Where {**file name**} is the complete, explicit name of a file.

If you must examine a file that resides on a disk in a non-default drive, this drive must be specified immediately before the file name specification. For example, if the file "LOGDRIVE.TXT" resides on a disk in drive B, its contents will be displayed on the screen when the following command is entered:

A>**TYPE B:LOGDRIVE.TXT**CR

You cannot specify wildcard file names in a TYPE command.

## 2. SCREEN DISPLAY CHARACTERISTICS

The display produced by TYPE will scroll by on the console until the entire contents of the file have been displayed. You can terminate the display during its execution and return to the operating system by pressing any character other than "CTRL-S". The display scroll can be halted temporarily by pressing the "CTRL-S" character, and re-started by pressing "CTRL-S" (or any other character) again.

Some of the special features (bold-faced type, underlining, etc.) inserted into text files by some word processing systems might not be indicated in screen displays produced by the TYPE command.

## 3. PRACTICAL USAGE OF TYPE

The most desirable usage of the TYPE command is to display text files composed of ASCII characters (printable letters and numbers). If the TYPE command is issued for a file with a ".COM" extension (or for any file stored on the disk in binary form) the resulting screen display will contain a meaningless series of characters.

To obtain a listing of the file, enter a **CTRL-P** before entering the TYPE command line. Listings will not exhibit special features (bold-faced type, underlining, etc.) that are inserted into some text files by word processing systems. Enter CTRL-P a second time to end the listing.

# USER

*The Resident Command that Controls User Access to Disk Areas*

The USER command enables you to access only files in a specified area of the disk directory (1). You can log into USER areas by invoking a single USER command line (2). You must implement a sequence of commands to put files into a non-zero USER area (3).

## 1. USER INVOCATION

File directories supported by the CP/M system are divided into 16 user areas. When a user is logged into a particular USER area, then only the files within that area are accessible (except by implementing the PIP utility with the "G" parameter).

The USER command is useful when several users have files stored on the same disk. File directory (DIR) and status checks (STAT) invoked by a particular user will list only the files in the user's specified area.

Whenever a cold boot is performed, the user is automatically logged-in to USER area number 0. When in this area, only files in USER area 0 of the directory are accessible.

## 2. INVOKING USER

To make a user area other than number 0 accessible, you should respond to the system prompt by entering a command line in this form:

A **USER {area number}**↵

Where {area number} indicates the number, from 0 to 15, of an area that contains files that you wish to access.

The USER command can be issued whenever the system prompt appears. The currently-logged USER area will be in effect for all of the disks in the hardware environment, until a different USER area is logged or a cold boot is performed.

## 3. PULLING FILES INTO USER AREAS

Files can be pulled into a USER area that contains the file PIP.COM, but to put the file PIP.COM into that USER area in the first place requires a special process.

To put the file PIP.COM into a USER area (other than area zero), you should follow these steps, in sequence:

3.1 Insert disks that have the PIP.COM utility and at least 8 kilobytes of space available.

3.2 Log USER area zero (or whatever USER area contains PIP.COM) by entering the command **USER 0** ↵.

3.3 Invoke the PIP utility by entering the command **PIP** ↵. (Specify the name of the drive that contains PIP.COM if it is not the default drive.)

3.4 Press ↵ at the asterisk (*) prompt.

3.5 Log USER area by entering the command **USER n** ↵, where **n** is the number (0-15) of the USER area you wish to establish.

3.6 Save the portion of computer memory that now contains the file PIP.COM by entering the command **SAVE 29 PIP.COM.** ↵. (Specify the name of the drive on which you wish to establish the USER area if it is not the default drive.)

These six steps will place the file "PIP.COM" into the desired USER area (n). Once PIP.COM resides within a USER area, it can help you pull other files into that USER area. The following example display demonstrates how this procedure might appear on the console:

(3.2) A>**USER 0**CR
(3.3) A>**PIP**CR
(3.4) * CR
(3.5) A>**USER 5**CR
(3.6) A>**SAVE 29 PIP.COM**CR

If you have followed the preceding example procedure, additional file copies can now be pulled into USER area 5 by the entry of commands in the form:

A>**PIP B:={drive}:{filename.ext}[G{source area}]**CR

Where {**drive**} is the name of the drive from which file copies are being pulled;

where {**filename.ext**} is the name of the file being pulled into the new USER area from the {**source area**}; and

where {**source area**} is the number of the USER area from which files are being pulled. This area must contain {**filename.ext**}.

# XSUB

*The Utility that Batches Commands Within Utility Programs for Automatic Processing*

The XSUB utility extends the power of the SUBMIT utility to enable you to invoke utility programs, and then execute commands within these programs, while making only one entry at the terminal.

## 1. XSUB OPERATION

The XSUB command line is entered as the first line in a SUB file. A SUB file is a text file composed of command lines that are executed in sequence when a SUBMIT command line is entered at the system prompt.

When the XSUB command is executed, it moves beneath the component of the operating system known as the Console Command Processor (CCP). All command lines after the XSUB command line are processed by XSUB, so that programs which usually prompt you to make entries on the keyboard will accept entries directly from the SUB file.

The XSUB program remains in memory after execution, and displays the message `"(xsub active)"` each time a warm boot is performed during the execution of the $$$.SUB file. Thus the initial XSUB command in a SUB file can remain in effect throughout the execution of every command in the SUB file.

XSUB will display the message "Xsub Already Present" as you enter the command **XSUB** at the system prompt (through the console). However, this remnant of the XSUB program will not go into effect when subsequent SUB files are submitted. Therefore, you must make "XSUB" the first line of any SUB file in which XSUB execution is desired.

NOTE: XSUB will not support the submission of input for programs which, when run, prompt for single character input. The only programs that can effectively be used in a SUB file under XSUB are programs that accept input lines ended with a carriage return, such as DDT, PIP, LIST, and ED.

## 2. XSUB EXAMPLE

The file "MEGAJOB.SUB" contains the following command lines:

```
XSUB
DDT
I$1.HEX
R
G0
SAVE 4 $2.COM
```

Processing of this SUB file could be initiated by entering the following command line:

```
A>SUBMIT MEGAJOB PRGRMY PRGRMZ<CR>
```

The preceding entry creates a $$$.SUB file in which the primary file name "PRGRMY" is substituted for the prototype name "$1" of the SUB file, and "PRGRMZ" is substituted for the "$2" prototype parameter.

As commands are read from the $$$.SUB file, the XSUB utility enters computer memory. Then DDT enters and executes the DDT commands "IPRGRMY.HEX", "R", and "G0", which are also included in the $$$.SUB file. The DDT command "G0" has the same effect as a warm boot, which enables the operating system to process the final command in the $$$.SUB file, "SAVE 4 PRGRMZ.COM".

*Appendix A*

# OPERATING SYSTEM ERROR MESSAGES

This appendix explains error messages produced by the CP/M Operating System.

```
HARD (operation) ERROR ON DRIVE (d): STATUS (w) TRACK (x) SIDE (y) SECTOR (z)
```

Where "operation" is either "READ" or "WRITE", depending on which operation was being performed through CP/M when the error occurred;

where "d" is the letter of the drive that contains the disk on which the error occurred (A, B, C, or D);

where "w" is the number of the status of the error. Status numbers are explained in your Z-100 hardware manual;

where "x" is the number of the disk track on which the error occurred;

where "y" is the side of the disk on which the error occurred; and

where "z" is the number of the disk sector on which the error occurred;

If it is possible to skip over the disk location that could not be read from or written to, CP/M will follow the "HARD" error message with the following prompt:

```
PRESS <CTL-C> TO ABORT, <RETURN> TO IGNORE:
```

**Causes:** If the message says "READ", the CP/M system (or some operation conducted through CP/M) tried to read from a disk and failed. If the message says "WRITE", the CP/M system (or some operation conducted through CP/M) tried to write to a disk and failed. This failure might have occurred because: 1) Disk used for a data transfer operation is damaged or extremely worn. 2) Disk drive controller is malfunctioning. 3) Disk being accessed was FORMATted by a disk drive controller other than the Z207 controller. 4) Disk being accessed has not been FORMATted at all. 5) Drive being accessed contains no disk. 6) Accessed disk is write-protected.

**Remedies:** If no prompt is displayed (as during an attempted FORMAT activity), the system will automatically abort the operation after briefly displaying the "HARD" error message. If the prompt is displayed, you can either abort the operation by entering a **CTRL-C** (performs a warm boot), or ignore the error by entering a ⓒⓡ (skips over the sector where the error occurred). However, after making one of these two entries, you should 1) Inspect or replace disk. 2) Inspect or replace controller card. 3) & 4) FORMAT disks in your own hardware environment. 5) Access a drive with a disk. 6) Write-enable disk and perform warm boot before trying to write data to disk.

```
Bdos Err On x: Select
```

**Cause:** The variable "x:" is the name of a drive which the user referred to in a command, but which CP/M cannot access or does not recognize.

**Remedy:** Press any keyboard character after such an error message, and the operation will be aborted as if you entered a warm boot. CP/M will then display a system prompt.

```
Bdos Err On x: R/O
```

**Causes:** (Where "x:" is the name of a drive at which CP/M refuses to read or write data. 1) User tried to write to a file that had been given "Read/Only" status by the STAT utility. 2) User tried to write to a disk that had been given temporary "Read/Only" status by the STAT utility. 3) User tried to switch disks in a drive and then write to the new disk.

**Remedies:** In all cases, pressing any keyboard character after such an error message will cause a warm boot. 1) Change file status to "Read/Write" using the STAT utility. 2) Perform a warm boot in any fashion (such as the **CTRL-C** entry) before trying to write data to the disk. 3) Perform a warm boot (with a **CTRL-C**) after switching disks, but before accessing a switched disk.

```
NO FILE
```

**Causes:** 1) User invoked DIR Resident Command for a directory of disk files, and the file(s) specified were not on the disk in the logged user area. 2) User invoked TYPE Resident Command to display contents of a file that was not on the disk in the logged user area. 3) User invoked ERA Resident Command to erase a file that was not on the disk in the logged user area. 4) User invoked REN Resident Command to try to rename a file that was not on the disk in the logged user area.

**Remedies:** 1) Unless such a message gives you adequate information, log to a different user area with the USER Resident Command before trying DIR command. 2), 3), & 4) Check disk directory using DIR Resident Command or STAT utility to obtain correct file name.

```
FILE EXISTS
```

**Cause:** User tried to rename a file (with REN Resident Command) using a name that another file on the same disk already has.

**Remedy:** Give a different name to the file you are renaming, or move the file that already has the name to a different disk or user area.

```
NO SPACE
```

**Cause:** User invoked SAVE Resident Command to store memory contents on a disk that didn't have enough room for the data from memory.

**Remedy:** Reenter SAVE command either by specifying the name of a drive containing a disk that has sufficient room for the file, or by specifying fewer pages of data.

```
SYNCHRONIZATION ERROR
```

**Cause:** User tried to invoke a copy of the MVCPM207 utility that did not bear the same serial number identification as the Operating System.

**Remedy:** Only use a MVCPM207 utility and system that reside on the same CP/M Distribution Disk (or on a backup copy of this disk).

```
LOAD ERROR
```

**Cause:** User tried to perform bootstrap with a disk that does not contain a usable copy of the BIOS85.SYS, BIOS88.SYS, and/or ALTCHAR.SYS files.

**Remedy:** User should copy properly customized BIOS files to the disk, rename current BIOS files to the names "BIOS85.SYS" and "BIOS88.SYS", and/or copy an alternate character font file to ALTCHAR.SYS.

*Appendix B*

# BOOTSTRAP

The Start-Up Procedure in Volume I: CP/M Introductory Guide provides instructions on performing automatic bootstrap with one of your drives.

However, the Z-100 microcomputer enables you to vary start-up procedures by switching the automatic bootstrap feature off, and entering manual bootstrap commands that can reference any of your disk drives. This appendix explains the start-up variations that are possible with the Z-100.

## THE AUTOMATIC/MANUAL BOOTSTRAP FEATURE

The method you use to boot your Z-100 computer is determined by the setting of a switch inside the computer. This switch is labelled "SW-101", and it is illustrated in Figure B-1 of this appendix. (See your Z-100 hardware manual for further details on this switch.) When your Z-100 is shipped from the factory, it is preset to automatically perform bootstrap with your Z-100's lefthand drive (if your Z-100 is a Low-Profile) or with your Z-100's upper drive (if your Z-100 is an All-in-One) as soon as you turn on your computer.

See following page also

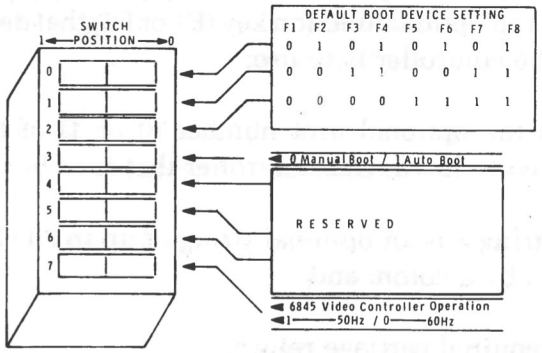



**Figure B-1**
Automatic/Manual Bootstrap Setting (Switch SW-101)

# Entering Manual Bootstrap Commands

When the SW-101 switch is set for automatic bootstrap, your Computer automatically boot up with your Z-100's lefthand drive (if your Z-100 is a Low-Profile) or with your Z-100's upper drive (if your Z-100 is an All-in-One). If you wish to boot up with a different drive, you must first set the SW-101 for manual bootstrap. Refer to your Z-100 hardware manual for instructions on setting this switch.

When the SW-101 switch is set for manual bootstrap, the console should display the pointed finger prompt and a flashing cursor when you turn on the Z-100 and enter **CTRL-RESET**, as shown:

You can enter your manual bootstrap command from the keyboard in response to the "pointing finger" prompt. There are several options that are available so that you can perform bootstrap from any of the drives in your hardware environment.

The bootstrap command syntax is:

**<B>**OOT **[<dev>][<#>][:<boot string>]**CR

Where **<B>** is required input that the computer completes with "OOT";
where **<dev>** is an optionl function key (F1 or F2) that determines which device the controller is to use;

where **<#>** is the optional unit number (0 or 1) of the device type connected to the device controller that is to be used;

where **<boot string>** is an optional string of up to 79 characters that is preceded by a colon; and

where CR is a required carriage return.

*Appendix B*

# BOOTSTRAP

The Start-Up Procedure in Volume I: CP/M Introductory Guide provides instructions on performing automatic bootstrap with one of your drives.

However, the Z-100 microcomputer enables you to vary start-up procedures by switching the automatic bootstrap feature off, and entering manual bootstrap commands that can reference any of your disk drives. This appendix explains the start-up variations that are possible with the Z-100.

## THE AUTOMATIC/MANUAL BOOTSTRAP FEATURE

The method you use to boot your Z-100 computer is determined by the setting of a switch inside the computer. This switch is labelled "SW-101", and it is illustrated in Figure B-1 of this appendix. (See your Z-100 hardware manual for further details on this switch.) When your Z-100 is shipped from the factory, it is preset to automatically perform bootstrap with your Z-100's lefthand drive (if your Z-100 is a Low-Profile) or with your Z-100's upper drive (if your Z-100 is an All-in-One) as soon as you turn on your computer.

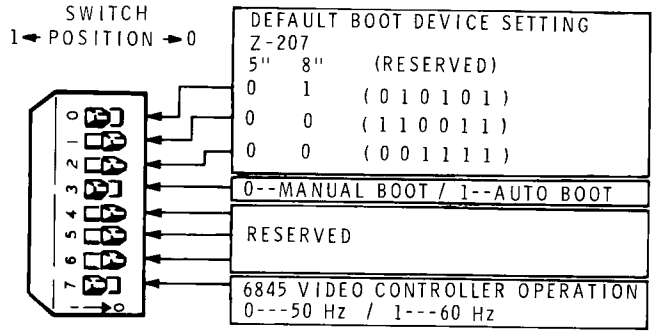



**Figure B-1**
Automatic/Manual Bootstrap Setting (Switch SW-101)

# Entering Manual Bootstrap Commands

When the SW-101 switch is set for automatic bootstrap, your Computer automatically boot up with your Z-100's lefthand drive (if your Z-100 is a Low-Profile) or with your Z-100's upper drive (if your Z-100 is an All-in-One). If you wish to boot up with a different drive, you must first set the SW-101 for manual bootstrap. Refer to your Z-100 hardware manual for instructions on setting this switch.

When the SW-101 switch is set for manual bootstrap, the console should display the pointed finger prompt and a flashing cursor when you turn on the Z-100 and enter **CTRL-RESET**, as shown:

You can enter your manual bootstrap command from the keyboard in response to the "pointing finger" prompt. There are several options that are available so that you can perform bootstrap from any of the drives in your hardware environment.

The bootstrap command syntax is:

**<B>**OOT **[<dev>][<#>][:<boot string>]**CR

Where **<B>** is required input that the computer completes with "OOT";
where **<dev>** is an optionl function key (F1 or F2) that determines which device the controller is to use;

where **<#>** is the optional unit number (0 or 1) of the device type connected to the device controller that is to be used;

where **<boot string>** is an optional string of up to 79 characters that is preceded by a colon; and

where CR is a required carriage return.

Table B-1 shows how you can boot up with any of the drives in your hardware environment. This table lists the drive accessed, the actions or commands used to access this drive, and the setting of the SW-101 switch that are necessary.

| Drive | User Actions to Cause Bootstrap | SW-101 Settings (3, 2, 1, 0) |
|---|---|---|
| A: | Power On (Auto-Boot) | 1, 0, 0, 0 |
| | <B>CR | 0, 0, 0, 0 |
| | <B><F1>CR | 0, x, x, x |
| | <B><F1><0>CR | 0, x, x, x |
| B: | <B><F1><1>CR | 0, x, x, x |
| C: | Power On (Auto-Boot) | 1, 0, 0, 1 |
| | <B>CR | 0, 0, 0, 1 |
| | <B><F2>CR | 0, x, x, x |
| | <B><F2><0>CR | 0, x, x, x |
| D: | <B><F2><1>CR | 0, x, x, x |
| NOTE: "x" is SW-101 switch setting indicating that position may be set to anything. | | |

**Table B-1**
Bootstrap Alternatives

To make the manual bootstrap command easier to work with, it uses "logical devices" to make the distinction between the different drive types connected to your system. The difference between the actual drive controller board and the device type used by the bootstrap command is illustrated in Figures B-2 and B-3.

Because the Z-207 controller card is designed to control both 5.25-inch and 8-inch disks, a method must be used that makes a distinction between these two. This method must also allow for future expansion where one controller card might control only one type of device, or it might control several devices. (See Figure B-4, which illustrates the theory of such expansions.)

NOTE: Figures B-2 through B-4 illustrate features of the Z-100 Computer. These features are not all necessarily used by this CP/M system.

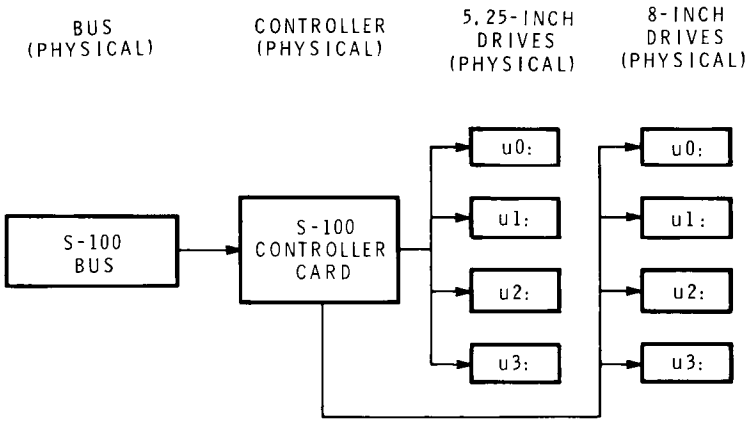



**Figure B-2**

Physical Connections from the Hardware Viewpoint

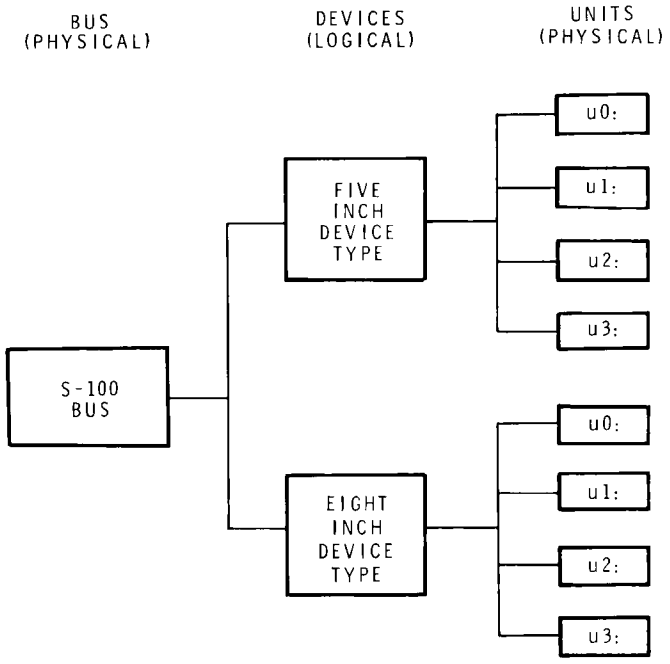



**Figure B-3**

Logical connections from the Bootstrap Command Viewpoint

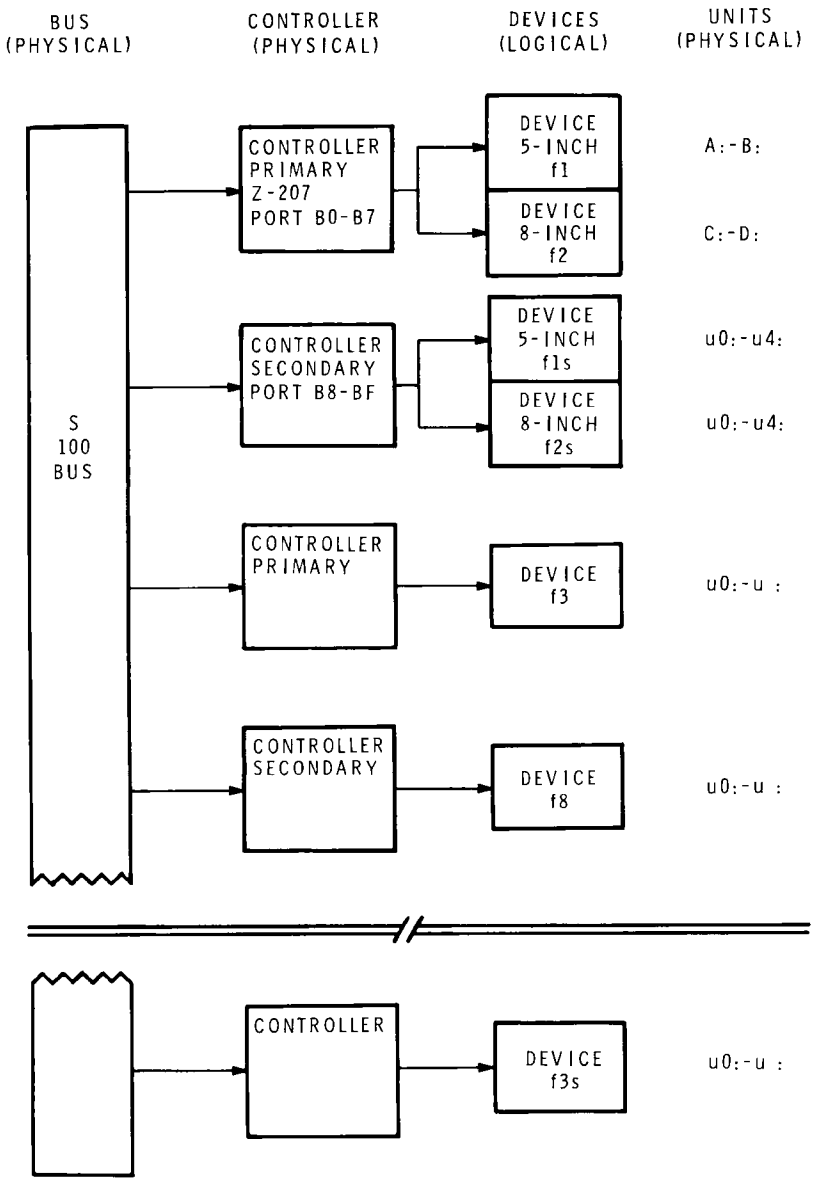



**Figure B-4**

Logical Device Extensions With SW-101 DIP Switch

The controller shown at the top of Figure B-4 shows the Z-100's standard controller card (Z-207), the other controllers that are shown are just theoretical for this diagram, which illustrates how the manual bootstrap command syntax works for possible device expansion.

# Manual Bootstrap Examples

**Example 1:** To manually boot up from one of the 5.25-inch disks, you type **B** and the computer completes the command with "oot", spelling "Boot".

NOTE: With no other options entered, **B**<CR> defaults to the 0 unit of the switch selected device. If you pressed <CR> before selecting any of the other bootstrap command line options, the system would boot up from the Z-207 controller on the 0 unit of the device selected with your SW-101 DIP switch (5.25-inch device if the setting for positions 0, 1, and 2 are set to 0, 0 and 0, respectively).

Next, you press the <**f1**> function key, which selects the 5.25-inch device on the controller at port B0. In your system, the Z-207 (shown as the top device in Figure B-4) is the one selected.

NOTE: With no other options entered, **B** <**f1**> <CR> defaults to the 0 unit of the device specified by <**f1**>. If you pressed <CR> before selecting any of the other bootstrap command line options, the system would boot up from the A: drive.

Then, to select a particular 5.25-inch drive that is connected to the Z-207 controller, you would press **0** or **1** (the unit number), which applies to drives "A:" or "B:" respectively.

**Example 2:** To boot up from an 8-inch disk, you type **B** and the computer completes the command with "oot", displaying "Boot".

NOTE: With no other options entered, **B** ⓒ defaults to the 0 unit of the switch selected device. If you pressed ⓒ before selecting any of the other bootstrap command line options, the system would boot up from the Z-207 controller on the 0 unit of the device selected with your SW-101 DIP switch (8-inch device if the setting for positions 0, 1 and 2 are set to 1, 0 and 0, respectively).

Next, you press the <**f2**> function key and this selects the 8-inch device from the controller at port B0. In your system the Z-207, shown as the topmost controller in Figure B-4, is the one selected.

NOTE: With no other options entered, **B** <**f1**> ⓒ defaults to the 0 unit of the device specified by <**f1**>. If you pressed ⓒ before selecting any of the other bootstrap command line options, the system would boot up from the C: drive.

Then, to select a particular 8-inch drive that is connected to the Z-207 controller, you would press **0** or **1** (the unit number), which applies to the "C:" or "D:" drives respectively that are software supported in the current system.

Or, if you wanted to select from the secondary controller (at port B8) you would type <**B**> <**S**> to boot from the 0 unit of the 0 device on that controller.

# Manual Bootstrap Procedure

In general, the Manual Bootstrap Procedure requires you to:

- Secure correct power and peripheral connections.
- Turn power on.
- Secure System Distribution disk in drive A:.
- Wait for "pointing finger" monitor prompt.
- Enter the boot up command.
- Await system prompt.

After you have unpacked your Computer and connected any peripheral device to it according to the procedures given in the Z-100 Series User's Manual, proceed with the following steps.

Specifically, the Manual Bootstrap Procedure requires you to:

1. Make certain that the power switch, which is located on the back panel of your Computer, is in the OFF position.

2. Make certain that the power switches on any peripheral devices (printer, modem, video terminal, disk drives) that are connected to your Computer, are in their respective OFF positions.

3. Make certain that the power outlets used for your Computer and peripheral devices are properly grounded in accordance with the appropriate electrical codes. (If you are not certain, have a qualified electrician check!)

4. Double check all cables and cords to make certain that they are securely connected.

5. Turn the power switch on each of your peripheral devices to the ON position.

6. Turn the power switch on your Computer to the ON position.

7. Within a few seconds, you should see a "pointing finger" (the Monitor Prompt) in the upper lefthand corner of your video terminal's screen.

NOTE: If you do not see the pointed finger prompt, then you should:

- Turn the Computer's power switch to the OFF position.
- Turn off the power switches to any connected peripheral devices.
- Refer to the "In Case of Difficulty" section in your Z-100 Series User's Manual.

8. Place CP/M Distribution Disk I into the lefthand drive (for Low-Profile) or upper drive (for All-in-One) of your Z-100 and close the drive latch.

9. Type the letter **B**. The word "Boot" appears to the right of the finger.

10. Type ⓒⓡ to boot from 5-inch drives.

11. A message in the following form should appear after several seconds:

```
CP/M-85 VERSION 2.2.100  07/07/82

A>
```

NOTE: If the CP/M identification message and system prompt have not appeared after a minute, press **CTRL-RESET** (both the CTRL and RESET keys are located on the left side of the keyboard). The pointing finger prompt should appear in the upper lefthand corner of your screen.

- If the pointing finger appears, try the manual bootstrap procedure again, beginning from Step 9.
- If no pointing finger appears, press **CTRL-RESET** and repeat this procedure, beginning from Step 9. However, if the pointing finger prompt still does not appear, refer to the section entitled "In Case of Difficulty" in the Z-100 Series User's Manual.

*Appendix C*

# ALTERNATE CHARACTER FONTS

The CP/M-85 Operating System is equipped with extra character fonts that enable your Z-100 or H-100 Computer to display characters from one of seven different foreign languages or an assortment of graphic characters.

The characters for each font are stored in files with the .CHR file name extension. The primary file name of each file identifies the type of characters it contains. The character font files reside on Distribution Disk I under the following names:

DANISH.CHR
ENGLISH.CHR
FRENCH.CHR
GERMAN.CHR
ITALIAN.CHR
SPANISH.CHR
SWEDISH.CHR
GRAPHICS.CHR

When stored in files by these names, the fonts are inactive. However, when any of these files is copied to the file named ALTCHAR.SYS, the font becomes active.

## ACTIVATING ALTERNATE FONTS

To activate one of these alternate character fonts (cause its characters to display, copy this file the file name ALTCHAR.SYS. Use a command in the following form:

A>**PIP ALTCHAR.SYS={fontname}.CHR[V]**↵

Where **PIP** is the file copying command, explained on Page 2-148;
where **ALTCHAR.SYS** is a file name that will be loaded into memory during a cold boot if it exists on the disk in drive A;
where {**fontname**} is the primary name of an alternate character font file; and
where [**V**] is a command parameter that causes PIP to verify the accuracy of the copy operation.

After entering such a command, reset and cold boot the system.

For example, you can obtain screen displays with Italian characters by entering the following command:

A>**PIP ALTCHAR.SYS=ITALIAN.CHR[V]**CR

and then resetting and cold booting the system.

## ALTERNATE FONT OPERATION

When you cold boot the system, CP/M-85 checks the disk in drive A for the file ALTCHAR.SYS. If this file is on the disk, it is loaded into memory with the system kernel and the BIOS files. When in memory, the ALTCHAR.SYS file will cause screen displays of the characters it contains.

These characters will have been copied from a foreign language file or the graphics file; or they may be the characters stored in the default ALTCHAR.SYS file supplied with your distribution software. (The default ALTCHAR.SYS file is identical to the GRAPHICS.CHR file.)

## ALTERNATE FONT FILE FORMAT

The format of an alternate font file consists of three parts: the keyboard mapper, the font plotter, and the display mapper.

The keyboard mapper is in the following form:

| Keyboard {code} | Map {swap} |
|---|---|
| • | • |
| • | • |
| • | • |
| FF | FF |

Where {**code**} is the value generated by the keyboard processor;
where {**swap**} is the value that {code} is to be mapped to; and
where **FF** (in hexadecimal) is the terminating byte for both the {code} and {swap} data.

The font plotter follows the FF terminator of the keyboard map. For each mapped character, the plotter includes a one byte value representing the character and a nine byte description of the character. Each of the description bytes indicates the state of the eight pixels in a single line within the eight-by-nine pixel matrix. The font information for all of the mapped characters terminates with FFH or an End-Of-File (1AH) value.

For example, Figure C-1 illustrates the eight-by-nine pixel matrix used to plot the "bullet" graphics character (*). The value for this character is 5EH, and the nine description bytes of the pixels turned on and off for this character are listed to the right of each pixel line in Figure C-1.

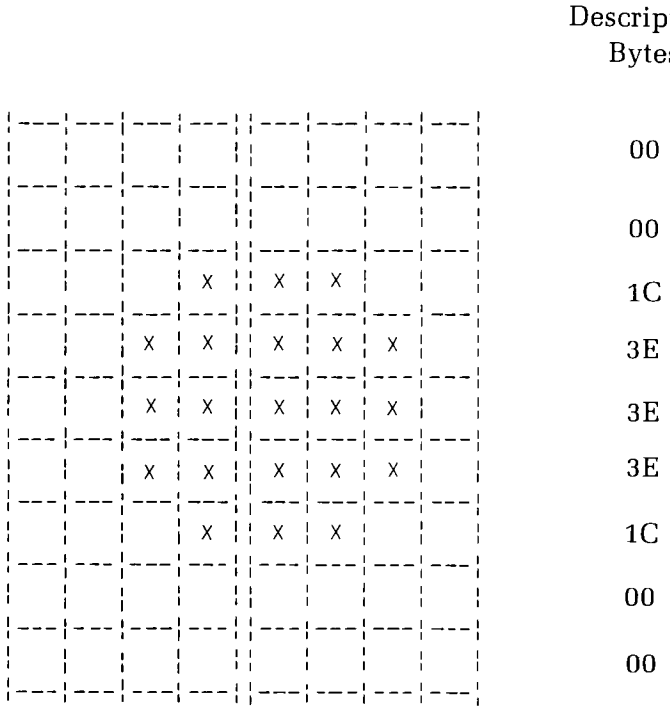



**Figure C-1**
Plotting of the "Bullet" (*) character in ALTCHAR.SYS

NOTE: Each box in Figure C-1 represents one pixel. If the box is empty, the pixel is off. If the box is filled in, the pixel is on. The first nibble of each description byte applies to the first four pixels in a line. The second nibble of each description byte applies to the second four pixels in a line.

Hence, the font plotter bytes for the "bullet" character would be "5E 00 00 1C 3E 3E 3E 1C 00 00".

The display mapper is formatted similarly to the keyboard mapper. However, display mapper values have had 20H (ASCII space) subtracted from them.

*Appendix D*

# Assembler Operation Codes

Assembly language operation codes form the principal part of assembly language programs, and form the operation field of the instruction. In general, ASM accepts all the standard mnemonics for the Intel 8080 microcomputer, which are given in detail in the Intel manual "8080 Assembly Language Programming Manual." Labels are optional on each input line and, if included, take the value of the instruction address immediately before the instruction is issued. The individual operators are listed briefly in the following sections for completeness, although it is understood that the Intel manuals should be referenced for exact operator details. In each case,

| | |
|---|---|
| e3 | represents a 3-bit value in the range 0-7 which can be one of the predefined registers A, B, C, D, E, H, L, M, SP, or PSW. |
| e8 | represents an 8-bit value in the range 0-255. |
| e16 | represents a 16-bit value in the range 0-65535. |

which can themselves be formed from an arbitrary combination of operands and operators. In some cases, the operands are restricted to particular values within the allowable range, such as the PUSH instruction. These cases will be noted as they are encountered.

In the sections which follow, each operation code is listed in its most general form, along with a specific example, with a short explanation and special restrictions.

## JUMPS, CALLS, AND RETURNS

The Jump, Call, and Return instructions allow several different forms which test the condition flags set in the 8080 microcomputer CPU. The forms are:

| | | | | |
|---|---|---|---|---|
| JMP | e16 | JMP | L1 | Jump unconditionally to label |
| JNZ | e16 | JMP | L2 | Jump on non zero condition to label |
| JZ | e16 | JMP | 100H | Jump on zero condition to label |
| JNC | e16 | JNC | L1+4 | Jump no carry to label |
| JC | e16 | JC | L3 | Jump on carry to label |
| JP0 | e16 | JPO | $+8 | Jump on parity odd to label |
| JPE | e16 | JPE | L4 | Jump on even parity to label |
| JP | e16 | JP | GAMMA | Jump on positive result to label |
| JM | e16 | JM | a1 | Jump on minus to label |
| | | | | |
| CALL | e16 | CALL | S1 | Call subroutine unconditionally |
| CNZ | e16 | CNZ | S2 | Call subroutine if non zero flag |
| CZ | e16 | CZ | 100H | Call subroutine on zero flag |
| CNC | e16 | CNC | S1+4 | Call subroutine if no carry set. |
| CC | e16 | CC | S3 | Call subroutine if carry set |
| CPO | e16 | CPO | $+8 | Call subroutine if parity odd |
| CPE | e16 | CPE | S4 | Call subroutine if parity even |
| CP | e16 | CP | GAMMA | Call subroutine if positive result |
| CM | e16 | CM | b1$c2 | Call subroutine if minus flag |
| | | | | |
| RST | e3 | RST | 0 | Programmed "restart", equivalent to CALL 8*e3, except one byte call |
| | | | | |
| RET | | | | Return from subroutine |
| RNZ | | | | Return if non zero flag set |
| RZ | | | | Return if zero flag set |
| RNC | | | | Return if no carry |
| RC | | | | Return if carry flag set |
| RPO | | | | Return of parity is odd |
| RPE | | | | Return if parity is even |
| RP | | | | Return if positive result |
| RM | | | | Return if minus flag is set |

## IMMEDIATE OPERAND INSTRUCTIONS

Several instructions are available which load single or double precision registers, or single precision memory cells, with constant values, along with instructions which perform immediate arithmetic or logical operations on the accumulator (register A).

| | | | | |
|---|---|---|---|---|
| MVI | e3,e8 | MVI | B,255 | Move immediate data to register A, B, C, D, E, H, L, or M (memory) |
| ADI | e8 | ADI | 1 | Add immediate operand to A without carry |
| ACI | e8 | ACI | 0FFH | Add immediate operand to A with carry |
| SUI | e8 | SUI | L + 3 | Subtract from A without borrow (carry) |
| SBI | e8 | SBI | L AND 11B | Subtract from A with borrow (carry) |
| ANI | e8 | ANI | $ AND 7FH | Logical "and" A with immediate data |
| XRI | e8 | XRI | 1111$0000B | "Exclusive or" A with immediate data |
| ORI | e8 | ORI | L AND 1+1 | Logical "or" A with immediate data |
| CPI | e8 | CPI | 'a' | Compare A with immediate data (same as SUI except register A not changed) |
| LXI | e3,e16 | LXI | B,100H | Load extended immediate to register pair (e3 must be equivalent to B, D, H, or SP) |

## INCREMENT AND DECREMENT INSTRUCTIONS

Instructions are provided in the 8080 repetoire for incrementing or decrementing single and double precision registers. The instructions are:

| | | | | |
|---|---|---|---|---|
| INR | e3 | INR | E | Single precision increment register (e3 produces one of A, B, C, D, E, H, L, M) |
| DCR | e3 | DCR | A | Single precision decrement register (e3 produces one of A, B, C, D, E, H, L, M) |
| INX | e3 | INX | SP | Double precision increment register pair (e3 must be equivalent to B, D, H, or SP) |
| DCX | e3 | DCX | B | Double precision decrement register pair (e3 must be equivalent to B, D, H, or SP) |

## DATA MOVEMENT INSTRUCTIONS

Instructions which move data from memory to the CPU and from CPU to memory are given below:

| | | | | |
|---|---|---|---|---|
| MOV | e3,e3 | MOV | A,B | Move data to leftmost element from rightmost element (e3 produces one of A, B, C, D, E, H, L, or M). MOV M,M is disallowed |
| LDAX | e3 | LDAX | B | Load register A from computed address (e3 must produce either B or D) |
| STAX | e3 | STAX | D | Store register A to computed address (e3 must produce either B or D) |
| LHLD | e16 | LHLD | L1 | Load HL direct from location e16 (double precision load to H and L) |
| SHLD | e16 | SHLD | L5+x | Store HL direct to location e16 (double precision store from H and L to memory) |
| LDA | e16 | LDA | GAMMA | Load register A from address e16 |
| STA | e16 | STA | X3-5 | Store register A into memory at e16 |
| POP | e3 | POP | PSW | Load register pair from stack, set SP (e3 must produce one of B, D, H, or PSW) |
| PUSH | e3 | PUSH | B | Store register pair into stack, set SP (e3 must produce one of B, D, H, or PSW) |
| IN | e8 | IN | 0 | Load register A with data from port e8 |
| OUT | e8 | OUT | 255 | Send data from register A to port e8 |
| XTHL | | | | Exchange data from top of stack with HL |
| PCHL | | | | Fill program counter with data from HL |
| SPHL | | | | Fill stack pointer with data from HL |
| XCHG | | | | Exchange DE pair with HL pair |

## ARITHMETIC LOGIC UNIT OPERATIONS

Instructions which act upon the single precision accumulator to perform arithmetic and logic operations are:

| | | | | |
|---|---|---|---|---|
| ADD | e3 | ADD | B | Add register given by e3 to accumulator without carry (e3 must produce one of A, B, C, D, E, H, or L) |
| ADC | e3 | ADC | L | Add register to A with carry, e3 as above |
| SUB | e3 | SUB | H | Subtract reg e3 from A without carry, e3 is defined as above |
| SBB | e3 | SBB | 2 | Subtract register e3 from A with carry, e3 defined as above |
| ANA | e3 | ANA | 1+1 | Logical "and" reg with A, e3 as above |
| XRA | e3 | XRA | A | "Exclusive or" with A, e3 as above |
| ORA | e3 | ORA | B | Logical "or" with A,e3 defined as above |
| CMP | e3 | CMP | H | Compare register with A, e3 as above |
| DAA | | | | Decimal adjust register A based upon last arithmetic logic unit operation |
| CMA | | | | Complement the bits in register A |
| STC | | | | Set the carry flag to 1 |
| CMC | | | | Complement the carry flag |
| RLC | | | | Rotate bits left, (re)set carry as a side effect (high order A bit becomes carry) |
| RRC | | | | Rotate bits right, (re)set carry as side effect (low order A bit becomes carry) |
| RAL | | | | Rotate carry/A register to left (carry is involved in the rotate) |
| RAR | | | | Rotate carry/A register to right (carry is involved in the rotate) |
| DAD | e3 | DAD | B | Double precision add register pair e3 to HL (e3 must produce B, D, H, or SP) |

## CONTROL INSTRUCTIONS

The four remaining instructions are categorized as control instructions, and are listed below:

| | |
|---|---|
| HLT | Halt the 8080 processor |
| DI | Disable the interrupt system |
| EI | Enable the interrupt system |
| NOP | No operation |

# INDEX

The following is a list of error status numbers generated by the Z-100 hardware in floppy disk systems (see Page A-1). For detailed explanations consult the appropriate technical manual.

| Number | Explanation |
|---|---|
| 80 | Not ready. |
| 40 | Write protect violation. |
| 20 | Head is loaded. |
| 20 | Record type. |
| 20 | Write fault. |
| 10 | Seek error. |
| 10 | Record not found. |
| 8 | CRC error. |
| 4 | Found track O |
| 4 | Lost data. |
| 2 | Index hole. |
| 1 | Busy. |

Note that where more than one explanation is offered, the number generated is operation-dependent.

HEATH | ZENITH data systems

# 8″ Diskette Step Rate

Zenith Data Systems is providing 8″ drivers installed in the CP/M-85 disk systems for development purposes and to support future products. Zenith Data Systems does not guarantee proper operation of the 8″ drivers with disk systems obtained from other vendors. However, for the benefit of customers who wish to experiment with non-Zenith hardware at their own risk, the track-to-track stepping rate, which is set at 3 milliseconds, may be changed by using DDT and the SAVE command. In the example shown below, the rate is changed from 3 milliseconds to 15 milliseconds. User input is shown in bold:

```
A>STAT BIOS85.SYS $R/W

BIOS85.SYS set to R/W
A>REN BIOS85.OLD=BIOS85.SYS
A>DDT BIOS85.OLD
DDT VERS 2.2
NEXT PC
0600 0100
-S381
0381 00 03          {00 = 3 mS, 01 = 6 mS, 02 = 10 mS, 03 = 15 mS}
0382 04  .
-S399               {00 = 3 mS, 01 = 6 mS, 02 = 10 mS, 03 = 15 mS}
0399 00 03
039A 04 .
-G0

A>SAVE 5 BIOS85.SYS
A>STAT BIOS85.SYS $SYS

BIOS85.SYS set to SYS
A>STAT BIOS85.SYS $R/O

BIOS85.SYS set to R/O
A>
```

HEATH | ZENITH data systems

# Memory Checking

Modern personal computers are extremely reliable. Your Z-100 operates at a speed of 5 million clock cycles per second and will typically operate for several thousand hours between service calls. However, computers are machines, and machines do, on occasion, develop problems. Among the problems which may occur in all computers are memory failures.

Memory failures may be classified as either "hard" or "soft". Soft memory errors occur randomly at an average frequency of roughly once every several thousand hours of operation and do not indicate the presence of a hardware problem. Hard memory failures are due to defective components within the computer, and will recur frequently until the defective component is replaced.

Your Z-100 computer has special circuitry and additional memory to automatically detect any memory failures (hard or soft) which do occur, through a technique known as "parity checking". When a "parity error" occurs, a display similar to the one below will appear on the screen:

```
ERROR - MEMORY OR BUSS

F =XXXX   IP=XXXX   CS=XXXX   DS=XXXX   ES=XXXX   SS=XXXX   SP=XXXX
AX=XXXX   BX=XXXX   CX=XXXX   DX=XXXX   DI=XXXX   SI=XXXX   BP=XXXX

SYSTEM HALT
```

The purpose of providing the parity error message is to let you know that you may have a problem which may not be readily apparent and which may require the attention of service personnel. Parity checking is an advanced feature found in only a few microcomputer systems. In systems without parity checking, the memory error usually goes unnoticed for a period of days or weeks until the amount of data destroyed becomes so large that it can no longer be ignored.

When an error occurs, the system will display the parity error message described above and halt. The system must then be reset and rebooted, with the consequence that all work in the computer's memory and any unclosed files on the disk will be permanently lost. It is generally best not to use the system any further until the memory test (supplied with your Z-DOS operating system) has been run (preferably from a write-protected disk!). If the memory test does not turn up problems after several hours of operation, you should resume normal operation. If subsequent memory failures occur, you should copy down all of the data presented on the screen by the parity error routine and retain it for use by service personnel. Also, record which program and operating system you were using at the time. If multiple errors occur within 30 days, a serviceman should be called, even if the memory test does not indicate the presence of problems.

OS-53-2/595-2824
OS-63-4/595-2827
591-3930

HEATH | ZENITH data systems

CP/M-85 4-21-83

## IMPORTANT NOTICE

Dear Customer,

We have made every attempt to provide you with the best possible product in the most timely manner. Several changes and enhancements were made in response to your needs just prior to production. Where those enhancements affect the manual, we are providing you with replacement pages. Please take these page(s) (included in this document holder) and use them to replace the original(s) in your Manual.

Thank you,
Zenith Data Systems


595-2824
591-3932