Z-DOS™

Volume I

593-0042
CONSISTS OF

MANUAL
595-2827
FLYSHEET
597-2810

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

ZENITH data systems

HEATH

IMPORTANT NOTICE

The Z-DOS™ and MS™-DOS operating systems have undergone a number of performance enhancements to both the system and some of the utilities. In an effort to keep you up to date, we are supplying the enclosed update which contains the latest version of the operating system and the affected utilities. Please note that the utilities supplied on the update disk should be used only with the new version of the operating system. **Do not try to use your old utilities with the new system, or the new system with the old utilities**. The best way to ensure proper installation of the new system is to follow the installation procedure supplied with this letter. This procedure will generate a new set of master distribution disks which can be used to generate your working disks.

Thank you,

Zenith Data Systems Corporation

Z-DOS
Revd 2-27-85

™ MS is a trademark of Microsoft Corporation.
™ Z-DOS is a trademark of Zenith Data Systems Corporation.

Installing The Update:

We recommend that you use this update to generate a new set of distribution disks for your operating system. This will free you from having to remember which files to copy from the update disk, and which to copy from the original disks at a later time. Use the following procedure to generate the new distribution set. You will need your original distribution disks and 2 blank disks in addition to the update disk.

1. Boot the update disk on your machine. Follow the instructions provided with your original disks on booting from a 5.25-inch floppy disk.

2. You will now start the actual installation procedure using one of two variations depending on your hardware configuration.

Method 1:

If you have a Winchester and only one floppy drive, it will be fastest to use a part of the Winchester as a temporary storage area while you are creating the new distribution disks. Note that this method will NOT work for the version 1 DOS update. To use this method, you must have a Winchester DOS partition with at least 600K bytes of free space. Use the ASSIGN command to assign it to a valid drive letter. A subdirectory will be created on this partition which will be used during installation. This directory and its subdirectories are only temporary, and may be removed after the update procedure is complete. At the system prompt, type the command

INSTALL1 W <drive> <name>

where <**name**> is the optional name of the directory that will be created, and <**drive**> is the optional drive letter that the partition is assigned to. Do not put a colon after the drive letter. If neither <**name**> nor <**drive**> is specified, the defaults will be used. The default drive on a Z-100 is E and on a Z-100 PC is C. The default name for both machines is $$UPD$$. You may specify a drive letter, but not a directory name if you wish, but you CANNOT specify a directory name without a drive letter. For example, if you wanted to use the default directory name, but you wanted it on drive F:, you would type the following command:

INSTALL1 W F

If you wanted both the default name and drive letter, you would type the command

INSTALL1 W

Finally, if you wanted to name your directory UPDATE and put it on drive G:, you would type

INSTALL1 W G UPDATE

Follow the instructions given by the program.

Method 2:

If you do not have a Winchester or have more than one 5.25-inch floppy drive on your system, or are installing a version 1 update, then you should use this method. At the system prompt, simply type

INSTALL1 RETURN

If you have only one 5.25-inch drive, then all references to drive B: should be interpreted as drive A: ONLY when you are prompted to insert disk B: into drive A:. You will be prompted to change the disk in drive A: fairly often if you have only one floppy drive, so make sure you keep track of which disk is A: and which disk is B:. You should make sure that your original distribution disks and your update disk are all write protected in case you make a mistake and insert them into the drives at the wrong time.

3. After INSTALL1 has quit running, you will be asked to run a program called INSTALL2. You may also be asked to specify two drive letters on the command line. This is the next step in the installation process. To run it, type the command

   INSTALL2 <d1> <d2>

where **<d1>** and **<d2>** are the drive letters that were specified at the end of INSTALL1. Again, just follow the directions given by the program, changing disks when prompted to do so.

4. After INSTALL2 is done, you will be asked to run INSTALL3. Again, you may be asked to specify two drive letters on the command line. This is the third and final step in the installation procedure. When it is done running, you will have a new set of distribution disks for your operating system. The new disk I may still contain a file called INSTALL2.BAT which you may delete if desired. Likewise, the new disk II may have a file called INSTALL3.BAT which may be deleted. It is advisable to make copies of the new distribution set and store one copy with your original distribution disks.

5. Extensive modifications have been made to the format module. Therefore, if you have a Winchester hard disk in your system, we strongly recommend that you back up all of the files from it onto floppies and reformat the hard disk using the new system and new format program. You may then restore your files from the floppies. It is important that you do not overwrite the new system when you restore your files from the floppy disks. In order to avoid this problem, you should follow the suggested outline.

    1. Boot the update disk, then backup all files using the BACKUP utility supplied on the update.

    2. Format the Winchester partition with the /S switch to put the new system on the disk.

    3. Recreate the directory structure (if any) on the partition with the MKDIR command (MS-DOS ver 2 only).

    4. Copy all of the files from the update disk onto the partition, placing them in the appropriate directories.

    5. Now restore your files from the floppies using the RESTORE command. When RESTORE tries to restore a file which has been replaced by one on the update disk, it will issue a message stating that the file in question already exists. It will then ask if you want to over write it. Answer **N** for no. This will ensure that your partition has the most recent system and utilities.

TABLE OF CONTENTS

TABLE OF CONTENTS

Commanding Z-DOS

Getting Started

TABLE OF CONTENTS

PART 2. Z-DOS Reference Guide

Z-DOS Commands

TABLE OF CONTENTS

PART 3. Comprehensive Utility Details

TABLE OF CONTENTS

TABLE OF CONTENTS

MACRO-86

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

PART 4: Appendices and Index

Appendices

Appendix Q Procedure to Change Disk Parameters

Memory Checking

ZENITH | data systems

6-18-85

Dear Customer,

We have made every attempt to provide you with the best product in the most timely manner. Since the initial production of the Z- DOS Winchester Supplement, some additional information has been added to the Z-DOS Operating System Manual (part number 595- 2827).

Please check page 5.18 of your Z-DOS Operating System Manual. If it lists "Method 3: Single Disk Backup Procedure" then this Notice can be disregarded.

If page 5.18 does not list "Method 3: Single Disk Backup Procedure" then replace the corresponding pages in your manual with the new pages supplied with this Notice.

NOTE: If your system has only one physical disk drive slot that can accommodate your Z-DOS distribution disk media, then you should follow the backup procedure in "Method 3: Single Disk Backup Procedure" instead of either Method 1: Make Backup Procedure or Method 2: Interaction Backup Procedure because these procedures will require you to insert disks into the drive an inconvenient number times.

Thank you,

ZENITH DATA SYSTEMS

Z-DOS Preface

This reference manual is designed with a specific goal in mind: provide a useful tool for both a first-time computer user and for an experienced user. To aid in this goal the manual is divided into four principal parts:

PART 1. General Introduction to Z-DOS

PART 2. Z-DOS Reference Guide

PART 3. Comprehensive Utility Details

PART 4. Appendices, Glossary and Index

Each of the logical topics in this manual are subdivided into categories, where it is applicable. This was done so that the information may be used to your utmost advantage, whatever your level of experience is with computers. These categories are:

- Brief (major point synopsis)
- Details (basic descriptions)
- Checkpoint (verify and validate)
- Application (system integration)

Physical Organization

Brief

"Brief" is a concise synopsis of the key points covered in that section. It is always located at the beginning of every applicable topic.

- If you have had *experience* with a computer system before, you will want to read the "Brief." If that tells you enough information, you can then read the "Application" or go on to another section.

- If you have not used a computer system before or feel that you are just a *beginner*, read the Brief and continue with the information that follows this synopsis. Then, if you need to go back and reread a section to refresh your memory, reading just the Brief may be enough.

Details

"Details" is a basic and easy-to-follow explanation of all the information covered within the section. The Details are always the first information in a section unless preceded by a Brief.

- If you are an *experienced user*, you may want to skip over this information unless you want a more detailed explanation than is covered within the Brief.

- If you are a *beginning user*, you should read this information for a general understanding of the subject that is covered.

Checkpoint

The information marked "Checkpoint" is used periodically to provide you with a method to verify your progress as you proceed through the steps. You can check what is happening on your computer against what should be happening, in the case that they are not the same. When the checkpoint category is used in sections of this manual, you may expect to find specific steps to follow if you get an error message. Checkpoint also gives guidelines to ensure correct operation while using the function being covered. These checkpoints are labeled and they appear at the appropriate places throughout the information.

- If you are an *experienced user*, you may want to skip over this information until you need it; although you may want to read these to alert yourself to the steps in the ongoing process. Checkpoint also notes possible errors and specific steps to recover from these errors.

- If you are a *beginning user*, you definitely should read this information.

Application

"Application" is at the end of each applicable section. The Application portion of a section gives an in-depth look at the way the material covered integrates into the whole system. Application is labeled as such.

- If you are an *experienced user*, you may want to read these for a look at the impact of what is being described.

- If you are a *beginning user*, you will want to read this information so that you can see how the topic may be applied for your particular needs.

Content Organization

Content Organization

PART 1, "The General Introduction to Z-DOS" begins on Page 1.3, "The General Overview". This chapter assists you in understanding your operating system and manual.

The first two sections, "How Your Computer Operates" and "Introduction to Operating Systems", starting on Pages 1.3 and 1.7, respectively, are primarily for the individual that has little or no prior experience with a computer system. These sections provide you with an elementary understanding of what your computer does and what a basic operating system is, and how both work.

Under the heading "Z-DOS and the Operating Environment," you will find specific information that relates specifically to both Z-DOS and your Z-100 computer.

"About Boot-up and System Initialization", beginning on Page 2.1 introduces you to "Boot-up" and explains what goes on when you first start your computer using Z-DOS. "System Resources", on Page 2.6 gives a more detailed look at how the various components of hardware are managed by the operating system.

PREFACE

Z-DOS Preface

"Chapter Three: Z-DOS Conventions", on Page 3.1, covers the conventions that you use to get the operating system to work for you.

"Files and Filenames" begins on Page 3.1. "Referencing Several Files at One Time" begins on Page 3.6. "Disk Drives" begins on Page 3.10.

"Chapter Four: Commanding Z-DOS" gives you details on how to get Z-DOS to do what you want.

"Commands and Command Lines" on Page 4.1, discusses what the types of commands are and how they work with the operating system. "Command Line Entry," Page 4.7, instructs you in the easiest methods to issue commands. "COMMAND.COM and Command Execution" on Page 4.21 details how the Z-DOS command processor operates.

"Chapter Five: Getting Started", on Page 5.1, covers the precautions you should observe while using disks in the "Care and Handling of Disks" section on Page 5.1. How to start your Z-100 computer with Z-DOS is in "Z-DOS System Start Up" on Page 5.2. The necessity of making copies of your disks is in "The Importance of Making Backup Disks" on Page 5.16. A procedure for making backup copies for novice users is found in "Create a Backup System Disk" on Page 5.18.

PART 2, "Z-DOS Reference Guide", is the technical reference guide. It uses the Brief/Details/Checkpoint structure extensively. In this part of the manual, the Checkpoint gives you information on what may go wrong while you are using each command and how to correct the error.

The commands are listed in alphabetical order in "Commands in Brief", Page 6.4.

Beginning on Page 6-7, "List of Disk File Contents and Command Summary", provides a quick summary of the files on your Z-DOS System Disk and the commands that are available with Z-DOS. In the "Command Reference Guide", the Z-DOS commands are given alphabetically by command name. Those commands, which are utilities and that have extensive options for operation (such as DEBUG, EDLIN, or MASM), have each of their options listed under them in alphabetical order. Further details on these extensive utilities are in separate chapters in PART 3.

PART 3, "Comprehensive Utility Details" contains: "DEBUG", "EDLIN", "FILCOM", "MACRO-86", "LINK", "LIB" and "CREF".

PART 4 contains the "Appendices" and the "Index". The "Appendices" contain additional reference information on the following subjects:

- Error Messages (including what to do after you get such a message).
- Z-DOS hardware environment (a brief description of the Z-100 computer system).
- Z-DOS on a single disk drive system (how Z-DOS utilities work on a system with only one drive).
- Disk Allocation, Directory Structure and File Allocation Table.
- Interrupts and Function Calls.
- Control Blocks and Work Areas.
- I/O Address and System Memory Maps.
- Macro Assembler Directives (alphabetical and topical lists).
- 8088 (8086) Mnemonic Instructions (list of 8088 opcodes).

Notation Used in this Manual

The notation that is used in this manual may be defined as follows:

< > Angle brackets indicate an item that is to be entered (when boldface alone does not suffice).

If these brackets contain lowercase text, that text defines the type of response that you make (i.e., your response belongs to a class or type that is described by the text).

If these brackets contain uppercase text, that text is the literal name of a key or a command (file).

[ ] Brackets indicate that the item enclosed is optional.

. . . An ellipses means that the preceding item may be repeated as many times as is needed.

In addition to the above, the following notation is also used:

RETURN to signify the RETURN key (a carriage return is to be entered).

CTRL to signify the "CTRL" key (Control).

CTRL-<letter>	To signify that the "CTRL" key and one named by <letter> should be pressed simultaneously, where <letter> is the name of a key on the keyboard.
F<number>	To signify a special "function" key labeled with <number>, where <number> is any number from 0 to 12.
<d:>	To signify a disk drive name.
<dev>	To signify a logical device name.
<filename>	To signify the primary name of a file.
<.ext>	To signify the extension of a file.
<filespec>	To signify [<d:>]<filename>[<.ext>], where a specific file must be given. The drive name is needed only if the file is not resident on the current default drive. If there exists more than one file with the same primary name, the extension is needed. A <filespec> can also signify a logical device name (<dev>).
<parameters>	To signify a list of specifications or responses to a series of questions that are required by a command or a list of options that are to be used at your discretion.

Part 1

GENERAL INTRODUCTION TO Z-DOS

How Your Computer Operates

Brief

Computers "Input", "Process", "Store" and "Output" information. To perform these tasks, a computer uses three types of programs — Languages, Applications and Utilities.

Languages are used to create computer programs. These programs perform functions and tasks defined by the specialized vocabulary of the language.

Applications perform functions and tasks that are required by the user. Applications would be programs used for word processing, accounting, market forecasts, etc.

Utilities perform functions and tasks (housekeeping) necessary for the computer system. Utilities can be programs to check available disk or memory space, copy files from one location to another, erase files, etc.

Details

Functions

What goes on "inside" the computer?

When you use your computer, four basic operations go on inside. These operations are:

- Input
- Processing
- Storage
- Output

GENERAL OVERVIEW

How Your Computer Operates

Input

Input occurs when information enters the computer. This can occur in several ways. One way is through a typewriter style keyboard. The keyboard may be attached to a video unit. The video screen is not required for input. Input also occurs when the computer receives information from a disk, a lightpen, or a joystick.

An example of input would be typing in the name of a program to tell the computer which program you want to run.

Processing

The computer processes data (input), makes decisions, calculates mathematical problems, and generates an output. You will have an example of processing when you start Z-DOS and it asks you for the date. After you input a date, the computer checks to see if that date is possible (i.e., it processes your input to make sure you did not say February 30th).

Storage

A computer can store information temporarily in internal memory or more permanently on a disk.

Internal memory stores information only as long as it is being worked on by the computer. This type of storage usually takes place in electronic devices called RAM (Random Access Memory). Internal memory is generally considered temporary storage.

Disks store information when that information is not in immediate use. Disks are considered permanent storage because the information is retained when they are removed from the computer system.

Output

Output occurs when the computer transfers information from memory to a printer or terminal screen. Output that is not readily visible occurs when the computer writes information out to a disk for storage.

Programs

How Do Computers Know What To Do?

Computers must be told what to do. In order for computers to be useful, someone must write a set of detailed instructions for the computer to follow. A set of instructions that direct the computer and its associated components to produce desired results is called a "computer program".

Types of Programs

There are many different computer programs available to instruct the computer as to how to perform particular functions. For the purposes of this manual these programs will generally be referred to as either — Languages, Applications or Utilities.

Languages

A language program enables the computer to understand your instructions. Languages act as translators. They allow you to use commands more meaningful to you than the binary numbers the computer understands. Languages are groups of instructions that allow you to direct the computer's action with a specialized vocabulary.

Many different languages exist. Each has its own methods to get tasks done. It is often more expedient to use one language's features over another's because that language's design is better suited to perform the task at hand. Examples of common languages are BASIC, COBOL, FORTRAN and Pascal.

Applications

An application program generally accepts information, manipulates it in a specific way, and returns the results of those manipulations to you. Examples of different types of application programs include: word processing, an accounting package, a client name and address file, a statistics package, a game, or a program delivering a tutorial on some subject.

GENERAL OVERVIEW

How Your Computer Operates

Utilities perform housekeeping functions inside the computer. These programs report information about the system's status, perform tasks that are necessary for computer operation or move information from one location in the system to another. Examples of utilities you will come across in this manual are FORMAT, SYS, COPY, and FILCOM.

Utilities

Application

Until recently, most of the terms found in this section were the jargon of a very specialized group of people. Now these terms have come into common usage by the general public.

One of the new buzzwords used by the media today is "computer literacy". A computer literate is somone who knows about computers, what they do, how they operate, and what the most general computer terms mean. Computer literacy is becoming an important feature of society as more uses for computers are found daily in homes and businesses.

Introduction to Operating Systems

Brief

Traditional Operating Systems have four parts that facilitate software to hardware coordination:

- I/O Management
- Memory Management
- File Management
- Command Processor (Executive)

Details

What is an Operating System?

An "Operating System" is a set of programs to help other programs use the computer and its associated equipment. An operating system contains instructions that the computer needs in order to receive information from the keyboard, send information to a printer, through a modem or to the video unit's screen (CRT). An operating system also contains instructions the computer needs in order to store information on disks and to retrieve that information. Throughout this section, a model of an operating system is discussed.

The Reason There are Operating Systems

Every program used with your computer needs to direct the computer in specific ways to perform its tasks. Without an operating system, each program would have to contain separate instructions to direct the computer.

If the program required files to be written on a disk, it would require the detailed instructions telling the computer to do so. If the program required information to display on the video display, it would need to contain all of those instructions. This would make the program large and cumbersome, with more chances for the programmer to make errors in the instructions.

GENERAL OVERVIEW

Introduction to Operating Systems

Operating systems gather all of these and other instructions together and eliminate the need to write them into every program. With an operating system, the program asks the operating system to perform a required task. When a program makes this type of request to the operating system, it is known as a "system call" because it calls on the operating system to perform a task.

Figure 1.1 shows the relationship between the operating system, the computer (hardware) and the program (shown here as an application program to distinguish it from the utility programs that the operating system uses).

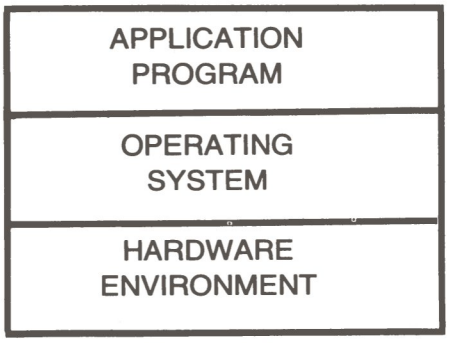


Figure 1.1
Relative System Structure

Four Parts of an Operating System

A typical operating system has four basic systems that:

1) Request programs and cause the computer to execute the instructions in the program;

2) Manage the input and output in the system;

3) Manage information within files; and

4) Manage information in memory.

Executive

One component of an operating system is the "executive." It is in control of the operating system, directs functions and sees that the correct programs are performed. The executive has three associate components that are called "managers." Each manager performs a different function in the organization.

It is the executive's responsibility to respond to your request for a program. When you make a request, the executive will direct the necessary operating system components to: locate the program, copy it into a workspace in memory, and begin execution of the program's instructions.

Input and Output Manager

One manager in the operating system directs all of the information traffic. This is the Input and Output (often abbreviated as "I/O") manager. The I/O manager's job is to coordinate all transfers of information between devices connected to the computer. This manager also knows which device (peripheral) is connected where (at what address) in the computer. This permits each device to send and receive information in different ways from different locations.

File Manager

The file manager handles all of the system's information that is to be stored in files and not in the computer; it packs the information into indexed groups to send to a disk file. The file manager also keeps track of where individual files are located on the disk in order to retrieve information when it is needed. It records the file's names and location in a directory that it writes onto each disk. The information that is stored by the file manager is considered permanent or long-term storage because the files remain stored on the disk after it is removed from the system.

Each time new disks are inserted into the computer's disk drives, the file manager must read the directory on the disk to learn what files are stored and where they are located.

GENERAL OVERVIEW

Introduction to Operating Systems

Memory Manager

Another manager handles all of the computer's immediate records. The memory manager allocates available memory resources. All of the records stored in the computer's memory are controlled by this manager.

Records in "RAM" are erased when the power is turned off. RAM is considered short term memory.

How do the Executive and Various Managers Collaborate?

In an operating system, you request a program from an input device such as a terminal. The executive monitors the terminal, interprets your request for a program and then requests the file manager to get the program from its files (unless it knows that the program requested is already in memory).

The file manager searches its directory for the file needed and loads (copies into memory) the program (if it finds the file in its directory). The program is loaded to a waiting area called a buffer. The file manager reports the program's status to the executive. If the file is not found, the programs status would be "not found". If the file is found, the manager reports back to the executive that the program is waiting in the buffer.

The executive then requests the memory manager to reserve a portion of memory for the program. The program moves from the buffer to the designated workspace and the executive starts the operation of the program.

The program may need additional memory while it performs its tasks, or the program may need to use a line printer, or store files. If so, it passes its request (system call) to the executive, who then refers the requests to the memory manager, I/O manager, or file manager, respectively.

When the program finishes its tasks, the executive regains control of the system and waits for additional commands.

Application

There are many different operating systems on the market. Each operating system has some feature or features unique to it and no other. However, most operating systems have the four basic components in common. Their approach may be different. This should make it easier for you to determine the basic differences and understand how your operating system makes computer use easier.

A prime factor in the selection of a microcomputer operating system is the amount of available software support. Zenith Data Systems/Heath has a complete line of software products for operation under Z-DOS, and more are being added.

Z-DOS AND THE OPERATING ENVIRONMENT

About Boot-up and System Initialization

Brief

During the Boot-up procedure, the system is initialized by the following sequence:

- The "Boot Loader" is loaded in memory by the MTR-100 ROM.
- IO.SYS (I/O Manager) is loaded into memory and displays its banner. Then,
- IO.SYS determines hardware status, and initializes the attached devices (including the logical devices CON, AUX and PRN); then,
- Z-DOS.SYS (File Manager) is loaded by IO.SYS. Then,
- Z-DOS.SYS displays its banner and is initialized by IO.SYS for internal working tables, a correct location for FATs (File Allocation Table), directory and data buffers, then,
- IO.SYS initializes the disk drives and looks for a file called ALTCHAR.SYS (see Appendix N) on the default drive.
- IO.SYS loads COMMAND. COM (Executive) into a location allocated by Z-DOS.SYS. Control passes to,
- COMMAND.COM (details on COMMAND.COM begin on Page 4.21), and COMMAND.COM's banner is displayed.

Details

Boot-up

How does Z-DOS get Started?

When you turn on your Z-100 computer with the Z-DOS System disk installed in the primary disk drive (the top drive in the All-in-One,the lefthand drive in the Low-Profile), there is a short pause and then the light on that drive will glow. When this occurs, your system is performing what is known as an automatic "boot-up" or "boot procedure".

Z-DOS AND THE OPERATING ENVIRONMENT

About Boot-up and System Initialization

Each time your computer is turned on, it normally will boot-up the operating system, automatically loading Z-DOS from disk into memory. The area of memory where the operating system is loaded is referred to as system memory.

What is a Boot Procedure?

The "boot" is a small program stored within the hardware (MTR-100) and serves to establish a communications link between Z-DOS and the various physical parts of the computer. The boot-up procedure is so named because, by means of this procedure, Z-DOS "pulls itself up by its bootstraps". The term "bootstrap" has been shortened by common useage to "boot". This procedure enables Z-DOS to lift itself off of the disk and into the computer's memory. Once Z-DOS is installed in memory, it issues instructions and coordinates the actions of the appropriate parts of the computer.

Procedure During Boot-up

During the boot-up procedure, the boot program checks the lefthand drive of a Low-Profile Z-100 (top drive in the All-in-One) for a disk and searches a particular location on the disk for a boot loader.

The boot loader is recorded on a disk whenever Z-DOS is placed on the disk. If the boot finds the boot loader, it knows that the disk is prepared for the operating system. The disk is considered "bootable". A bootable disk is one from which the boot can place the operating system into memory.

After the boot loader is found, it is copied from the disk into memory and continues the boot-up process. The boot loader looks for, locates, and copies (loads) the Z-DOS system from the disk into memory (if the operating system files are not present, it informs you). Once the operating system is in memory it examines the hardware environment to determine its characteristics (memory size, location and type of peripherals, etc.), and prepares both itself and the hardware for operation, and then takes control.

The Z-DOS Components

Z-DOS is composed of three basic components. When Z-DOS is copied from the disk into memory during the boot-up procedure, three files that make up Z-DOS are transferred. (An explanation of files and filenames is found on Page 3.2.)

These files are:

IO.SYS
Z–DOS.SYS
COMMAND.COM

When the boot loader is first copied into memory, Z-DOS checks the disk's directory to make certain that the first file listed is IO.SYS. The boot loader then loads this file into memory. If this file is not the first file in the directory, an error message is displayed on the video screen.

System Initialization

Z-DOS System Set Up

After IO.SYS is read into memory, control is passed to it from the boot loader and a series of setup routines is performed. These routines prepare the system for operation and prepare the peripherals that are connected. This preparation is known as "initialization".

Initialization is a term that you will see in many computer manuals and articles. It generally means that something is being made ready for immediate or first use. IO.SYS then moves the Z-DOS.SYS file to a specific location in memory and Z-DOS displays a banner that gives the information about the version in use.

Z-DOS.SYS internal structure is initialized and specific areas are set aside within memory. These areas are used for workspace and for storage of directory information from each disk drive that is connected. When IO.SYS finishes these tasks, the disk drives are initialized and IO.SYS looks on the default drive for a file named ALTCHAR.SYS (see Appendix N), which is used for special character fonts. If ALTCHAR.SYS is found, it is read into memory at a specific location.

IO.SYS then checks the disk to find a file named COMMAND.COM and loads it into a specific workspace in memory. IO.SYS then passes control over to COMMAND.COM, and COMMAND.COM displays its banner. COMMAND.COM is described on Page 4.21.

These three files combine to form the operating system that controls all system resources. Note that Z-DOS.SYS and IO.SYS are "hidden" files and are not displayed when a directory command is executed.

All of these steps occur in a few seconds: the boot occurs, Z-DOS initializes itself and its hardware, and control is passed to the command processor to wait for your commands.

Z-DOS AND THE OPERATING ENVIRONMENT

About Boot-up and System Initialization

System Resources

The system and peripherals connected to the computer are known as System Resources. The relationship between the operating system and the system resources is shown in Figure 2.1. General information about the System Resources begins on Page 2.6.

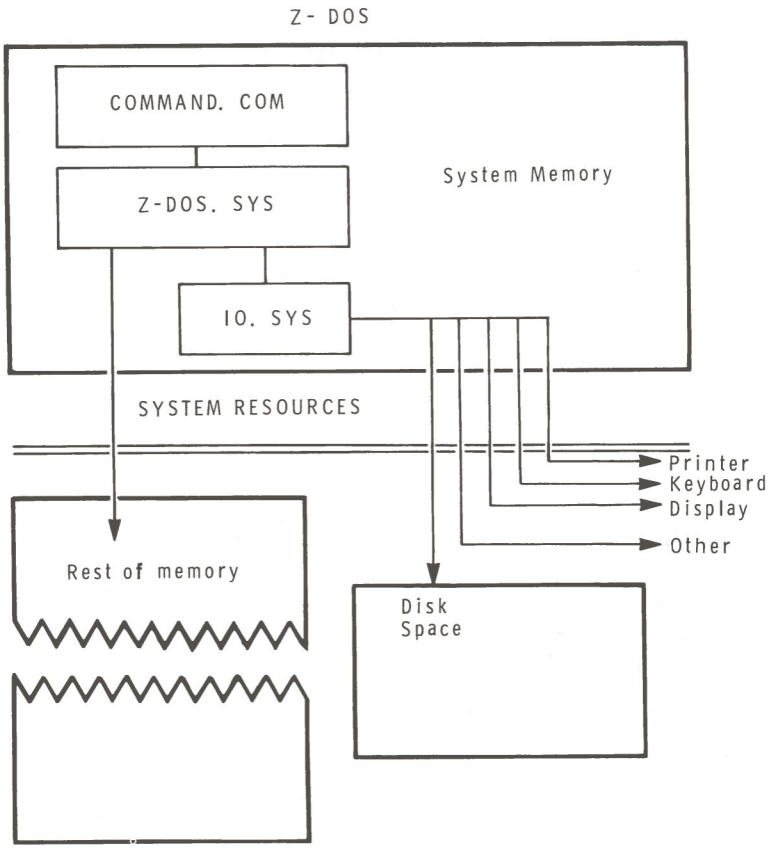


Figure 2.1
Z-DOS and Its Resources

Z-DOS AND THE OPERATING ENVIRONMENT

About Boot-up and System Initialization

What Part Does Each Component Play in Z-DOS?

Z-DOS performs the same tasks as the operating system presented as a model earlier in this manual. Z-DOS components relate to the model's components by function. IO.SYS is the I/O Manager. Z-DOS.SYS is the File Manager. COMMAND.COM is the Executive. The task of memory management performed in Z-DOS is simplified and does not require a component specifically for this task. The memory used by Z-DOS is handled by the individual components requiring the memory (generally Z-DOS.SYS and COMMAND.COM).

System Resources

Brief

System resources consist of:

- Disk Space
- Memory
- Peripheral Devices (printer, monitor, modem, etc.)
- Terminals

Details

System resources include peripheral devices such as terminals, printers, and serial lines. However, a system's most important system resources are its disk space and its memory.

Disk Space

In Z-DOS, disk space is divided into four parts:

Reserved Sectors

The reserved sectors contain information that is used each time Z-DOS is booted up—the Boot Loader. The technical details on reserved sectors are in Appendix H.

Directory

The directory contains information about each file on a given disk. This information includes the file's complete filename, its size, and its time and date of last modification. Additional details on the directory are given under the DIR command on Page 6.51. The technical details on the directory structure are found in Appendix H.

FAT's

The File Allocation Table (FAT) contains location information for the data that is contained in each file on a given disk. Note that Z-DOS does not require a file's contents to reside in physically contiguous disk sectors. The technical details on FAT are found in Appendix H.

Files

The greatest majority of disk space is reserved for the contents of files. An individual file does not necessarily reside in contiguous sectors on disk, and may be "scattered" so that all sectors may be used. Files are covered further on Page 3.1. The technical details on file deblocking are in Appendix H.

Memory

Besides controlling a system's disk space and its other devices, Z-DOS must also control main memory. This means that Z-DOS must be capable of loading files into memory either as data files or as files that are to be executed.

Files Loaded by IO.SYS

The actual loading of files is performed by IO.SYS, the lowest level of the Z-DOS operating system. Loading of executable files is supervised by COMMAND.COM. For most well-designed programs, control is returned to Z-DOS after either normal or abnormal termination of a program.

COMMAND.COM Overlaid

Part of COMMAND.COM may be overlaid to make room for a particularly large executable file. After execution of such a file, Z-DOS automatically loads the overlaid part of COMMAND.COM back into system memory. Normal execution of COMMAND.COM resumes.

If the overlaid part of COMMAND.COM is not available on disk because the disk on which it resides has been removed, the following message appears:

```
Insert system disk in default drive
and strike any key when ready
```

Also, if an incorrect version of COMMAND.COM is found, a similar message appears:

```
Invalid COMMAND.COM
Insert system disk in default drive
and strike any key when ready
```

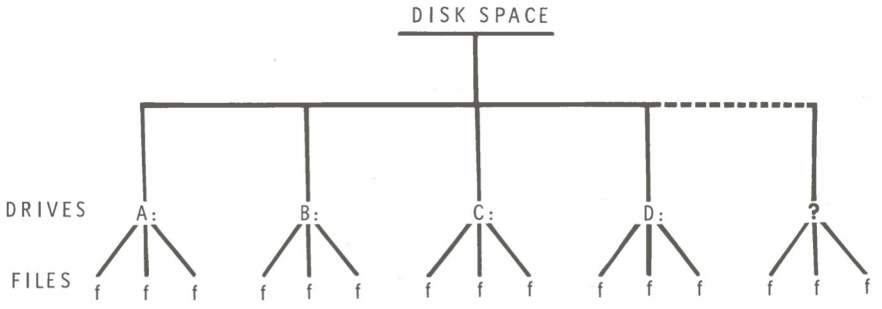
File System

The preceding discussion of system resources explored many of the internal aspects of the operating system. A file system, on the other hand, can be thought of as the external organization of system resources. It provides a way of talking about files and devices.

Device Independent Input/Output

Z-DOS supports device independent I/O. The distinction between files and devices is an internal distinction, not an external one. Therefore, you can treat files and devices alike, and refer to either with filenames. This is covered under the heading "Z-DOS Conventions."

However, disk space is slightly different because it is divided into drives. The disk space in a drive and on a particular disk is further subdivided into files, as shown below:



Disk Space Divisions

Drives are named so that up to 15 different drives can be referenced, though only four names are currently supported. Drives are named with the letters A through O. Each letter is followed by a colon (:). This colon separates the name of the disk from the names of individual files on the disk. This letter-colon combination is called a drive designation. Drives and drive names are discussed further on Page 3.10.

Application

Your Z-100 system has the capability to expand to meet your growth requirements. The Z-DOS operating system allows, and the hardware supports, a multitude of alternatives that you may select as the need arises.

Files and Filenames

Brief

Filename Format: [d:]<filename>[.ext]

where braces ([ ]) mean "optional"; angle brackets (< >) mean "required"; "d:" is a single-character disk drive name (A through D) followed by a colon (:); "filename" is the file's primary name, consisting of up to eight (8) characters; a period (.) is a required separator; and "ext" is the files extension, consisting of up to three (3) characters

Illegal Z-DOS Characters are:

? . , ; : = * { } / + "
(as well as non-printing characters or special function characters like backspace, escape or delete)

Legal Z-DOS Characters:

A-Z	0-9	$	&	#	@	!	_
%	'	(	)	-	<	>	
\	^	[	]	~	¦	`	

Details

Files

The Z-DOS operating system enables you to create, analyze, and manipulate information by storing this data in units called files. These files are stored on the surface of a disk and given a name that conforms to the Z-DOS file naming conventions.

Filenames

Z-DOS File Naming Conventions

A filename consists of two parts: the primary name and the extension. The primary name and the extension are separated by a period (.), in the following form:

```
[primaryname].[extension]
```

Primary Name

The primary name consists of one to eight characters, and is required for all files names. Throughout the rest of this text, when filename appears in a filespec (see below), it will be in reference to the primary name of a file. Filename is eight characters long. This will remind you that a primary name may only be eight characters in length.

Z-DOS supports a three-part name for files (and devices) called a file specification. A full file specification contains a drive designation, a filename, and a filename extension.

Extension

The extension is between one and three characters long, or is omitted entirely. Throughout the rest of this text, "ext" appearing in a filespec refers to the extension of a filename. Ext is three characters long to remind you of the maximum length of an extension.

Using "filename" for the primary name and "ext" for the extension, the filespec given above is:

```
[filename].[ext]
```

File Specification

The format of a file specification is:

```
[<d>:]<filename>[.<ext>]
```

The parts of a file specification are described below:

<d>: is the drive designation as described in the preceding section.

<filename> is a name consisting of from one to eight characters. Internally, all filenames are exactly eight characters. Filenames with seven or fewer characters have all remaining characters padded with spaces. Z-DOS performs this padding automatically.

Legal characters in filenames are:

```
A-Z  0-9  $  &  #  @  !  __
%    '    (  )  -  <  >
\    ^    [  ]  ~  ¦  `
```

Lowercase characters are converted to uppercase wherever they occur in a file specification. This means that a filename such as FiLe.ExT is converted to the filename FILE.EXT.

<ext> is the filename extension consisting of three or fewer characters. Padding of spaces occurs as described above for filenames. Legal characters are the same as for filenames. All characters are legal in any position in the extension, and all lowercase characters are converted to uppercase characters. Filenames with no extensions can also be specified by typing a period after the filename. Usually it is only necessary to use this form when a given command requires a default extension.

Illegal Characters

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

```
? . , ; : = * { } / + "
```

(Other illegal characters include non-printing characters or special function characters like backspace, escape or delete).

Z-DOS CONVENTIONS

Files and Filenames

The following filename examples all conform to these conventions:

FORMAT.COM	memo.doc	IO.SYS	4/27/81.TXT
JOB3.HEX	FILE#1	33%-RATE.DAT	ZMSGS.OVR

NOTE: "memo.doc" as shown above, is a legal filename but it is converted into "MEMO.DOC" by the operating system.

Although the extension is optional, it is useful to give your files extensions that describe the file's type or purpose or that conform to established conventions.

The following list contains conventional extensions that are applied to files:

Common Extension Use

EXTENSION	FILE PURPOSE
ASM	Assembler source file
BAK	Backup file from prior edit
BAS	BASIC source file
BAT	Batch File
BIN	Binary file
COM	Command file (binary file executable under Z-DOS)
CRF	Compact cross-reference file generated by MACRO-86
DAT	ASCII data file (or MAKE batch file)
DIF	Difference file created by FILCOM
DOC	ASCII document file
EXE	Executable binary file that relocates when loaded
FOR	FORTRAN source file
HEX	Intel HEX object file
INT	Intermediate BASIC file (used with BASIC-E or CBASIC)
LIB	Library source file that can be inserted during editing
LST	ASCII list produced by MACRO-86 Assembler
MAP	ASCII file of a Linked Module
OBJ	Object Code from Compiler or Assembler
PRN	Print file of assembly listing
REF	Expanded cross-reference file generated by CREF-86
REL	Relocatable Assembler Code
TMP	Temporary File (VM.TMP) created by LINK
$$$	Temporary work file

Device Filenames

Certain 3-letter filenames are reserved for the names of devices. These names are listed below:

Logical Device Names

AUX Used when referring to input from or output to an auxiliary device.

CON Used when referring to either keyboard input or to output to the terminal screen.

LST or PRN Used when referring to the line printer.

NUL Used when you do not want to create a particular file, but the syntax of a command requires an input or output filename. This is the "null device".

Even if given drive designations or extensions (<d:><device>.<ext>), these filenames remain associated with the devices listed above.

Thus, A:CON.LST still refers to the terminal console and is not the name of a disk file. This device naming scheme permits treating devices as if they were files, and is a consequence of Z-DOS's device independent I/O.

Application

Whether you are giving a command to Z-DOS or deciding on a method to name all of your files, filenames are at the core. If you were going to keep all of your correspondence filed on disks rather than keep file folders with copies of each letter, it would be a bad idea to name all of the files LETTER.DOC. (Note that no two files on the same disk may have the same name. The file created last would overwrite a previous file which had that name.) It would be hard to remember which letter was on what disk.

The briefness of a filename disguises its versatility. You might use the extension .TXT for all correspondence, and use an eight letter (or less) client name for the filename. All of your correspondence to Zenith Data Systems could be labeled ZENITH.TXT or ZDS.TXT. Z-DOS stamps the date on each file automatically so that you know when a file was created. This gives you a correspondence filing system by date and company (or recipient).

Z-DOS CONVENTIONS

Referencing Several Files at One Time

Brief

Wild card characters are used to reference more than one file at a time.

An asterisk (*) substitutes for the primary name or the extension. Only two asterisks (*.*) are needed to reference a complete filespec.

A question mark (?) substitutes for any character in the primary name or extension. Up to eleven question marks (????????.???) may be used to reference a filespec, though ????????.??? is equivalent to *.*.

Details

Wild Cards

Wild Card Filenames

Many of the commands that you issue refer to files by name. When you want to issue the same command for several files with similar names, it is more convenient to enter a "wild card filename" with the command.

Wild Card Characters

There are two wild card characters that can be used in file specifications: the asterisk (*) and the question mark (?).

By using these characters, a shorthand notation is created for specifying multiple files. This notation is particularly useful when file specifications are required as parameters for commands. Z-DOS makes full use of this capability for commands such as DIR, DEL, and COPY.

A wild card filename represents several filenames— much like a "joker" in a deck of cards can stand for any card in the deck. (Wild card filenames are sometimes called "ambiguous" or "global" filenames.) A wild card filename contains asterisks (*), question marks (?), or both.

Asterisks

Asterisk Wild Card

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

```
16MAY82.*
```

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

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

```
*.COM
```

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

The wild card "*.*" stands for *any* file on the disk.

Question Marks

Question Mark Wild Cards

Question marks (?) are used in a wild card file name to replace single characters at fixed file name positions. For instance, the wild card

```
JOB?.HEX
```

refers to any file with the extension "HEX", a primary name with the characters "J", "O", "B", and one or fewer additional characters. The files "JOB0.HEX", "JOB1.HEX", "JOBY.HEX", and "JOB.HEX" are just a few of the files that are referenced by such a wild card (if these files existed on the disk).

Application

Applications for wild cards are best shown by examples. Examine the following command:

```
DEL AB?DE.EXT
```

This command deletes all files whose names begin with AB, end with DE, have no more than one character between AB and DE, and have the filename extension .EXT. For example, Z-DOS might delete the following files:

```
ABCDE.EXT
ABODE.EXT
ABIDE.EXT
```

You can use up to eleven question marks in a wild card filename (eight in the primary name, three in the extension). The actual filenames referenced are those with the same characters as those that are explicitly stated in the wild card plus any characters in the place of the question mark characters.

The following are equivalent:

```
DEL *.*
DEL ????????.???
```

The asterisk is easier to type than a series of question marks.

The command DEL *.* deletes all files stored on the disk in the default drive, regardless of their filenames or extensions.

The wildcard "????????.COM" refers to any file on the disk with the "COM" extension—just like the "*.COM" wild card.

Here are some other examples, with their equivalents in question mark characters:

```
DEL FILE.*
DEL FILE.???
```

(Deletes all variations of FILE regardless of extension)

```
TYPE *.EXT
TYPE ????????.EXT
```

(Displays all files with the filename extension .EXT)

Finally, in this example:

```
TYPE ABC*.E*
TYPE ABC?????.E??
```

(Displays all files whose names begin with ABC and that have a filename extension that begins with .E)

Z-DOS CONVENTIONS

Disk Drives

Brief

Legal Drive Names: A B C D ... O

Supported Drive Names: A B C D

After a boot, the booted drive is the default drive. All requests assume the default drive unless otherwise specified. To assign another drive as the default drive from the "A:" prompt:

A: **d: RETURN**

where "A:" is the current drive's prompt; "d:" is the name of the drive to log-in; and RETURN is a carriage return.

The new drive (shown as "d:") then becomes the default drive and the system prompt is for example, "D:".

To request a program from a non-default drive:

A: **d:filename[.ext] RETURN**

where "A:" is the current (default) drive's prompt; "d:" is the drive name where the program resides; "filename" is the primary name of the file; "[.ext]" is a "BAT", "COM" or "EXE" extension; and RETURN is a carriage return.

Details

Drive Names

A disk drive is a device that copies data to and from disk storage media. These data transfers occur between memory and the disk. Because of the Z-DOS device-independent input/output handling, memory may seem transparent (not obviously apparent, nor present and involved) during some types of transfers. What this means is that you can type in data on your keyboard and it may be directed to a disk, although it actually does go from keyboard to memory and then to disk.

To allow you to refer to disks and files within your disk drives, the Z-DOS operating system recognizes each drive in your hardware environment by a distinct drive name. Z-DOS allows a drive name to consist of a letter of the alphabet from "A" to "O", and a colon (:).

Currently, the supported drive names are "A", "B", "C" and "D". "A" and "B" refer to your system's 5.25-inch disk drives. "C" and "D" refer to your system's 8-inch disk drives (providing you have the corresponding hardware connected). These name assignments can be changed by the MAP utility included in your Z-DOS software (described on Page 6.94). The utility to rename your drives is included for those special instances when renaming may be desirable.

The Default Drive

The drive that Z-DOS is copied from is the initial "default drive". When you boot without specifying an alternate drive, the `A:` system prompt that your console displays, confirms that Z-DOS is in the computer. If you boot from another drive, the default drive's prompt <`d:`> is displayed. You can always be certain which drive is your default drive because the system prompt shows the name of the current default drive. The default drive is the drive to which the system will refer, unless you tell the system to refer to a different drive.

Z-DOS CONVENTIONS

Disk Drives

Executing Program Files

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 "BAT", "COM" or "EXE" extension) in response to the system prompt, as:

A:<**filename**> **RETURN**

where "A:" is the default drive prompt (or system prompt); and <filename> 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 ".BAT", ".COM" or ".EXE" extension. If three files on your disk have the same primary name but one has a ".BAT" extension, one a ".COM" extension, and the third an ".EXE" extension, the <filename>.COM would be executed. COMMAND.COM searches for <filename>.COM first. If it does not find it, it will look for <filename>.EXE. If neither of these files exist, COMMAND.COM looks for <filename>.BAT. If none are found, you will see the "Bad command or filename" message.

If the application program file <filename> does not reside on default drive A, the system displays an error message, such as:

```
A:<filename>
Bad command or filename
```

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

Logging-in Different Drives

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:

A: **B: RETURN**

This entry produces a new system prompt, indicating that drive B: is now the default drive, as:

A: **B: RETURN**
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 disk. However, the default drive disk does *not* have to contain a copy of the Z-DOS Operating System, because the system already resides in computer memory.

Experiment by switching default drives by entering drive names (with a carriage return) one-at-a-time at the system prompt. When you switch default drives in this fashion, Z-DOS assumes that a program you request is on the disk in the new default drive.

Logging-in Another Drive

A "non-default drive" is a valid 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 of the drives with names "B" through "D".

Z-DOS CONVENTIONS

Disk Drives

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 entering the program's primary name and a carriage return:

A: **B:<filename> RETURN**

where "A:" is the default drive prompt (or system prompt); "B" is the name of the desired non-default drive; and <filename> 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 non-default drive B, and have a "COM", "BAT", or "EXE" extension.

The Z-DOS system responds to such an entry by "logging-in" disk B to get the application program indicated by <filename>, copying an image of this program into computer memory, and executing the program.

Application

If you plan to do a lot of work from one drive, changing the default to that drive makes working with it a little easier. If all of the programs that you are going to use are on drive D, for instance, you should log that drive (type **D:** and then **RETURN** in response to the system prompt). You can request commands by filename without needing to type in the drive name for each one. This means that at least one drive in your system does not have to be called by name each time you use it. The system assumes you mean the default drive when you leave out a drive specification.

Commands and Command Lines

Brief

Z-DOS commands are of two types (distinguished by their location) — "System Resident" and "File Resident."

System resident commands are calls to routines built into Z-DOS and are resident in memory after boot.

File resident commands are calls for utilities that reside in a disk file and are not loaded to memory until requested.

Details

Commands

In general, a command is a program that can help you 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 colon ":" character. When you start up Z-DOS, the system prompt is displayed on your console, as shown:

```
A:
```

The system prompt tells you that Z-DOS is ready to receive a command in the form of a "command line".

COMMANDING Z-DOS

Commands and Command Lines

Command Lines

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", and the "carriage return".

Function

The function is entered first, and it indicates the activity that is performed. Frequently, the function is a filespec ([<d:>]<filename>).

Argument

The argument is entered one space after the function. The argument indicates what data (files, systems, disks, drives, etc.) the function's activity involves. Occasionally, the argument is a filespec ([<d:>]<filename> [< .ext>]).

Execute Command

After you enter the function and argument, enter a carriage return to tell Z-DOS that the entire command line is ready for execution.

Command Location

There are two kinds of commands that execute in a Z-DOS operating environment: "System Resident" commands and "File Resident" commands.

The basic difference between these two commands is their location in Z-DOS.

System Resident Commands

System Resident commands are requests for functions that are built into the operating system. These commands are available anytime you use Z-DOS because they are loaded into memory during the system initialization. Since these commands are a part of the operating system they are sometimes referred to as "resident", "built-in" or "internal" commands.

System Resident commands are incorporated in COMMAND.COM and are always available when COMMAND.COM is resident in memory. Unlike the File Resident commands discussed next, these commands need not be available on disk when they are executed. Note that most System Resident commands are simple and easy to use. This is in contrast to some of the File Resident commands which are larger and more complex.

The system resident commands are listed below. Each is described thoroughly in the reference section, beginning on Page 6.1.

COPY	REM
DATE	REN
DEL	RENAME
DIR	TIME
ERASE	TYPE
PAUSE	

File Resident Commands

File Resident commands are requests for functions that are performed by one of the utility programs. These commands are available anytime one of your disk drives has a disk that contains that particular utility in a file. Since these commands are not part of the operating system they are sometimes known as "transient" or "external" commands.

Any file with the filename extension .COM or .EXE is considered a valid File Resident command. Such commands are executed by entering the name of the file without its .COM or .EXE extension. Programs that you create with most languages will be .EXE files.

File resident commands include:

CHKDSK	FILCOM
COMMAND	FORMAT
CONFIGUR	LIB
CREF	LINK
DEBUG	MAKE
DSKCOMP	MAP
DSKCOPY	MASM
EDLIN	PRINT
EXE2BIN	RDCPM
	SYS

NOTE: The .EXE files created with MASM (the MACRO–86 assembler) can be converted to .COM files with the command EXE2BIN.EXE. The format of a .COM file is special, so .EXE files cannot be arbitrarily converted. Also, all .COM commands execute in less than 64K of memory; .EXE files, on the other hand, may require more than 64K of memory to execute.

Throughout the rest of this manual the terms "System" and "File" describe the location of these two types of commands.

In the reference section of this Manual, you see:

Command
Location: System

for the System Resident type of commands, and

Command
Location: File

for the File Resident type of commands.

Application

Understanding this concept of commands and command lines (covered in the following section) is of major importance for using an operating system. Next most important is knowing when to use each command (or, knowing what each command does for you).

Although you should thoroughly understand the commands that you use, you may find that you will never use the more complicated commands such as DEBUG or MASM.

The Reason Command Location is Important

The location (type) of a command is important when you begin to use your Computer system more thoroughly.

Disks for different applications (word processing, general ledger, a computation spread sheet, etc.) could contain some of the operating systems commands. There are two File commands that are useful for your specialized disks; one that prepares a new disk for use and makes it bootable (FORMAT), and one that checks to see if two disks contain exactly the same information (DSKCOMP). To use FORMAT or DSKCOMP, they must be on one of the disks in your system. COPY and ERASE, on the other hand, are system commands and are always present after you boot Z-DOS.

COMMANDING Z-DOS

Commands and Command Lines

To use a command that invokes a system resident routine, you don't need to do anything special. The command is present and waiting for your request. However, to use a command that invokes a file resident routine, a disk that contains that command must be in one of your disk drives. Knowing where these commands are will help you to determine what you need to do to invoke them.

Command Line Entry

Brief

Command Entry Format:

A: **[<d:>]<command> [<argument1>,[<argument2>...]] RETURN**

where "A:" is the current (default) drive's prompt; [<d:>] is the drive name of a file resident program, if the program that performs the command is resident on a non-default drive; <command> is the name of a system resident routine or the primary name of a file resident utility; [<argument1>, [<argument2>...]] is a list of arguments (if required); and **RETURN** is a carriage return.

Details

Command Types by Extension

COMMAND.COM allows you to execute commands in four different groups. These are distinguished by their extension type (or lack of it).

1. System Resident commands such as DIR, REN, TYPE and DEL (no extension).

2. .COM commands such as CHKDSK.COM, DEBUG.COM, and EDLIN.COM (file resident).

3. .EXE commands such as LINK.EXE, MASM.EXE, and CREF.EXE (file resident).

4. .BAT command files that contain multiple instances of the above commands (file resident).

COMMANDING Z-DOS

Command Line Entry

User Interface

Z-DOS acts as an interface between you and the computer system's resources, with your communication normally occurring through keyboard input. This keyboard input is the raw data that is used to edit the Z-DOS command line. When editing is complete, the command line is passed on to COMMAND.COM where it is scanned for command names and parameters. Thus, the user interface consists of two levels of processing: command line editing and command interpretation. These levels are discussed in the next two sections.

COMMAND LINE EDITING

Command Line Buffers

Z-DOS offers a variety of functions that operate on the command line buffer. A buffer is a special area set aside in memory for a specific function and it is usually of a specific size. These functions make command line editing a simple and efficient task. The command line buffer is intimately related to another buffer called the "template," which is used in many of Z-DOS's special editing functions.

The model for command line input is as follows:

1. When you enter text from the keyboard, it is held in the command line until you press the **RETURN** key.

2. When you press **RETURN** the contents of the command line are sent to COMMAND.COM for processing.

3. When you press **RETURN** the line command is also copied to the template.

Thus, the template always contains the last entered command line.

Altering the Command Line

The command line is altered by entering one of the following kinds of input:

- Alphanumerics
- Punctuation
- Special editing functions
- Control character functions

Alphanumeric and punctuation characters are entered into the command line as they are typed. COMMAND.COM converts all lowercase letters to uppercase.

COMMAND INTERPRETATION

The Z-DOS user interface permits editing of command lines with the special editing and control character functions. Once a command line has been edited, it is sent to COMMAND.COM for processing. COMMAND.COM is the hub of the operating system, acting as the interface between the lower levels of the operating system and user input. It is in COMMAND.COM that the commands that are entered on the command line are interpreted.

COMMANDING Z-DOS

Command Line Entry

Commands themselves are either System Resident or File Resident.

System Resident commands are all resident in memory as part of COMMAND.COM. They are loaded into memory when the operating system is booted up. File Resident commands, on the other hand, are loaded into memory from the disk only when needed. File Resident commands reside in disk files and have a name with either a .COM or .EXE extension (a .BAT extension is also a designation of a special type of command; .BAT is covered briefly on Page 4.22 and in depth in the reference section on Page 6.14). COMMAND.COM itself is a File Resident command, by definition of its .COM extension.

A picture of the command interface is shown in Figure 4.1.

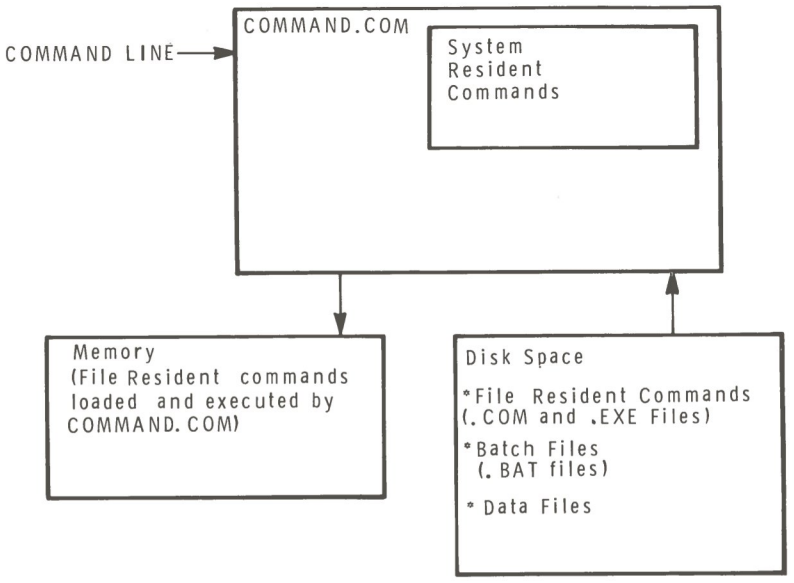


Figure 4.1
Z-DOS Command Structure

COMMAND.COM Scans for Command Names

Command interpretation begins when COMMAND.COM scans the command line for the name of a legal command. If it is a System Resident command, it is executed immediately; if it is a batch file, commands are executed indirectly from a .BAT file. If it is a File Resident .COM or .EXE command, the appropriate file is loaded from disk into memory where it is executed. (The batch facility and .BAT commands are discussed later in the chapter on Page 4.22).

NOTE: If more than one executable file exists on a disk with the same filename, COMMAND.COM interprets the invoked command on a priority based on the extension. .COM files have the highest priority. Then .EXE files, and finally .BAT. (For instance, if both TEST.COM and TEST.EXE are on the same disk, TEST.COM is executed by COMMAND.COM. If TEST.EXE and TEST.BAT are on the same disk, COMMAND.COM would execute TEST.EXE).

COMMAND.COM provides Z-DOS' characteristic colon prompt (d:) in the form of a drive designation letter and a colon (:).

For example:

```
A:_
```

The cursor is the focus of any editing actions that you perform. The cursor is indicated by an underline in this Manual; though the cursor may be an underline or a block on the Z-100.

At system start-up, the default prompt is always A:, assuming you booted from the default drive and did not select one of the other drives. After start-up, the user may change the default (that is, the currently selected drive). To select a new drive, simply enter the designation letter followed by a colon:

```
A:_                 (prompt; default drive)
A:B: RETURN         (user enters new drive designation)
B:_                 (new prompt; new default drive)
```

Note that COMMAND's main goal is to identify commands and then execute them. All commands consist of a command name followed by optional parameters. When parameters are present, they must be separated from the command name and from each other. Spaces, tabs, and commas are the only legal separators; though MASM, LINK, CREF, LIB, and FILCOM accept only commas.

For example:

```
COPY  OLDFILE.REL,NEWFILE.REL.
           ↑
(comma separator)

RENAME  THISFILE  THATFILE
              ↑
(space or tab separator)
```

Command Entry

The Z-DOS Operating System is selective when it comes to accepting command lines. Spell all components of a command line correctly and include the names of non- default disks whenever a referenced file is not on the default disk. If you don't, Z-DOS is not able to execute your command. Z-DOS responds to an error such as this with "Bad command or file name".

Entering Commands

Enter command lines in the following form:

A: <**function**> **[argument] RETURN**

where "A:" is the system prompt; <function> is required for all commands; [argument] is optional for some commands; and RETURN is required for all commands

NOTE: You must always separate the command line function and the command line argument with one space. Furthermore, any command entered at a prompt with the ":" character must end with a carriage return. However, commands themselves often display prompts in the course of their operation. When such a prompt (one given during the operation of a command) ends with a colon (:), a carriage return may not be required in your response.

Editing the Command Line

Z-DOS does aid you in its inflexibility at interpreting a command by providing a way to edit your response to the system prompt, as the following lists of features show.

These lists explain the single keys and combinations of keys that you can press to edit a command line into shape before submitting it to Z-DOS for execution.

NOTE: In this text, "CTRL" followed by a hyphen and a character indicates that you should press the key marked CTRL (control key) and the key marked with the specified character simultaneously.

The special editing and control character functions make Z-DOS very easy to use. All of the Z-DOS commands, from DEBUG to FILCOM, can make use of these functions wherever input is required from a terminal. These functions are always resident in the operating system.

COMMANDING Z-DOS

Command Line Entry

Special Editing Functions

Special Editing Functions — Because they depart from the "normal" way in which most operating systems handle command input, the special editing functions deserve particular emphasis. They relieve you of repeatedly typing in the same sequences of keys because the last command line entered is automatically "remembered" and placed in a "template".

Therefore, by using the template and executing the special editing functions, you receive the following advantages:

1. A command line can be instantly repeated in two keystrokes.

2. An erroneous command line can be edited and retried without reentering the entire command line.

3. A command line similar to a preceding command line can be edited and executed with a minimum of typing.

The relationship between the command line and the template is shown in Figure 4.2.

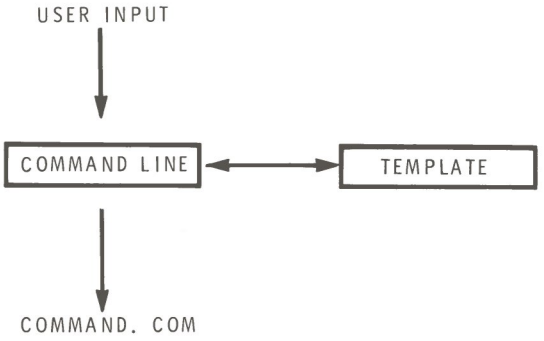


Figure 4.2
The Command Line and the Template.

Table 4.1, on Page 4.15, contains a complete list of the special editing commands. Each command is more fully described in Chapter Eight, "EDLIN," where these special editing commands become a subset of the commands available within the Z-DOS editor, EDLIN.COM, and are called the intraline editing commands.

Edit Keys

Note that the special editing commands are assigned to the function keys to make Z–DOS easier to use. Therefore, each command is identified by a function key name rather than by a specific command name.

Page 8.8 detailed description

Key	Editing Function
<F1>	Copy one character from the template to the command line.
<F2><c>	Copy all characters up to the character specified from the template to the command line. Where <c> is the specified character.
<F3>	Copy all remaining characters in the template to the command line. of entire line
<F4>	Skip over (do not copy) a character in the template.
<F5><c>	Skip over (do not copy) the characters in the template up to the character specified. Where <c>is the specified character.
<SHIFT-F0>	Void the current input; leave the template unchanged.
<F6>	Enter insert mode.
<F7>	Exit insert mode (toggle from insert); this is the default.
<F0>	Make the new line the new template

Table 4.1
Special Editing Functions

COMMANDING Z-DOS

Command Line Entry

As an example of the use of the special editing keys and command entry in general, pretend that you have entered the following command:

```
A: DIR PROG.COM      RETURN
```

This command displays the directory entry for the file PROG.COM on the terminal screen. It also has the useful side effect of saving the command line in the template. To repeat the command, all you have to do is type two keys: <**F3**> and **RETURN**.

The repeated command is displayed on the screen, as you type, as shown below:

```
A: <F3>DIR PROG.COM    RETURN
```

Pressing the **F3** key causes the contents of the template to be copied to the command line buffer; pressing **RETURN** causes the command line to be processed by COMMAND.COM.

Now pretend that you want to display the directory entry for the file named PROG with the .ASM extension. To do this, use the template, and type

```
<F2>C
```

Entering **F2 C** copies characters from the template to the command line buffer up to the character "C":

```
DIR PROG._
```

Note that the underline is your cursor. Now type:

```
ASM
```

The result is:

```
DIR PROG.ASM _
```

The desired command line is now in the command line buffer. To send this command line on to COMMAND.COM, the command interpreter, simply enter **RETURN**.

The template contains the following command line:

```
DIR PROG.ASM
```

Now pretend that you want to execute the following command:

```
TYPE PROG.ASM
```

To do this, type:

TYPE<F6> <F3> RETURN

Normal alphanumeric characters are entered directly into the command line buffer; they automatically replace corresponding characters in the template. This automatic replacement is turned off when the **F6** key is typed. Thus, the characters "TYPE" replace the characters "DIR" in the template. To insert a space between "TYPE" and PROG.ASM, you first typed **F6** and then a space. Finally, to copy the rest of the template to the command line, you typed **F3**, followed by a **RETURN**.

When you hit **RETURN**, the entire command line is copied to the template; in this case:

```
TYPE PROG.ASM
```

If you had misspelled "TYPE" as "BYTE", a command error would have occurred. Still, you could save the mistyped command line by creating a new template with the **F0** key:

BYTE PROG.ASM**<F0>**

You can then edit this erroneous command line by typing:

T<F1>P<F3>

The **F1** key copies a single character from the template to the command line buffer. The resulting command line then is what you want:

```
TYPE PROG.ASM
```

As an alternative, you could have used the same template containing BYTE PROG.ASM, and used the **F4** and **F6** commands to achieve the same result:

<F4><F4><F1><F6>YP<F3>

To illustrate how the command line buffer is affected as you type, examine the keys typed on the left and their effect on the contents of the command line buffer, shown on the right:

<F4>	_	(Skips over 1st template char)
<F4>	_	(Skips over 2nd template char)
<F1>	T_	(Copies 3rd template char)
<F6>YP	TYP_	(Inserts two characters)
<F3>	TYPE PROG.ASM _	(Copies rest of template)

Note that F4, like F5, does not affect the command line buffer; rather, it affects the template by deleting the first character in the template. Similarly, F5 deletes characters in the template up to a given character (this given character is the next one typed).

As you can see from the above examples, these special editing function keys can add greatly to your effectiveness at the keyboard. The next section describes control character functions that complement the above functions.

Special Function Keys

Control Character Functions—While commands are being entered, Z-DOS recognizes seven control character functions. These control characters and the functions associated with them are shown in Table 4.2.

Control Character	Function
<CTRL-N>	Cancel echoing of output to line printer.
<CTRL-C>	Abort current command.
<CTRL-H>	Remove last character from command line, and erase character from terminal screen.
<CTRL-J>	Insert physical end-of-line, but do not empty command line. Use <CTRL-J> (or linefeed) to extend the current logical line beyond the physical limits of one terminal line.
<CTRL-P>	Echo terminal output to the line printer.
<CTRL-S>	Suspend display of output to terminal screen. Press any key to resume.
<CTRL-X>	Cancel the current line, empty the command line, and then output a back slash (\), carriage return, and line feed. The template used by the Special Editing commands is not affected.

Table 4.2
Control Character Functions

<CTRL-I
<CTRL-M = RETURN

Application

The application of this section is easy. It makes working with the operating system as simple as possible. All of these editing features may be thought of as abbreviations of more complicated tasks. Learning these thoroughly saves you a lot of time and effort.

COMMAND.COM and Command Execution

Brief

COMMAND.COM handles interactive or batch commands

When COMMAND.COM is loaded during the bootstrap procedure:

it immediately looks for a file named AUTOEXEC.BAT (Automatic Execution batch file)

if the file is found, it is executed

if the file is not found, COMMAND.COM waits for interactive commands

---

Details

What does the Command Processor Do First?

COMMAND.COM takes control of the system at the end of the bootstrap procedure where it was copied into its assigned workspace in memory. COMMAND.COM immediately causes a search for a batch file called AUTOEXEC.BAT on the booted disk. If it does not find the file, it asks for the date and time and then waits for your command. If COMMAND.COM does locate AUTOEXEC.BAT, it automatically executes the command or commands that it finds in that file.

Processing Modes

Command processing on your Z-100 Computer can occur in one of two ways: "interactive" or "batch" mode.

Interactive Mode

An interactive processing mode waits for you to input a command and then immediately performs the task(s) required by the command.

COMMANDING Z-DOS

COMMAND.COM and Command Execution

Batch Mode

A batch processing mode allows you to enter any number of commands (up to the limitations of the system). It does not execute any of them until you tell it to do so, and then it executes them all in sequence.

Any of the Z-DOS commands may be used in either interactive or batch mode. If the commands are entered into a batch file (.BAT extension) they will be processed as a batch. If the commands are entered in response to the system prompt (d:, where "d" is a valid drive name), they will be processed interactively. However, you will find that two of the Z-DOS commands have a much greater use in the batch mode of operation — REM and PAUSE.

Batch Files

A "batch file" is a list of commands contained in a file that you may create any time you desire to do so. When you want the commands to be executed, you request Z-DOS to execute the batch file. If the batch file contains more than one command, COMMAND.COM executes them in the order that they are listed. When the first command (command1) is finished, COMMAND.COM begins the second listed command (command2) and continues in this manner for each command that is in the batch file.

Automatic Execution of a Batch File

AUTOEXEC.BAT is a special batch filename that is reserved for a specific use: automatic command execution when the bootstrap is completed. The ".BAT" extension informs COMMAND.COM that the file is a batch file, which contains a list of commands (or a command) to be executed. More information on the AUTOEXEC.BAT and batch files is contained in the reference section under ".BAT", which begins on page 6.14.

Processing Interactive Commands

Command entry by the user to the Z-DOS prompt ("d:", where "d" is any drive designation from A through H) is handled by the command processor (COMMAND.COM), which is loaded into memory during the bootstrap procedure. The COMMAND.COM program converts your command into the appropriate "system call", i.e., your command goes to COMMAND.COM and COMMAND.COM transfers control to the appropriate portion of Z–DOS where the requested task(s) are performed.

Interactive Mode/System Resident

If your command is "System Resident", control is immediately passed to the routine in the system that performs the requested task.

Interactive Mode/File Resident

If your command is "File Resident", control is temporarily passed to Z-DOS.SYS while the disk file is located and brought into a workspace in memory. Then, when the File Resident command is in the correct workspace, COMMAND.COM passes control to that program's starting point and the program performs the requested tasks.

Application

The most important reason to distinguish between the two modes of operation is the ease of working with this system to obtain your desired result.

Interactive mode makes it possible for you to immediately execute single commands or execute a constantly varying sequence of commands. If your computer use does not follow a set routine, or if you only need to execute one command, interactive processing provides you with fast interaction.

Batch mode makes it quicker for you to execute non- varying sequences of commands. By allowing you a method of defining the steps in the sequence once, you save time typing and retyping commands each time the sequence is executed. It also saves you from omitting something; once the sequence is defined correctly, it is always correct and waiting for you.

Care and Handling of Disks

Use and Care of Disks

Disks are fragile. They last a long time if they are handled properly. Careless treatment causes an irretrievable loss of data. Here are some guidelines:

Do's...

- When you load a disk, remove it from its protective sleeve and insert it according to your system's instructions. After the disk is fully inserted, gently close the compartment door.
- Always replace the disk into its protective sleeve.
- Magnetic fields near the disk can scramble data. Keep the disk away from objects that can cause such interference (the top of the video unit, an electrical motor, or any magnets or magnetized objects).
- Store the disks at a temperature of 10-52° Centigrade (40-125° Fahrenheit). When disks are stored for transportation or later use, put them in protective sleeves and set them upright in protective boxes.
- When you write on a disk label, use a felt or fiber-tip marker. Ballpoint pens or pencils can damage the disk.
- Decide on a standard place to put labels (usually the upper corner) and be consistent. Remove old labels.

Don'ts...

- Do not erase on the label. Rubbing damages the disk.
- Do not rubber-band or paperclip the disk.
- Never remove the disk from the disk drive while the red light on the drive is glowing.
- Do not leave the disk lying around. Dirt, dust or stains cause loss of data and disk errors.
- Do not expose the disk to sunlight or excessive heat.
- Never put your fingers or thumb on the disk through the slot that exposes the magnetic surface. The surface is easily contaminated, and this causes disk errors.
- Do not "fold, spindle or mutilate" the disk.
- Do not place heavy objects (such as books) on top of the disk.

Z-DOS System Startup

Brief

There are two methods that you can use to boot Z-DOS on the Z-100:

Method 1: Auto-Boot

Method 2: Manual Boot

Command Format:

<B>OOT[<dev>][<#>][<S>][:<boot string>] RETURN

where <B> is required input that the computer completes with "OOT"; <dev> is an optional function key (F1 through F8) that determines which device the controller is to use; <#> is the optional unit number of the device type connected to the device controller that is to be used; <S> is optional and specifies that the secondary device controller is to be used; <boot string> is an optional string of up to 79 characters, that is preceded by a colon; and RETURN is a required carriage return.

Details

Switch SW-101

The method you use to boot your Z-100 Computer is determined by a switch setting inside the Computer. The switch is numbered SW-101 in your hardware manual (see Figure 5.1). This switch selects the default boot and comes preset at the factory for Auto-Boot from default drive A.

To determine if your computer is set for Auto-Boot without checking Switch SW-101, turn the Power switch to ON when your Z-100 computer is fully connected to its components and to an appropriate power supply. If the A drive's active light turns on after a brief moment, the switch is set for Auto-Boot.

If this switch has not been changed, you can follow Method 1, beginning on Page 5.6, for a step-by-step procedure. If you turn the power on and no drive's active light comes on, you are set for manual boot—follow Method 2, beginning on Page 5.10, for a step-by-step procedure.

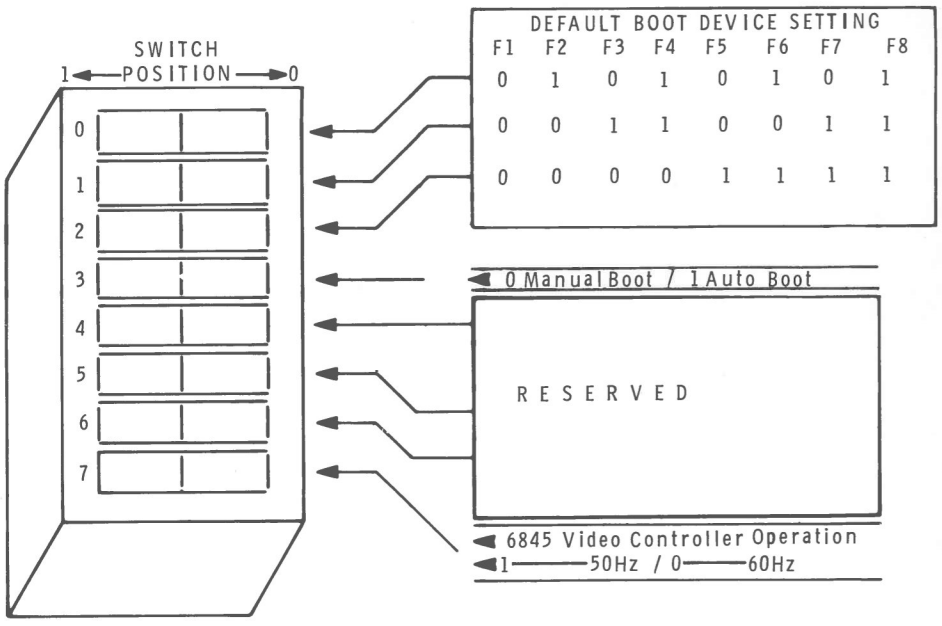


Figure 5.1

Manual/AutoBoot Setting (Switch SW-101)

Boot Command

The MTR-100 ROM Boot command is performed from the keyboard in response to the "pointing finger" prompt. There are several options available so that you can boot from any of the drives in your system.

The Boot command syntax is:

<**B**>OOT[<**dev**>][<**#**>][<**S**>][:<**boot string**>] **RETURN**

where <**B**> is required input that the computer completes with "OOT"; <**dev**>is an optional function key (F1 through F8) that determines which device is to be used. <**#**> is the unit number to be used; <**S**> is optional and specifies that the secondary device controller is to be used; <**boot string**> is an optional string of up to 79 characters that is preceded by a colon; and **RETURN** is a required carriage return.

Drive	Command Format	SW-101 (3, 2, 1 & 0)
A	Power On (Auto-Boot) **<B> RETURN** **<B><f1> RETURN** **<B><f1><0> RETURN**	1, 0, 0, 0 0, 0, 0, 0 0, x, x, x 0, x, x, x
B	**<B><f1><1> RETURN**	0, x, x, x
C	Power On (Auto-Boot) **<B> RETURN** **<B><f2> RETURN** **<B><f2><0> RETURN**	1, 0, 0, 1 0, 0, 0, 1 0, x, x, x 0, x, x, x
D	**<B><f2><1> RETURN**	0, x, x, x
NOTE: "x" in SW-101 switch setting indicates that position may be set to anything.		

Table 5.1

Possible Ways to Boot Drives A, B, C, and D

GETTING STARTED

Z-DOS System Startup

Method 1: Auto-Boot

Method 1 consists of:

- Securing the correct power and peripheral connections.
- Turning the power on and waiting for the drive active light.
- Placing the System Distribution disk in drive A.
- Waiting for the first system prompt.

Auto-Boot Procedure

After you unpack your Computer and connect any peripheral device to it according to the procedures given in the Z-100 Series User's Manual, proceed with the following steps.

☐ 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. Be 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 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. Place your Z-DOS System Distribution Disk Volume I into the A drive of your Z-100 and close the drive door.

☐ 8. You may hear the fan inside the computer start and, in a moment, the red drive-active indicator LED on your A drive will light.

Checkpoint

If a red LED does not light on your disk drive:

- Turn the Computer's POWER switch to the OFF position.
- Turn off the switches to any connected peripheral devices.
- Double check SW-101 to see if it is set for Auto-Boot, and for the appropriate device.
- Refer to the "In Case of Difficulty" section in your Z-100 Series User's Manual.

☐ 9. If the red LED does light, you should see the first Z-DOS prompt shortly. That system prompt is given in Step 10.

NOTE: If an AUTOEXEC.BAT file exists on the disk that you boot from, a system prompt may not display. Instead, the screen will display information that is controlled by the command in the batch file. If a "pointing finger" appears in the upper lefthand corner of the screen, read the following Checkpoint.

If you are using your original Z-DOS distribution disk during this procedure, date and time will be requested. The MAKE program will then begin and will announce itself. A stepwise procedure for this MAKE begins on Page 5.18.

Checkpoint

If after a minute a pointing finger appears in the upper lefthand corner of your monitor's screen, proceed to Method 2 on Page 5.10.

☐ 10. The screen should now display the message:

```
Z-DOS/MS-DOS BIOS release 1.00, version x.xx

Z-DOS/MS-DOS release 1.00, version x.xx
(C) Copyright 1982 Zenith Data Systems

Z-DOS/MS-DOS Command release 1.00, version x.xx
Current date is Tue 9-15-81
Enter new date:_
```

Checkpoint

If after a minute this message has not occurred, press **CTRL-RESET** (both the CTRL and RESET keys are located on the left of the keyboard).

A pointing finger should appear in the upper lefthand corner of your monitor.

- If the pointing finger appears, proceed with Method 2 on Page 5.11 from Step 8.
- If no pointing finger appears, refer to the section entitled "In Case of Difficulty" in the Z-100 Series User's Manual.

☐ 11. Type in the current date to the message shown on the screen. Do so in the format shown:

```
Current date is Tue 9-15-81
Enter new date <mm>-<dd>-<yy>  RETURN
```

where <mm> is a month in the range from 1 to 12; <dd> is a day in the range from 1 to 31 (depending on the month); and <yy> is a year in the range from 80-99 (or 1980 to 2099).

☐ 12. Next, you will see the following message:

```
Current time is 8:35:12.50
Enter new time:
```

☐ 13. Type in the current time to this message using the format:

```
Current time is 8:35:12.50
Enter new time: <hh>:<mm>[:<ss>]   RETURN
```

where <hh> is the current hour in the range 0-23; <mm> is the correct minute in the range 0-59; and <ss> is optional and accepts the current second in the range 0-59. If <SS> (seconds) is not given, it defaults to 00 seconds.

☐ 14. The "d:" prompt will now appear on your screen, where d: is the drive name (in the range from "A" through "D") of the boot drive that was assigned by SW-101 (see Figure 5.1 on Page 5.3 for brief details on SW-101 configuration). You have completed the boot. Now turn to Page 5.16.

Checkpoint

If the drive name does not appear at this point, or if the requests for system date and time did not appear, or if nothing appeared, press **CTRL-RESET** and go to Step 8 of Method 2 on Page 5.11.

Method 2: Manual Boot

Method 2 consists of:

- Securing the correct power and peripheral connections.
- Turning the power on.
- Placing the System Distribution disk in drive A.
- Waiting for the "pointing finger" monitor prompt.
- Performing the boot procedure on the selected device.
- Waiting for the system prompt.

Manual Boot Procedure

After you unpack your Computer and connect any peripheral device to it according to the procedures given in the Z-100 Series User's Manual, proceed with the following steps.

☐ 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. Be 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. In a moment, you should see a "pointing finger" (the monitor prompt) in the upper lefthand corner of your video terminal's screen.

Checkpoint

If you do not see the pointing finger, then you should:

- Turn the Computer's POWER switch to the OFF position.
- Turn off the switches to any connected peripheral devices.
- Refer to the "In Case of Difficulty" section in your Z-100 Series User's Manual.

☐ 8. Place your Z-DOS System Distribution Disk Volume I into the A drive of your Z-100 and close the drive door.

☐ 9. Type the letter **B**. The word "Boot" appears to the right of the finger.

☐ 10. Type <**F1**> to boot from 5-inch drives.

☐ 11. Type **0** to boot the A drive.

☐ 12. The following message should appear after a brief moment:

```
Z-DOS/MS-DOS BIOS Release 1.00, version x.xx

              elease 1.00, version x.xx
(C) Copyright 1982 Zenith Data Systems

Z-DOS/MS-DOS Command release 1.00, version x.xx
Current date is Tue 9-15-81
Enter new date:_
```

NOTE: If an AUTOEXEC.BAT file exists on the disk that you boot from, a system prompt may not display. Instead, the screen will display information that is controlled by the command in the batch file. If a "pointing finger" appears in the upper lefthand corner of the screen, read the following Checkpoint.

NOTE: If you are using your original Z-DOS Distribution Disk during this procedure, date and time will be requested. The MAKE Program will then begin and will announce itself. A stepwise procedure for this MAKE begins on Page 5.18.

Checkpoint

If after a minute this message or some other message has not occurred, press **CTRL-RESET** (both the CTRL and RESET keys are located on the left of the keyboard).

A pointing finger should appear in the upper lefthand corner of your monitor.

- If the pointing finger appears, attempt Method 2 again, beginning on Page 5.11 from Step 8.
- If no pointing finger appears, or if after attempting this procedure a second time with no success, refer to the section entitled "In Case of Difficulty" in the Z-100 Series User's Manual.

☐ 13. Type in the current date to this message. Do so in the format shown:

```
Current date is Tue 9-15-81
Enter new date: <mm>-<dd>-<yy>   RETURN
```

where <mm> is a month in the range from 1 to 12; <dd> is a day in the range from 1 to 31 (depending on the month); and <yy> is a year in the range from 80-99 (or 1980 to 2099)

☐ 14. Next, you will see the following message:

```
Current time is 8:35:12.50
Enter new time:
```

☐ 15. Type in the current time to this message using the format:

```
Current time is 8:35:12.50
Enter new time: <hh>:<mm>[:<ss>]  RETURN
```

where <hh> is the current hour in the range 0-23; <mm> is the correct minute in the range 0-59; and <ss> is optional and accepts the current second in the range 0-59. If <SS> (seconds) is not given, it defaults to 00 seconds.

☐ 16. The A: prompt should now appear on your screen, where A is the drive's name.

Checkpoint

If the drive name does not appear at this point, or if the requests for system date and time did not appear, and nothing else appeared, press **CTRL-RESET** and go back to Step 8 of Method 2 on Page 5.11 and try again. If, after a second attempt, the system still does not respond, consult "In Case of Difficulty" in your Z-100 Series User's Manual.

You have now completed the Boot procedure. Turn to Page 5-16 of this manual for important information about disk backups.

Boot Command's Logical Devices

To make the Boot command easier to work with, the Z-100 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 boot command is illustrated in Figures 5.2 and 5.3.

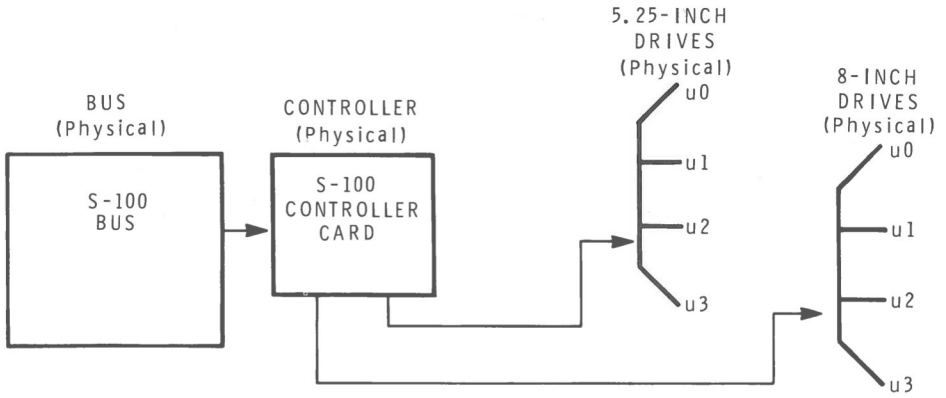


Figure 5.2

Physical Connections from the Hardware Viewpoint

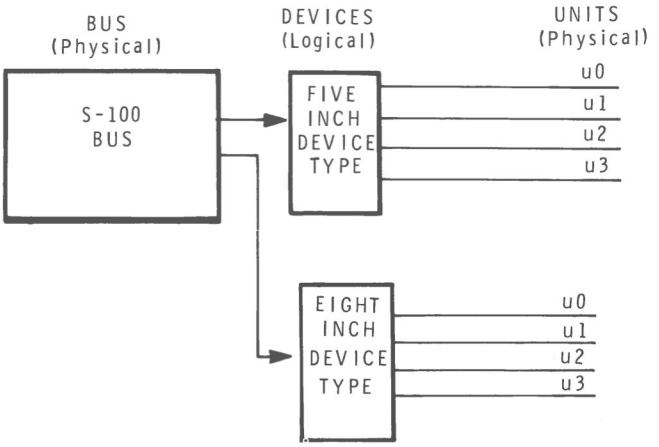


Figure 5.3

Logical Connections from the Boot Command Viewpoint

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 (see Figure 5.4 which illustrates the theory of such expansions).

The controller shown at the top of Figure 5.4 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 boot command syntax works for possible device expansion.

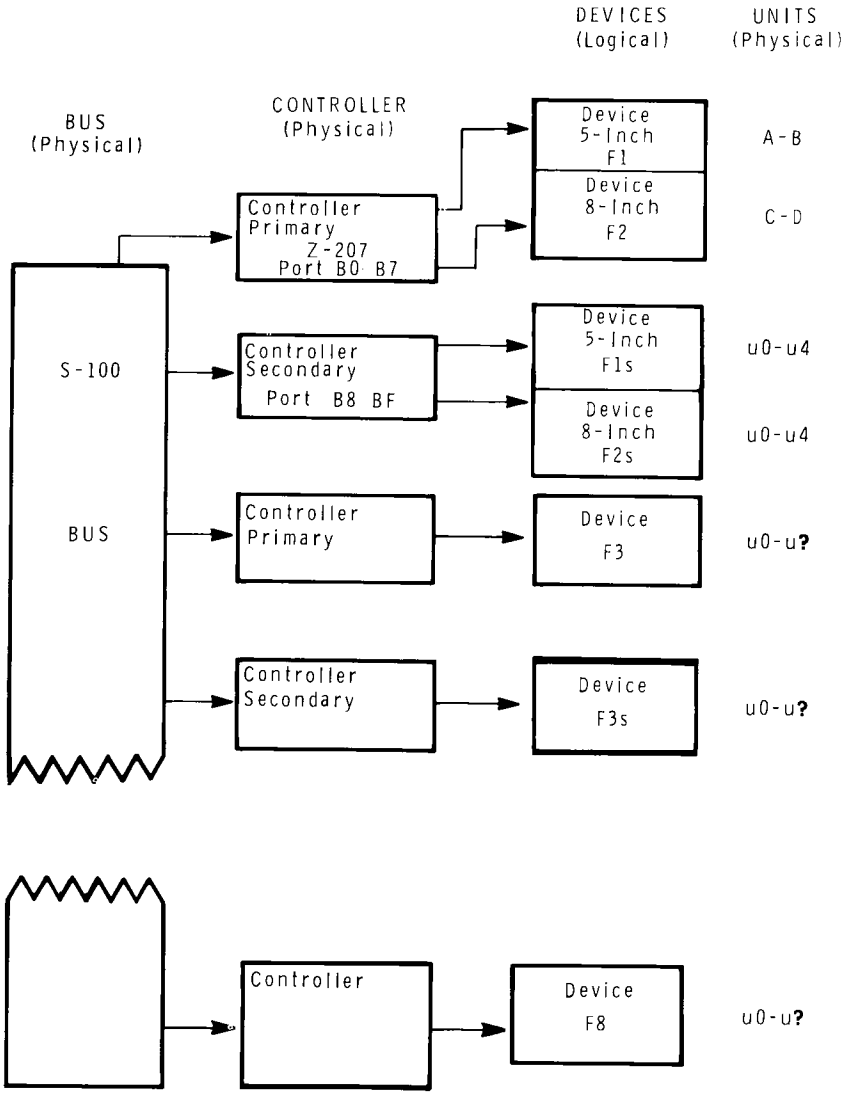


Figure 5.4

Logical Device Extensions With SW-101 DIP Switch

The Importance of Backup Disks

What is a backup disk?

A backup disk is simply an up-to-date duplicate copy of a disk that contains important data.

Why are backups inportant?

As you can tell from the instructions on the care of disks, there are numerous ways to destroy the data on a disk. If you damage a disk and have no back-up, all the information on that disk is gone and cannot be retrieved. This situation can be avoided if you make backup copies of all of your disks regularly.

What should I backup on Z-DOS?

The first backup you should make is a copy of the original Heath/Zenith Data Systems' Z-DOS disks (Disk I and Disk II). Store the master in a secure place such as a fireproof safe. You may wish to store it off the premises.

The second backup you should make is a copy of the Z-DOS System disk that you just copied. This is the disk you use daily and should be backed up because it may physically wear out.

Procedure For Backup

The specific procedure for making a backup copy of a disk depends upon your particular system. If you are not sure how to copy disks on your system, read the information on backup creation beginning on Page 5-18.

Here are some pointers on backing up disks:

- It is a good idea to use three disks for each disk you are backing up. Label one disk the "Working System Disk" and the other two disks "System Backup 1" and "System Backup 2". You should add volume numbers to these titles if they are backups of a specific distribution disk volume.

- If you change or create new programs on these disks, rotate the backup disks by taking turns using them to make new backup copies. Note on a log or on the labels which disk was used last so you know which contains the most up-to-date data on it.

- Place the backup disk in a secure location. Some people prefer to store backups in another physical location to prevent the possibility of loss in case of fire, flood, etc.

Application

Knowing why you should make backup copies of your disks should make you more conscientious about doing so on a regular basis. Backup copies can never be emphasized enough.

If you have a regular routine that you perform, such as updating accounts in a General Ledger software package, or writing correspondence with a word processing package, work backup creation into the routine. This is a good practice, and once backup creation is part of an established routine, you will seldom need to think about the backups until you need them.

GETTING STARTED

Create a Backup System Disk

Brief

There are three basic methods for creating backup system disks.

Method 1: MAKE Backup Procedure

A: **MAKE RETURN**

Method 2: Interactive Backup Procedure

A: **FORMAT/S/V B: RETURN**
A: **COPY*.* RETURN**

Method 3: Single Drive Backup Procedure

A: **DSKCOPY/V RETURN**

Details

Each of these three methods requires that the system setup procedure has been performed (that is, your Computer has been connected properly and plugged into a power source, turned on, and booted). Start-up is detailed beginning on Page 5.3.

Method 1: MAKE Backup Procedure provides you with a great deal of assistance in making a backup system disk. During this procedure, you will use the MAKE utility, which guides you through each step of the procedure with explicit screen displayed prompts. This method is designed for users with two physical disk drives that can accommodate the system disk.

Method 2: Interactive Backup Procedure provides you with a great deal of flexability in making a backup system disk. During this procedure you will enter commands at the Z-DOS system prompt, with the opportunity of specifying different file names and switches to vary the outcome of the procedure. This method is designed for users with two physical disk drives that can accommodate the system disk.

Method 3: Single Drive Backup Procedure provides you with a great deal of convenience in making a backup system disk. During this procedure, you will be prompted to alternately insert different disks into the same disk drive slot. This method is designed for users with only one physical disk drive that can accommodate the system disk.

Method 1: MAKE Backup Procedure

The first time that you boot your Z-DOS Distribution Disk I, an AUTOEXEC.BAT file is found by COMMAND.COM. The AUTOEXEC.BAT then begins to execute the commands that it finds in that file.

DATE AND TIME

The first two commands that COMMAND encounters in the batch file are the DATE and TIME commands (see Page 5.12 for details), which will ask you to enter the date and then the time.

MAKE Procedure

The next command in the batch file is MAKE SYSCOPY.DAT, which begins a MAKE batch file that guides you through making a System (bootable) backup of Distribution Disk I.

SYSCOPY.DAT

MAKE requests that you insert the different disks at the appropriate times. The SYSCOPY.DAT file is prepared so that MAKE can format your blank disk, copy the operating system to the disk, and then copy all of the utility files from your distribution disk to the new disk.

MAKE next requests a new blank disk and that Distribution Disk II be placed into the drives. (At this point, make a label for your new backup, like "Z-DOS SYSTEM DISK BACKUP", date it, and affix the label carefully.)

DATACOPY.DAT

MAKE uses the DATCOPY.DAT file on Distribution Disk II to guide you as it creates a backup copy of this disk. Distribution Disk II is a data disk and does not need to be bootable. All files from Distribution Disk II are copied to this blank disk.

MAKE next requests that you place Distribution Disk I and its backup (now labeled "Z-DOS SYSTEM DISK BACKUP", or whatever you decided) back into the disk drives. At this point, MAKE turns control back over to COMMAND.COM, and COMMAND goes to the next instruction in the AUTOEXEC.BAT file—ERASE B:AUTOEXEC.BAT. (Here, you should make a label for your new backup of Distribution Disk II, like "Z-DOS DATA DISK BACKUP", date it, and affix the label carefully.)

GETTING STARTED

Create a Backup System Disk

COMMAND erases AUTOEXEC.BAT from your backup copy so that you can use the backup disks that you have just made. If it did not erase the batch file from your backup, the backup would request that you make a backup of it when you booted with it on your computer. Place your distribution disks in a place for safe keeping.

If the backup you have just made gets damaged, you can boot your Computer on your Distribution Disk I and it will again instruct you to make a backup disk. The AUTOEXEC.BAT and MAKE batch files are still on the original Distribution Disk I.

Additional Backups

If you want to make more backup copies of Z-DOS, go on to Method 2 on Page 5.22. Where it asks that you place Distribution Disk I in a disk drive, use your Z-DOS System Disk Backup. Where it asks for Distribution Disk II, use your Z-DOS Data Disk Backup.

Foreign Language Character Fonts

NOTE: If you are going to be using a foreign language character font, you should read Appendix N. You will want to copy the correct foreign language character font file to ALTCHAR.SYS, which is the alternate character system file that IO.SYS looks for during initialization. To do this, your newly created backup of Distribution Disk I should be in drive B. Select the correct command line below, and enter it on the keyboard.

A: **COPY B:DANISH.CHR ALTCHAR.SYS RETURN**

A: **COPY B:ENGLISH.CHR ALTCHAR.SYS RETURN**

A: **COPY B:FRENCH.CHR ALTCHAR.SYS RETURN**

A: **COPY B:GERMAN.CHR ALTCHAR.SYS RETURN**

A: **COPY B:ITALIAN.CHR ALTCHAR.SYS RETURN**

A: **COPY B:SPANISH.CHR ALTCHAR.SYS RETURN**

A: **COPY B:SWEDISH.CHR ALTCHAR.SYS RETURN**

When you next boot this disk, the new character set will be in place. Copy the ALTCHAR.SET file to bootable disks when you create them with the MAKE command.

Method 2: Interactive Backup Procedure

To make backup copies of your disks without the MAKE program, you can use the following procedure after you have performed the setup procedure and booted your computer with a Z-DOS System Disk in drive A.

NOTE: Though the procedure below is designed for you to make copies of your Z-DOS distribution disks, the procedure is essentially the same for other disks. Just be certain to remember what disk you are to put into the drive in place of Distribution Disk I and Distribution Disk II.

Using FORMAT for Backups

☐ 1. With a system disk in drive A and a blank disk in drive B, type:

A: **FORMAT/S/V B: RETURN**

This command tells Z-DOS to format the disk in drive B. (More about FORMAT is covered on Page 6.62.) The /S is called a switch, and it requests FORMAT to copy the system files — IO.SYS, Z-DOS.SYS, ALTCHAR.SYS and COMMAND.COM—after formatting the disk. The /V switch requests FORMAT to verify that it has formatted the disk correctly, and that no mistakes have occurred in the process. The /V switch also isolates any bad sectors that occur on the disk.

☐ 2. When the command prompt (A:) reappears, remove your system disk and place your Z-DOS Distribution Disk I in drive A (the newly formatted disk stays in B).

☐ 3. Now type:

Using Copy for Backups

A: **COPY *.*B: RETURN**

This instructs Z-DOS to copy all files (*.*) from the current drive (A) to the disk in drive B.

☐ 4. As each file is copied, the name of the file will appear on your screen.

Create a Backup System Disk

☐ 5. When all the names have finished copying, the system prompt will reappear on your screen.

A:

☐ 6. Make a label for your newly created backup disk, and affix it carefully to the disk.

☐ 7. Now place Distribution Disk II in drive A and a different blank disk in drive B.

☐ 8. Now return to Step 1 on Page 5.21 and continue down through Step 6, where you should make a label for the Distribution Disk II backup.

Backup of Distribution Disk II

NOTE: If you are going to be using a foreign language character font, you should read Appendix N. You will want to rename the correct foreign language character font file to ALTCHAR.SYS, which is the alternate character system file that IO.SYS looks for during initialization. To do this, your newly created backup of Distribution Disk I should be in drive B. Select the correct command line below, and enter it on the keyboard:

A: **COPY B:DANISH.CHR ALTCHAR.SYS RETURN**

A: **COPY B:ENGLISH.CHR ALTCHAR.SYS RETURN**

A: **COPY B:FRENCH.CHR ALTCHAR.SYS RETURN**

A: **COPY B:GERMAN.CHR ALTCHAR.SYS RETURN**

A: **COPY B:ITALIAN.CHR ALTCHAR.SYS RETURN**

A: **COPY B:SPANISH.CHR ALTCHAR.SYS RETURN**

A: **COPY B:SWEDISH.CHR ALTCHAR.SYS RETURN**

When you next boot this disk, the new character set will be in place. Copy the ALTCHAR.SET file to bootable disks when you create them with the MAKE command.

Method 3: Single Drive Backup Procedure

The DSKCOPY activity helps you copy all of the software from a system disk (such as Z-DOS Distribution Disk I) to a blank floppy disk, which will become a Backup System Disk.

Before you begin this procedure, you should insert a system disk into the drive, boot up, enter the date, and enter the time.

The system disk you use should contain a copy of the file DSKCOPY.COM. This method is particularly useful in hardware environments where only a single physical drive can accommodate the system disk.

If you need any further information on using the DSKCOPY utility, follow the instructions in "Chapter 6: Z-DOS Commands." If you need any further information on using Z- DOS with a single drive, follow the instructions in "Appendix G: Instructions for Single Disk Drive Users".

☐ 1. If the MAKE utility is automatically invoked, you should enter **CTRL-C** at the `Do you wish to continue (Y/N) <Y>` prompt.

If the system prompt is displayed with no command line following it, then proceed to Step 3.

☐ 2. Press **Y** at the `Terminate batch job (Y/N)?` prompt.

☐ 3. Type the command **DSKCOPY/V RETURN** at the A: system prompt. This entry invokes DSKCOPY, which will display a message and prompt in the following form:

```
DSKCOPY version 1.02
Copyright(C) 1982 Zenith Data Systems

Source drive name? (A-F) _:
```

☐ 4. Type **A**. Then DSKCOPY will display the prompt:

```
Destination drive name? (A-F) _:
```

GETTING STARTED

Create a Backup System Disk

☐ 5. Type **B**. Then DSKCOPY will display the prompt:

```
Place source diskette in A: and destination diskette in B:
Hit RETURN when ready.
```

☐ 6. Leave the system disk in the drive, and press **RETURN**. Then DSKCOPY will display the prompt:

```
Formatting destination... Place disk B in drive A:.
Hit RETURN when ready.
```

☐ 7. Remove the system disk from the drive.

☐ 8. Label a blank floppy disk "Backup System Disk".

☐ 9. Insert the Backup System Disk in the floppy disk drive slot, close the drive latch, and press **RETURN**. The floppy disk drive light will glow for several seconds. Then DSKCOPY will display the following prompt:

```
Copying... Place disk A in drive A:.
Hit RETURN when ready.
```

☐ 10. Remove the Backup System Disk from the drive, insert the system disk, and press **RETURN**. DSKCOPY will continue to display prompts in the following form:

```
Place disk X in drive A:.
Hit RETURN when ready.
```

☐ 11. When a prompt in this form reads `Place disk B in drive A:`, insert the Backup System Disk and press **RETURN**.

When a prompt in this form reads `Place disk A in drive A:`, insert the system disk and press **RETURN**.

☐ 12. Continue switching the two disks as DSKCOPY displays a prompt in the following form:

```
Verifying... Place disk A in drive A:.
Hit RETURN when ready.
```

☐ 13. Continue switching the two disks until DSKCOPY displays the following prompt:

```
Copy another? (Y/N) <N>
```

☐ 14. Type **N** and press **RETURN**. Then the system prompt will be displayed again, as shown:

```
A:_
```

☐ 15. Store your system disk away in a safe place, and leave your Backup System Disk in the drive.

☐ 16. Type **DIR RETURN** at the system prompt. The following prompt will appear on the screen:

```
Place disk A in drive A:.
Hit RETURN when ready.
```

☐ 17. Press **RETURN** at the Place disk A prompt. (Leave your Backup System Disk in the drive.) Directory characteristics of the copied files will be displayed.

☐ 18. If you wish to copy another disk (such as Z-DOS Distribution Disk II), then repeat Step 3 through Step 7 using a system disk (such as Z-DOS Distribution Disk I). Then repeat Step 8 through Step 17 — substituting the disk you now wish to copy for the "system disk" and substituting an appropriately labelled backup disk for the "Backup System Disk".

If you do not wish to copy any more disks, then you have finished Method 3: Single Drive Backup Procedure.

GETTING STARTED

Create a Backup System Disk

NOTE: If any of your application programs require user entry of foreign language characters or display of such characters, then you should copy the appropriate foreign language character font to the file ALTCHAR.SYS on a Backup System Disk. This activity will cause the system from this disk to automatically load the foreign language character font into memory when you boot up.

Enter a COPY command specifying the name of the desired foreign language character font file, as shown in the following examples:

A:COPY B:DANISH.CHR ALTCHAR.SYS RETURN

A:COPY B:ENGLISH.CHR ALTCHAR.SYS RETURN

A:COPY B:FRENCH.CHR ALTCHAR.SYS RETURN

A:COPY B:GERMAN.CHR ALTCHAR.SYS RETURN

A:COPY B:ITALIAN.CHR ALTCHAR.SYS RETURN

A:COPY B:SPANISH.CHR ALTCHAR.SYS RETURN

A:COPY B:SWEDISH.CHR ALTCHAR.SYS RETURN

If the foreign language character font file (with the .CHR extension) is not on the disk on which you wish to change the character font, then specify the name of the drive that contains the font file.

If you have only one physical disk drive slot to accommodate the necessary disks, then Z-DOS will display prompts in the form `Place disk X in drive A:. Hit RETURN when ready.` Respond to such prompts by inserting the specified disks and pressing **RETURN**.

Refer to "Appendix G: Instructions for Single Disk Drive Users" and "Appendix N: Character Font Files" for further information.

Part 2

Z-DOS Reference Guide

Z-DOS Command Reference

This chapter contains reference material for all Z-DOS commands. You will find each of the commands listed in alphabetical order. The instructions (commands) that are required for operation by some Z- DOS utility are listed alphabetically under the utility's name.

Z-DOS COMMANDS

Commands in Brief

A table that shows the command name, the command's purpose, and the format(s) in which the command may be used is shown immediately below and continues to the next few pages. It will provide you with quick-access reference to the basic commands. Command names shown in italics are "system resident". All other command names are "file resident".

NAME	PURPOSE	COMMAND ENTRY FORMAT
.BAT	Process as a batch file	**[d:]<filename>[<parameters>...]**
CHKDSK	Compare directory to actual files and report status	**CHKDSK [d:]**
CONFIGUR	Configure the system for specific printers or modems	**CONFIGUR [?]**
COPY	Copy file(s) specified to specified location	**COPY[</x>]<filespec> [d:][<filespec>]**
CREF	Build cross reference of Macro Assembler Source	**CREF [<crffile>,<listing>]**
DATE	Show current system date and/or enter new date	**DATE [<mm>-<dd>-<yy>]**
DEBUG	Load, change, display the content, or control run any file	**DEBUG [<filespec>]**
DEL	Delete specified file(s) (same as ERASE)	**DEL [<filespec>]**
DIR	List directory or entries specified on every track	**DIR [<filespec>][</x>]**
DSKCOMP	Compare data on two disks to see if they are alike	**DSKCOMP [d:] [d:]**

Table 6.1

Z-DOS Command List (part 1)

NAME	PURPOSE	COMMAND ENTRY FORMAT
DSKCOPY	Duplicate source disk's data on destination disk.	**DSKCOPY[</x>] [d:] [d:]**
EDLIN	Create, edit, display or delete text (ASCII) files	**EDLIN <filespec>**
ERASE	Erases specified file(s) (same as DEL)	**ERASE <filespec>**
EXE2BIN	Convert executable files (.EXE) to binary files (.BIN or .COM)	**EXE2BIN <filespec> [d:] [<filename>][<.ext>]**
FILCOM	Compare files to see if they are identical	**FILCOM [<s1>[,<s2>][,<list>][</x>...]**
FORMAT	Initialize disk for Z-DOS and check for bad tracks	**FORMAT [d:][</x>...]**
LIB	Library management for Macro Assembler Modules	**LIB [<library><operations>,<list>]** **LIB @<filespec>**
LINK	Creates .EXE (executable) file from an object code file	**LINK [<filenames>[</x>]]** **LINK @<filespec>**
MAKE	Batch Command to aid in the installation of new software.	**MAKE [<filespec>] [d:]**
MAP	Physical to Logical Device mapping utility	**MAP [?][</x>][d: n]**

Table 6.1

Z-DOS Command List (part 2)

Z-DOS COMMANDS

Commands in Brief

NAME	PURPOSE	COMMAND ENTRY FORMAT
MASM	Creates object code from a source code file	**MASM [<filenames>[</x>...]]**
PAUSE	System pause with or w/o an optional remark; press any key to continue	**PAUSE [remark]**
PRINT	Prints ASCII files to the PRN: assigned device.	**PRINT <filespec>[</x>...]** **PRINT ?**
RDCPM	Copy CP/M to Z-DOS Format	**RDCPM <filespec>[d:][filename.ext]** **RDCPM DIR <filespec>** **RDCPM ?**
REM	Display remark during batch file execution	**REM [remark]**
REN	Name current file with new name (same as RENAME)	**REN [<filespec>][<filespec>]**
RENAME	Name current file with new name (same as REN)	**RENAME [<filespec>][<filespec>]**
SYS	Writes system files to a new disk	A:**SYS [d:]**
TIME	Display current system time and/or enter new time	A:**TIME [<hh>[:<mm>[:<ss>]]]**
TYPE	Display contents of file	A:**TYPE <filespec>**

Table 6.1

Z-DOS Command List (part 3)

List of Disk File Content and Command Summary

A brief description of the system files and the system utilities are given here:

Distribution Disk I Contents

ALTCHAR.SYS contains the H/Z-19 graphics characters (also in GRAPHICS.CHR). ALTCHR.SYS is read into memory during the boot procedure. (See Appendix N).

AUTOEXEC.BAT is the name of a redefinable, automatically executed batch file. When this file exists on a booted disk, its contents are executed immediately upon booting. The AUTOEXEC.BAT contained on your distribution disk has been defined to request the current date, time, and then execute the MAKE backup utility, and finally erase itself so that the batch file does not execute each time you boot your distribution disk. (See additional details under MAKE.COM, SYSCOPY.DAT and DATCOPY.DAT).

CHKDSK.COM is a command used to check and verify the contents of a disk. It is further described on Page 6.19.

COMMAND.COM is the command interpreter used to interface between the user and the underlying operating system. It allows you to perform file management functions such as rename and delete, as well as to load and execute programs. COMMAND.COM is described on Page 4.21.

CONFIGUR.COM is a utility to configure your system to use a line printer or modem. It informs the system what type of line printer or modem you are using, where they are located (port address) and the method of relaying information to the device (communications protocol). It is discussed on Page 6.22.

CREF.EXE is the Microsoft MS-CREF cross-reference utility used to create a cross-reference listing from an assembly source listing. CREF.EXE is further described on Page 6.39 and as a comprehensive utility in Chapter 13.

DANISH.CHR is an alternate set of Danish characters for the Z-100 character font. See Appendix N.

Z-DOS COMMANDS

Commands in Brief

DEBUG.COM is a debug program used to provide a controlled testing environment for executable object files. DEBUG.COM is described on Page 6.44 and in Chapter 7.

DSKCOMP.COM is a utility that you can use to verify that the contents of two disks are identical. DSKCOMP.COM is described on Page 6.53.

DSKCOPY.COM copies the content of one disk to another disk so that the content of the copy is identical to the content of the original. DSKCOPY.COM is described on Page 6.57.

EDLIN.COM is the Z-DOS line editor. Intraline editing is performed using the special editing keys that are also available at the Z-DOS command level. EDLIN.COM is described on Page 6.61 and in Chapter 8.

ENGLISH.CHR is an alternate set of English (British) characters for the Z-100 character font. See Appendix N.

EXE2BIN.COM is used to convert .EXE files to .COM files. In general, only assembly language programs that have been specially formulated may undergo such conversions. EXE2BIN.COM is described on Page 6.65.

GRAPHICS.CHR contains the H/Z-19 graphics characters and these may be copied to ALTCHAR.SYS (if no foreign language font is used) to be loaded during the boot procedure. (See Appendix N.)

FILCOM.COM is a file comparison program used to check for differences between files. Either text or binary files may be compared. FILCOM is described on Page 6.67 and in Chapter 9.

FORMAT.COM is used to format disks so that they can be used with Z-DOS. FORMAT.COM is described on Page 6.71.

FRENCH.CHR is an alternate set of French characters for the Z-100 character font. See Appendix N.

GERMAN.CHR is an alternate set of German characters for the Z-100 character font. See Appendix N.

ITALIAN.CHR is an alternate set of Italian characters for the Z-100 character font. See Appendix N.

IO.SYS is the lowest level of the Z-DOS operating system, interfacing to all I/O devices. It is an Z-DOS "hidden file" and does not show up when a directory command is executed. IO.SYS is automatically loaded into memory when your system is booted up. See Chapter Two for more information.

LIB.EXE is the Microsoft MS-LIB library manager used to create, maintain, and manipulate libraries of object files. LIB.EXE is described on Page 6.80 and in Chapter 12.

LINK.EXE is the Microsoft MS-LINK linker used to link object files and object libraries to create executable .EXE files. LINK.EXE is further described on Page 6.83 and in Chapter 11.

MAKE.COM is a utility designed for easy installation of new software. It allows you to create backup copies of software disks with a minimum amount of effort. MAKE is covered on Page 6.88.

MAP.COM allows you to change the device unit names from their assigned names to alternates. This allows you to work with an application program that requires specific drives for its operations, even if the drive that is necessary is not operating properly, by changing the logical name assignments. MAP is covered on Page 6.94.

PRINT.COM is a utility that prints ASCII files from disk to your lineprinter. PRINT.COM is further described on Page 6.104.

MASM.EXE is Microsoft's relocatable macro-assembler for 8086 and 8088 microprocessors, MACRO-86. MASM.EXE is further described on Page 6.99. The comprehensive details are contained in Chapter Ten.

SPANISH.CHR is an alternate set of Spanish characters for the Z-100 character font. See Appendix N.

SWEDISH.CHR is an alternate set of Swedish characters for the Z-100 character font. See Appendix N.

SYS.COM is used to recopy Z-DOS.SYS and IO.SYS from a system disk to a formatted disk that already contains the Z-DOS operating system on it. It is described on Page 6.112.

SYSCOPY.DAT is a MAKE batch file that contains the necessary commands for the MAKE utility to create a backup of your Distribution Disk I on initial boot. (See AUTOEXEC.BAT and DATCOPY.DAT for additional details.)

Z-DOS.SYS is the heart of the Z-DOS operating system, where most management of system resources takes place. Z-DOS.SYS is intimately tied to COMMAND.COM and IO.SYS. Note that Z-DOS.SYS is a Z-DOS "hidden file" and does not show up when a directory command is executed. Z-DOS.SYS is automatically loaded into memory when your system is booted up. For further information, see Chapter 2.

Distribution Disk II Contents

NOTE: The source files on this disk are used to recreate the BIOS.

BAUXIO.ASM contains the AUXIN and AUXOUT entry points. BAUXIO.ASM gets assembled as an include during assembly.

BCHR.ASM contains the include statements for the character device drivers.

BCHRIO.ASM is the source for the BIOS Function Calls for the CON, PRN and AUX devices. BCHRIO.ASM is assembled as an include during assembly.

BCLOCK.ASM contains the time and date entry points and the timer interrupt handler. BCLOCK.ASM is assembled as an include during assembly.

BCONIO.ASM contains the source for the keyboard interrupt handler and the entry points for the CON: device. BCONIO.ASM is assembled as an include during assembly.

BDOSTB.ASM is the source for the disk tables used by the DOS. BDOSTB.ASM is assembled as an include during assembly.

BDSK.ASM contains the include statements for the disk drivers.

BDSKIO.ASM is the BIOS disk functions source and includes disk read and disk write. BDSKIO.ASM is assembled as an include during assembly.

BDSKLA.ASM contains the BIOS disk map table and routines to check for a changed disk. BDSKLA.ASM is an include during assembly.

BDSKTB.ASM contains the tables of disk controller functions used by the disk drivers (DSK.ASM). BDSKTB.ASM is an include during assembly.

BEND.ASM contains a label that constitutes the end of the BIOS.

BINIT.ASM contains the system initialization code. BINIT.ASM is an include during assembly.

BMSDOS.ASM is the main source code given to the Macroassembler to assemble the BIOS. BMSDOS.ASM contains include directives for all of the other .ASM files.

BMSDOSL is a LINK response file that is used during creation of the BIOS.

BPRNIO.ASM contains the BIOS entry points for the PRN: device. BPRNIO.ASM is an include during assembly.

DATCOPY.DAT is a MAKE batch file that contains the instructions given to the MAKE utility to create a backup of Distribution Disk II on initial boot. (See AUTOEXEC.BAT and SYSCOPY.DAT for additional details.)

DEF6821.ASM contains the 6821 PIA definitions. DEF6821.ASM is an include during assembly.

DEF6845.ASM contains the 6845 CRT controller definitions.

DEF8253.ASM contains the 8253 Programmable Interval Timer definitions. DEF8253.ASM is an include during assembly.

DEF8259A.ASM contains the definitions for the 8259A Programmable Interrupt Controller. DEF8259A.ASM is an include during assembly.

Commands in Brief

DEFASCII.ASM contains the definitions for the ASCII control characters. DEFASCII.ASM is an include during assembly.

DEFCHR.ASM defines the character devices used by the AUX: and PRN: BIOS function calls. DEFCHR.ASM is an include during assembly. I. 39

DEFCONFG.ASM contains the Z-100 system configuration parameters. I. 44

DEFDSK.ASM contains the structure of commands used for the disk function BIOS calls. DEFDSK.ASM is an include during assembly. I. 35

DEFEP2.ASM defines the 2661-2 USART. DEFEP2.ASM is an include during assembly.

DEFFMT.ASM contains the loader and disk label information used by FORMAT.COM. DEFFMT.ASM is an include during assembly.

DEFIPAGE.ASM contains the offsets for the interrupt vectors. DEFIPAGE.ASM is an include during assembly. APP I, 15

DEFMS.ASM contains offsets for the BIOS entry points, System Functions, DOS Interrupts, Program Header structure, FCB structure, Directory Entry structure, and Disk Errors. DEFMS.ASM is an include during assembly. I, 19

DEFMTR.ASM contains the global entry points to the ROM jump vectors. DEFMTR.ASM is an include during assembly. I, 48

DEFZ207.ASM contains the definitions for the Z-207 disk controller. DEFZ207.ASM is an include during assembly.

DOBIOS.BAT is a batch file that assembles and links all of the source modules that are used to create the BIOS.

DSK.ASM contains the source for the disk drivers. DSK.ASM is an include during assembly.

MEMTST.EXE is a menu-driven memory testing utility. MEMTST may be invoked from the system prompt as a standard command, and can test any bank of system memory or any plane of video memory. The information on this utility is contained in Appendix F.

PARMS.ASM contains the control parameters that define constants and flags that control the BIOS assembly process.

RDCPM.COM is used to read CP/M formatted files and copy them to Z-DOS formatted files. Information on RDCPM begins on Page 6.106.

BATCH PROCESSING (.BAT FILENAME EXTENSIONS) COMMANDS

Brief

Format: **[d:] <filename> [<parameters>...]**

Command Location: System (batch processing is handled by system)

Purpose: To allow batch processing of commands

Details

Batch Commands

Invoking Batch

The Z-DOS batch facility allows files containing commands to be submitted for processing internally by Z-DOS. "Batches" of commands in such files are processed as if they were typed at a terminal.

The command to call the batch facility consists of entering a filename only. The file must be named with the .BAT extension.

A batch file is submitted for execution by entering its filename without the extension. The command is entered:

A: **<filespec>[<parameters>]**

Replaceable Parameters

Replaceable Parameters

By creating a .BAT file with "dummy" parameters (parameters are replaceable with real values), you may pass parameters to the .BAT file when it is executed. You may specify up to 10 dummy parameters %0 through %9. The percent sign (%) used in front of the parameter number tells COMMAND.COM that the number is a parameter that will be replaced by a real value.

DEFMS.ASM
-1-1
6-26-83

```
;
; Definitions for MS-DOS
;
; Last modified: 8/27/82  dtp

    IFDEF BIOS
   FBIOS = BIOS
    ELSE
   FBIOS = 0
    ENDIF

    IF NOT FBIOS

LORGADDR    =       400H    ; Loader org address


;
; BIOS entry points
;

BIOS_SEG SEGMENT AT 40H             ; Segment where the BIOS is located

; MicroSoft(MS) defined entry points

   ORG 0*3
  BIOS_INIT         LABEL FAR       ; Initialization routine (only exists
                                    ;   at boot time)
   ORG 1*3
  BIOS_STATUS       LABEL FAR       ; Console input status
   ORG 2*3
  BIOS_CONIN        LABEL FAR       ; Console input
   ORG 3*3
  BIOS_CONOUT       LABEL FAR       ; Console output
   ORG 4*3
  BIOS_PRINT        LABEL FAR       ; Printer output
   ORG 5*3
  BIOS_AUXIN        LABEL FAR       ; Aux input
   ORG 6*3
  BIOS_AUXOUT       LABEL FAR       ; Aux output
   ORG 7*3
  BIOS_READ         LABEL FAR       ; Disk input
   ORG 8*3
  BIOS_WRITE        LABEL FAR       ; Disk output
   ORG 9*3
  BIOS_DSKCHG       LABEL FAR       ; Disk change status
   ORG 10*3
  BIOS_SETDATE      LABEL FAR       ; Set current date
   ORG 11*3
  BIOS_SETTIME      LABEL FAR       ; Set current time
   ORG 12*3
  BIOS_GETDATE      LABEL FAR       ; Get current date
   ORG 13*3
  BIOS_FLUSH        LABEL FAR       ; Flush keyboard buffer
   ORG 14*3
  BIOS_MAPDEV       LABEL FAR       ; Device mapping
   ORG 15*3
  BIOS_MRES9        LABEL FAR       ; Reserved for MicroSoft entry points
   ORG 16*3
  BIOS_MRES8        LABEL FAR
```

```
    ORG 17*3
   BIOS_MRES7       LABEL FAR
    ORG 18*3
   BIOS_MRES6       LABEL FAR
    ORG 19*3
   BIOS_MRES5       LABEL FAR
    ORG 20*3
   BIOS_MRES4       LABEL FAR
    ORG 21*3
   BIOS_MRES3       LABEL FAR
    ORG 22*3
   BIOS_MRES2       LABEL FAR
    ORG 23*3
   BIOS_MRES1       LABEL FAR


; Zenith Data System(ZDS) defined entry points

    ORG 24*3
   BIOS_DSKFUNC     LABEL FAR       ; Disk function
    ORG 25*3
   BIOS_PRNFUNC     LABEL FAR       ; PRN(Printer) function
    ORG 26*3
   BIOS_AUXFUNC     LABEL FAR       ; AUX(modem) function
    ORG 27*3
   BIOS_CONFUNC     LABEL FAR       ; CON(console) function
    ORG 28*3
   BIOS_ZRES4       LABEL FAR       ; Reserved for Zenith entry points
    ORG 29*3
   BIOS_ZRES3       LABEL FAR
    ORG 30*3
   BIOS_ZRES2       LABEL FAR
    ORG 31*3
   BIOS_ZRES1       LABEL FAR
    ORG 32*3
   BIOS_REL LABEL BYTE      ; Bios release number in hex
                            ;  (ie 012H is release 1.2x)
    ORG OFFSET BIOS_REL+1
   BIOS_CTADDR      LABEL WORD      ; Addr of configuration information
    ORG OFFSET BIOS_CTADDR+2

BIOS_SEG ENDS

;
; Configuration vecter
;

CONFG_DSK  EQU 0           ; Addr of disk vector
CONFG_PRN  EQU CONFG_DSK+2 ; Addr of PRN configuraiton table
CONFG_AUX  EQU CONFG_PRN+2 ; Addr of AUX configuration table
CONFG_CON  EQU CONFG_AUX+2 ; Addr of CON configuration table
CONFG_FNT  EQU CONFG_CON+2 ; Addr of Font information
  FNT_RAM    EQU 0           ; Ptr to font table in RAM
  FNT_ROM    EQU FNT_RAM+4   ; Ptr to font table in ROM
  FNT_SIZE   EQU FNT_ROM+4   ; Size of font table in ROM
  FNT_MSIZE          EQU FNT_SIZE+2  ; Space allocated for font table in RAM
CONFG_CLOCK          EQU CONFG_FNT+2 ; Addr of Date and time fields
  BIOS_DATE            EQU 0         ; Days since Jan 1, 1980
  BIOS_HRS   EQU BIOS_DATE+2 ; Hours since midnight
  BIOS_MIN   EQU BIOS_HRS+1  ; Minutes
```

```
  BIOS_SEC   EQU BIOS_MIN+1  ; Seconds
  BIOS_HSEC          EQU BIOS_SEC+1  ; Hundredths of seconds(a word)
CONFG_DOSTB        EQU CONFG_CLOCK+2; Addr of DOS disk tables
CONFG_MCL  EQU CONFG_DOSTB+2; Addr of memory control info
CONFG_SIZE EQU CONFG_MCL+2 ; Length of configuration vector

    ENDIF

BIOS_CREL  EQU 10H          ; Current release of BIOS
BIOS_WORKSP        EQU 256          ; Number of bytes needed for
                           ;   workspace in BIOS
BIOS_RELDATE       EQU 1046         ; Release date 11/12/82 (this changes
                           ;   for each release)
MS_SIZEMEM EQU 1           ; Flag for DOS to size memory at init


;
;  System functions for "interrupt 21"
;    (Note: functions followed by "*" are
;           not CP/M compatable
;

DOSF_TERM  EQU  0          ; Program terminate
DOSF_CONIN EQU  1          ; Console input
DOSF_CONOUT        EQU  2          ; Console output
DOSF_AUXIN EQU  3          ; Aux input
DOSF_AUXOUT        EQU  4          ; Aux output
DOSF_PRINTOUT      EQU  5          ; Printer output
DOSF_DRCIO EQU  6          ; Direct console I/O
DOSF_DRCI  EQU  7          ; * Direct console input
DOSF_DRCINE        EQU  8          ; * Console input(no echo)
DOSF_OUTSTR        EQU  9          ; Output string
DOSF_INSTR EQU 10          ; Input string
DOSF_STCON EQU 11          ; Status of console
DOSF_CONINF        EQU 12          ; * Flush keyboard buffer and input
DOSF_RSDISK        EQU 13          ; Disk system reset
DOSF_SELDISK       EQU 14          ; Select default disk
DOSF_OPFILE        EQU 15          ; Open file
DOSF_CLFILE        EQU 16          ; Close file
DOSF_SRHFI EQU 17          ; Search for first
DOSF_SRHNX EQU 18          ; Search for next
DOSF_DEFILE        EQU 19          ; Delete file
DOSF_SEQREAD       EQU 20          ; Sequential read
DOSF_SEQWRITE      EQU 21          ; Sequential write
DOSF_CRFILE        EQU 22          ; Create file
DOSF_REFILE        EQU 23          ; Rename file
DOSF_24            EQU 24          ; * not used
DOSF_GETDISK       EQU 25          ; Get default disk
DOSF_SDIOA EQU 26          ; Set disk I/O address
DOSF_GFATA EQU 27          ; * Get file allocation table addr

;* The remaining functions are not CP/M compatable

DOSF_GFATA128      EQU 28          ; Get file allocation table addr
DOSF_29            EQU 29          ; not used
DOSF_30            EQU 30          ; not used
DOSF_31            EQU 31          ; not used
DOSF_32            EQU 32          ; not used
DOSF_RANREAD       EQU 33          ; Random read
DOSF_RANWRITE      EQU 34          ; Random write
```

```
DOSF_GFSIZE         EQU 35              ; Get file size
DOSF_SFPOS EQU 36               ; Set file position
DOSF_SIVEC EQU 37               ; Set interrupt vecter
DOSF_CESEG EQU 38               ; Create segment
DOSF_RBLREAD        EQU 39              ; Random block read
DOSF_RBLWRITE       EQU 40              ; Random block write
DOSF_PARSE EQU 41               ; Parse file name
DOSF_GDATE EQU 42               ; Get date
DOSF_SDATE EQU 43               ; Set date
DOSF_GTIME EQU 44               ; Get time
DOSF_STIME EQU 45               ; Set time
DOSF_CVERF EQU 46               ; Set/Reset verify flag


;
; Define the interrupts
;

DOSI_TERM  EQU 20H              ; Program terminate
DOSI_FUNC  EQU 21H              ; Perform a function
DOSI_TADDR EQU 22H              ; Terminate address
DOSI_CADDR EQU 23H              ; ^C Exit address
DOSI_FERADDR        EQU 24H             ; Fatal error exit addr
DOSI_ADREAD         EQU 25H             ; Absolute disk read
DOSI_ADWRITE        EQU 26H             ; Absolute disk write
DOSI_TERMR EQU 27H              ; Program terminate, but stay resident


;
; Define the program header
;

PHD_TERM   EQU 000H             ; Termination point (has INT 20H)
PHD_MEMSIZE         EQU 002H            ; Memory size (first seg num
                                ;   after end of mem)
PHD_AFUNC  EQU 005H             ; Alternate function entry point
PHD_EXADDR EQU 00AH             ; Exit handler addr
PHD_ABADDR EQU 00EH             ; ^C handler addr
PHD_FEADDR EQU 012H             ; Fatal error handler addr
PHD_STACK  EQU 05BH             ; End of stack area
PHD_FCB1   EQU 05CH             ; First program argument
PHD_FCB2   EQU 06CH             ; Second program argument
PHD_DIOA   EQU 080H             ; Default disk transfer area
PHD_CODESTART       EQU 100H            ; Start of code (Size of a PHD)


;
; Define the "User" File control block (FCB)
;

FCB_DRIVE  EQU 0                ; Drive number
FCB_FNAME  EQU FCB_DRIVE+1 ; File name
FCB_EXT             EQU FCB_FNAME+8 ; Extension to file name
FCB_CURBLK EQU FCB_EXT+3    ; Current block
FCB_RECSZ  EQU FCB_CURBLK+2 ; Record size
FCB_FILSZ  EQU FCB_RECSZ+2 ; File size
FCB_DATE   EQU FCB_FILSZ+4 ; Date file modified
FCB_TIME   EQU FCB_DATE+2  ; Time file modified
FCB_RES             EQU FCB_TIME+2  ; Reserved
FCB_CURREC EQU FCB_RES+8    ; Current record(in block)
```

```
FCB_RANREC EQU FCB_CURREC+1 ; Random record number
FCB_SIZE   EQU FCB_RANREC+4 ; Size of a FCB


;
; Define the extended file control block
;

XFCB_FLAG  EQU 0            ; Flag field
XFCB_RES   EQU XFCB_FLAG+1 ; Reserved
XFCB_ATTR  EQU XFCB_RES+5  ; Attribute byte
  XFCBA_HID          EQU 02H          ; Hidden files
  XFCBA_SYS          EQU 04H          ; System files
XFCB_FCB   EQU XFCB_ATTR+1 ; Normal FCB
XFCB_SIZE  EQU XFCB_FCB+FCB_SIZE ; Size of a XFCB


;
; Define the directory entries
;

DE_FNAME   EQU 0            ; File name
DE_EXT             EQU DE_FNAME+8  ; Extension to file name
DE_ATTR            EQU DE_EXT+3    ; File attribute
  DEA_HID    EQU 02H          ; Hidden file
  DEA_SYS    EQU 04H          ; System file
DE_RES             EQU DE_ATTR+1   ; Reserved
DE_TIME            EQU DE_RES+10   ; Time the file was modified
DE_DATE            EQU DE_TIME+2   ; Date the file was modified
DE_START   EQU DE_DATE+2   ; Starting allocation unit
DE_FSIZE   EQU DE_START+2  ; File size
DE_SIZE            EQU DE_FSIZE+4  ; Size of a DE (should be 32)


;
; Define the "Drive parameter table" for MS-DOS usage
;  (Used only by the BIOS at init time)

DPT_SECSIZ EQU 0           ; Size in bytes of a physical sector
DPT_CLUSIZ EQU DPT_SECSIZ+2 ; Number of sectors in an
                            ;   allocation unit
DPT_RESSEC EQU DPT_CLUSIZ+1 ; Number of reserved sectors at
                            ;   start of disk
DPT_FATCNT EQU DPT_RESSEC+2 ; Number of FAT's
DPT_MAXENT EQU DPT_FATCNT+1 ; Number of directory entries
DPT_DSKSIZ EQU DPT_MAXENT+2 ; Number of physical sectors on
                            ;   the disk
DPT_SIZE   EQU DPT_DSKSIZ+2 ; Size of a DPT


;
; Define the disk errors
;

DSKE_WRITEP        EQU 0           ; Write protect
DSKE_NREADY        EQU 2           ; Not ready
DSKE_DATA  EQU 4           ; Data error
DSKE_SEEK  EQU 6           ; Seek
DSKE_SECT  EQU 8           ; Sector not found
DSKE_WFAULT        EQU 10          ; Write fault
```

```
DSKE_OTHER EQU 12          ; Anything else
```

The parameters are substituted in order on the command line. A one-to-one correspondence is set up between the prototype commands in the command line and the replaceable parameters in the batch file. This relationship is illustrated in Figure 6.1 where:

- "A:" is the prompt for the currently logged drive.
- "<command>" is the primary name of a file with a .BAT extension.
- "filespec" is the name for a prototype command in the form [<d:>]<filename>[.<ext>].
- "%" (percent sign) followed by a number "n" is the dummy parameter.
- The prototype command is substituted.

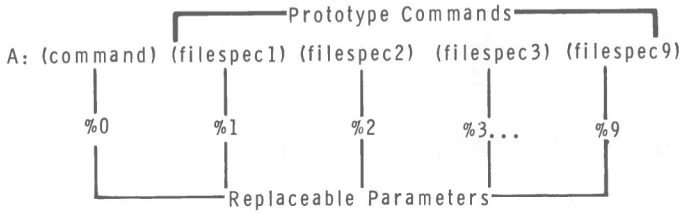


Figure 6.1
One-to-one Relationship for Replaceable Parameters

If the dummy parameter %0 is used, the batch facility substitutes the filename of the batch command itself for parameter %0. This allows the creation of batch commands that can be used on more than just one set of files, and that can be used to restart themselves.

Two Z-DOS commands are available expressly for use in batch files: REM and PAUSE. REM permits the inclusion of remarks and comments in batch files. PAUSE prompts the user with an optional message and permits either continuing or aborting execution of a batch file at a given point. REM and PAUSE are further described on Page 6.109 and 6.102, respectively.

AUTOEXEC.BAT

When you boot up your system, COMMAND.COM searches for the file AUTOEXEC.BAT. If the file exists on disk, then the batch facility is automatically invoked to execute the commands contained in AUTOEXEC.BAT. In this case, execution of the TIME and DATE commands at start-up is bypassed. If COMMAND.COM does not find AUTOEXEC.BAT, then the normal Z-DOS prompt is displayed.

Application

The following examples illustrate the batch facility.

Examples of Batch Use

Create a .BAT batch file using EDLIN (see Page 6.61, for a complete description of the EDLIN program). For example:

```
A: EDLIN NEWDISK.BAT  RETURN
```

(the .BAT extension must be given to open a batch file unless you want to use RENAME to later rename the file with .BAT as its extension)

```
EDLIN version1.02
*I

  1: REM This is file NEWDISK.BAT
  2: REM (the .BAT extension must be given)
  3: PAUSE Insert disk in B:
  4: FORMAT B:/S
  5: DIR B:
  6: CHKDSK B:
  7: <CTRL-C>
*E
```

The file NEWDISK.BAT now resides on the disk in drive A.

Execute this .BAT file, by entering the filename without the .BAT extension:

```
A: NEWDISK  RETURN
```

The result is the same as if you entered each of the lines in the .BAT file as individual commands.

To pass parameters to the .BAT file, create a .BAT file containing prototype commands with dummy entries.

For example:

```
A: EDLIN ASMFILE.BAT  RETURN
   *I
      1: REM This is A:ASMFILE.BAT
      2: REM START BATCH FILE
      3: COPY %1.ASM %2.ASM
      4: MASM %2,%2,%2;
      5: TYPE %2.PRN
      6: TYPE %0.BAT
      7: <CTRL-C>
   *E
```

The file ASMFILE.BAT now resides on the disk in drive A.

Execute this .BAT file and pass parameters, by entering:

```
A: ASMFILE  A:MYPROG  B:MYPROG  RETURN
```

The result is the same as if you had entered each of the following commands at your terminal:

```
A: REM This is A:ASMFILE.BAT  RETURN
A: REM START BATCH FILE  RETURN
A: COPY A:MYPROG.ASM B:MYPROG.ASM  RETURN
A: MASM B:MYPROG,B:MYPROG,B:MYPROG;  RETURN
A: TYPE B:MYPROG.PRN  RETURN
A: TYPE A:ASMFILE.BAT  RETURN
```

When you initially boot your Z-DOS Distribution Disk I, it contains an AUTOEXEC.BAT file containing the following commands:

DATE	(requests Date entry)
TIME	(requests Time entry)
MAKE SYSCOPY.DAT	(begins backup procedure)
ERASE B:AUTOEXEC.BAT	(removes batch file so that the next time the disk is booted, it will not require you to make a backup copy)

More information about MAKE is found on Page 6.88.

CHKDSK (CHECK DISK) COMMAND

Brief

Format: **CHKDSK [d:]**

Command Location: File

Purpose: Examine the directory of the default or designated drive and provide various information about the disk contents.

Details

Scans the directory of the default or designated drive and check it completely for consistency. CHKDSK should be run on each disk occasionally to verify the integrity of the directory structure. If any errors are detected, the appropriate error message is displayed and corrective action is attempted.

Invoking CHKDSK

If you have two working drives connected to your system that contain disks and they have the names A and B respectively, you could enter

A: **CHKDSK A: RETURN**

(CHKDSK message appears for A)

A: **CHKDSK B: RETURN**

(CHKDSK message appears for B)

After you enter the command line and press RETURN, CHKDSK will check the disk. When the disk has been checked, CHKDSK displays error messages, if any; then a status report.

The status report indicates:

- Number of disk files
- Size of the disk
- Number of bad sector bytes
- Amount of free space
- Total size of central memory
- Amount of memory available for program execution

A sample run of the CHKDSK program might appear as follows:

```
A: CHKDSK B:  RETURN

CHKDSK Version x.x
          160256  bytes total disk space
            8192  bytes in 2 hidden files
           30720  bytes in 8 user files
          121344  bytes available on disk

           65536  bytes total memory
           53152  bytes free
```

If an error is detected, CHKDSK returns one of the following error messages:

Error Messages

```
Allocation error for file <filename>
```

The named file had a data block allocated to it that did not exist (that is, a data block number larger than the largest possible block number). CHKDSK truncates the file short of the bad block.

```
Disk not initialized
```

No directory or file allocation table was found. If files exist on the disk, and the disk has been physically harmed, it may still be possible to transfer files from this disk to recover data.

```
Directory error-file: <filename>
```

No valid data blocks are allocated to the named file. CHKDSK deletes the file.

```
Files cross-linked: <filename> and <filename>
```

The same data block is allocated to both files. No corrective action is taken. To correct the problem, first use the COPY command to make copies of both files; then, delete the originals. Review each file for validity and edit as necessary.

```
File size error for file <filename>
```

The size of the file in a directory is different from its actual size. The size in the directory is automatically adjusted to indicate its actual size on the disk. (The amount of useful data may be less than the size shown because the last data block may not be fully used.)

```
XXXXXX bytes of disk space freed
```

Disk space shown as allocated was not actually allocated and has been freed.

Application

CHKDSK is useful to verify that the current disk contents and the disk directory do indeed coincide. CHKDSK can straighten out most of the problems it finds, which might cause errors if left unchecked.

Typing <**CTRL-P**> **CHKDSK** and **RETURN** sends a copy of the status report to your printer. This may then be cut to size and attached to the disk sleeve for reference. This may also be done with the DIR (Directory) command but if there are a lot of files on the disk, the listing produced could prove too lengthy to attach without folding.

CONFIGUR (PRINTER CONFIGURATION) COMMAND

Brief

Format: **CONFIGUR [?]**

Command Location: File

Purpose: Configures the I/O system to use the correct protocol for your printer, modem or other I/O devices.

Details

Configure is a utility program which allows you to configure your system's logical devices so that they fit the protocol requirements of your printer, modem or other hardware peripherals that you connect to your Z-100. "Protocol" is the word used when reference is made to the method of communications required for data transfer between a computer and a peripheral. The protocol also offers a means of controlling how the peripheral performs its specific operations. A device's protocol consists of a set of standard signals that the hardware device recognizes and uses so that its operation may be correctly directed.

CONFIGUR is a quick way for you to select the Z-DOS protocol that your devices can recognize. CONFIGUR enables the I/O Manager (IO.SYS) to communicate with your devices using the correct baud rate, communication lines, control and data signals necessary for information transfer to a particular peripheral device.

PRN and AUX Devices

The configurable logical devices are the PRN and the AUX devices. A logical device name is assigned to a group of information that is contained in IO.SYS. When that device name is used in a command to read or write data, IO.SYS uses the information that it contains about that device to carry out the command.

Logical devices configurable to different identities make it easy for the operating system to use a multitude of devices. If the operating system could not use a logical device to represent a physical device, either the operating system would need to be rewritten every time a different device was used, or it would be necessary for it to contain information about all devices. Rewriting an operating system is an extremely long and difficult task. An operating system that contained information for every possible device would be extremely large.

Logical devices present a very practical approach to enable the operating system to handle a lot of different devices. When confronted with one of the reserved logical device names (PRN or AUX), the operating system knows exactly where to look to get information currently used to describe the device.

CONFIGUR also knows exactly where the descriptions of the logical devices are stored. When CONFIGUR is told to save a new description, it overwrites the old device description. This means that if the PRN device is currently the description of a WH-14 printer, commands that have PRN as an argument cause the operating system to use the WH-14 description. If CONFIGUR is then invoked, and the PRN device is redefined to the characteristic protocol of a H/Z-25, the operating system then recognizes and uses the different protocol. Even though the actual physical device changes, Z-DOS is able to communicate to it because it looks up the devices definition under the logical name.

Invoking CONFIGUR

To invoke the command, type **CONFIGUR** at the system prompt:

A: **CONFIGUR ? RETURN**

CONFIGUR responds with a brief description of PRN, AUX, and CONFIGUR's function.

To begin to configure your system for a particular devices protocol, type:

A: **CONFIGUR RETURN**

CONFIGUR then displays the following list of configuration options:

```
                    CONFIGUR Version 1.01
                Copyright (C) 1982 Zenith Data Systems

   Use a combination of the following options to configure a device

A.  Configure PRN device
B.  Configure AUX device
D.  Exit with no changes

Enter selection (A-C):
```

At this Master Menu, you type in the letter **A**, **B**, or **C**, corresponding to the choice that you want to make.

Press **C** to return to the system prompt without making any configuration changes.

Depending on the selection you choose on this Master Menu (option A, PRN option B, AUX) additional menus appear that offer alternative system configuration options.

Ending CONFIGUR

Because there are several menus and options that are available depending on the options you select, ending the program is discussed first before describing the configurations that are available.

After making your selections, you will return to this Master Menu (notice that three additional options appear):

```
                    CONFIGUR Version 1.01
                Copyright (C) 1982 Zenith Data Systems

      Use one of the following options to configure a device

A.  Configure PRN device
B.  Configure AUX device
    Use the following options to modify an existing system
C.  Exit program
D.  Make changes to disk
E.  Make changes to memory
F.  Make changes to both disk and memory

Enter selection (A-F):
```

Exit Without Making Any Changes

Save Changes to Disk

Options D, E, and F appear on the Master Menu after you have made a selection. These options allow you to use the configuration options that you select.

You may still exit without saving any of the options that you have selected by selecting C.

Option D saves the changes that you selected to a disk. On selecting option D, you are asked which disk:

```
Enter drive name with system to modify (A-D):
```

Then redisplay above for exit

This feature makes it possible to configure the systems on disks other than your current system disk. If you have other disks that will be used with different (or particular) devices, this becomes useful. Another use for this feature is if you want to change your current system disks device configuration without disturbing the system configuration that is in the memory of your computer at the time you use CONFIGUR.

Save Changes to Memory

Selecting option E, alters the system configuration that is currently in your computer's memory, without making that configuration more permanent by saving it to a disk. When you press **CTRL-RESET** and reboot your computer, the system configuration you selected is erased and the one on the disk is again read into memory. This allows you to make a temporary change.

Save Changes to both Memory and Disk

Select the F option if you want to make the changes more permanent — to memory and disk. This option also asks you the drive name where you want to save the changes, so you can change memory and disks other than your booted disk if you desire.

CONFIGUR's Secondary Menus

From the Master Menu, both the A option (PRN device) and the B option (AUX: device) lead to a secondary set of menus. Both options have menus that look extremely similar.

NOTE: If Option E (the Diablo printer) is selected, consult your hardware manual for information on jumpering the port connector. If Option I (User defined) is selected, and ETX/ACK or DC3/DC1 protocols are selected, you also need to consult the hardware manual.

Here is the PRN (A Option) Menu:

```
Use one of the following options to select the PRN device

A.  Centronics or MX-80 Parallel (Parallel)
B.  MX-80 Serial (Serial A (J1), 4800 baud, DTR Pos. (Pin 20))
C.  H/Z-25 (Serial A (J1), 4800 baud, RTS Pos. (Pin 4))
D.  H-14/WH-24 (Serial A (J1), 4800 baud RTS Neg. (Pin 4))
E.  Diablo 630/1640 (Serial A (J1), 1200 baud, ETX/ACK)
F.  WH-23/WH-33/WH-43 Modem (Serial A (J1), 300 baud, No handshake)
G.  WH-12 Votrax Type-N-Talk (Serial A (J1), 4800 baud, RTS Pos. (Pin 4))
H.  System CRT
I.  User defined
J.  Exit with no changes

Note: Option F may require a special cable

Enter Selection (A-J):
```

Here is the AUX (B Option) Menu:

```
Use one of the following options to select the AUX device

A.  Centronics or MX-80 Parallel (Parallel)
B.  MX-80 Serial (Serial B (J2), 4800 baud, DTR Pos. (Pin 20))
C.  H/Z-25 (Serial B (J2), 4800 baud, RTS Pos. (Pin 4))
D.  H-14/WH-24 (Serial B (J2), 4800 baud RTS Neg. (Pin 4))
E.  Diablo 630/1640 (Serial B (J2), 1200 baud, ETX/ACK)
F.  WH-23/WH-33/WH-43 Modem (Serial B (J2), 300 baud, No handshake)
G.  WH-12 Votrax Type-N-Talk (Serial B (J2), 4800 baud, RTS Pos. (Pin 4))
H.  System CRT
I.  User defined
J.  Exit with no changes

Note: Options A through E and G may require a special cable

Enter Selection (A-J):
```

PRN and AUX Menu Differences

The differences between the PRN (option A) and AUX (option B) Menus may not be immediately apparent, though you should take careful note of what these differences are so that the configuration you select is correct.

The first difference, is in the top line of each menu. The top line states specifically whether the menu will apply to the AUX device or to the PRN device.

The second most noticable difference between the two menus are the notes that appear right above the "Enter Selection" line. These notes point out that a special cable may be required to connect peripherals to the referenced ports.

The most important differences between the two menus are not as easily noticed. Compare the second choice (selection B) in each menu:

PRN Menu selection:

```
B.  MX-80 Serial (Serial A (J1), 4800 baud, DTR Pos. (Pin 20))
```

AUX Menu selection:

```
B.  MX-80 Serial (Serial B (J2), 4800 baud, DTR Pos. (Pin 20))
```

The difference here is that the PRN Menu makes its changes for Serial Port A, which is marked J1 on the back of your Z-100. The AUX Menu makes its changes for Serial Port B, marked J2 on the Z-100. Logical device named PRN always refers to a peripheral connected at Serial Port A (J1) or to the Parallel Port, and AUX usually refers to a peripheral connected at Serial Port B (J2) but, may also be used for the Parallel Port.

Immediately after entering selections A through G from the PRN or AUX Menus, a diagram of the rear panel of the Z-100 appears on your computer screen. The diagram shows you the location of the correct receptacle for the port where the device you configured for should be connected. The rear panel diagram that appears, looks like:

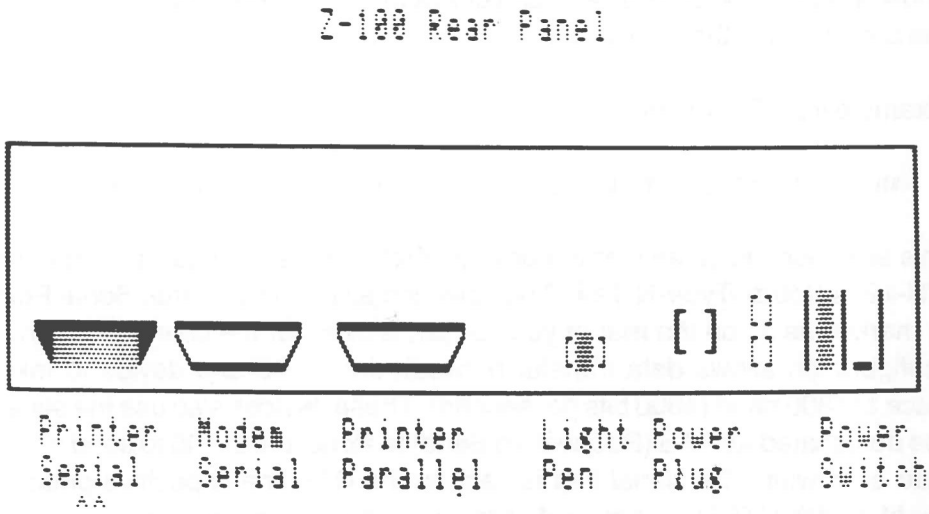


Figure 6.2

CONFIGUR's Z-100 Rear Panel Diagram

An arrow will point at the receptacle where you should connect your peripheral, in accordance with the configuration you selected.

PRN and AUX Selections

The selections available from the AUX and PRN Menus give the name or model number of the device to which each applies. Information that is contained in parentheses after the device names, indicates the standard configuration required by the device.

NOTE: Pressing the option letter (A-G), automatically sets the configuration to the default values contained within the options parentheses. The only other thing that you need do, is tell CONFIGUR where you want to save the configuration that you have just selected. CONFIGUR sets the protocol automatically.

This standard configuration applies to the way the peripheral was set when it left the Zenith Data Systems/Heath factory.

If any switches on the device have been changed since it left the factory, they may need to be reset to the values shown on the menu in parentheses in order for the Z-100 to work properly. Consult the manual that accompanied your peripheral device if you need to change any of the switch settings. It is a good idea to check the peripheral's switch settings so that you know what the switches do and can check to see that they do indeed match the configuration that you select.

Example from PRN Menu:

```
G.  WH-12 Votrax Type-N-Talk (Serial A (J1), 4800 baud, RTS Pos. (Pin 4))
```

This selection indicates that the devices that use this configuration are the WH-12, a Votrax Type-N-Talk. The selection also indicates that Serial Port A, marked as J1 on the rear of your Z-100, is used for the connection. This configuration allows data transfer between the Z-100 and device to take place at 4800 baud (4800 bits per second). These devices also use the serial line designated as RTS (Request To Send) to signal the Z-100 to send more data or to wait. The signal that is sent on the RTS line is positive polarity (high), and the RTS line is at pin 4 on the serial cable's receptacle.

System CRT as Logical Device

Option H (System CRT) is unique in the sense that you can send data to your monitor's screen that you might normally send to a printer. The rear panel diagram is not shown after the selection because your CRT is already connected to your computer during this programs operation. This option allows you to use commands like **COPY *.BAT PRN** in order to display on screen the contents of all batch files from the disk in your default drive.

User Defined Protocol

Option I (user defined) is the most complex option available from CONFIGUR's Master Menu. This should only be used if you are completely confident that the protocol options you select are correct.

Option I allows the support of non-standard configurations, or especially customized protocols.

The selection of Option I causes CONFIGUR to ask a series of questions about the nature of the protocol that you need. The user defined option always starts out with the following series of menus and questions:

```
Valid options for device type are:

A System CRT
B Serial device
C Parallel device


Answer the following questions with Y for Yes and N for No

Strip parity on input?  (Y/N) <N>
Strip parity on output? (Y/N) <N>
Map lower case to upper on input?  (Y/N) <N>
Map lower case to upper on output? (Y/N) <N>
```

The prompts and questions are self explanatory. Among the information that is requested for the various protocol configurations are:

```
PAD character to be used and the number of PAD
characters to be sent
```

If you do not wish a pad character, press **RETURN**, and then enter a 0 as the number of pad characters, otherwise type the actual key character you wish to pad.

For example, to pad after all line feeds, press **LINE FEED**.

```
Type the key corresponding to your desired pad character.  <LINE FEED>

Enter the number of pad characters to send (0-255):
```

The port number to be used for the device:

```
Select one of the following I/O ports

A.  Serial A (J1)   (E8H)
B.  Serial B (J2)   (ECH)

Enter Port Selection
```

The baud rate used for data transmission:

```
Select one of the following baud rates

A.    45.5                  I.    1200
B.    50                    J.    1800
C.    75                    K.    2000
D.   110                    L.    2400
E.   134.5                  M.    4800
F.   150                    N.    9600
G.   300                    O.   19200
H.   600                    P.   38400

Enter baud rate selection:
```

The type of handshake protocol (if ETX/ACK is selected, you are also asked for the number of characters between the ETX/ACK handshake)

NOTE: Consult your hardware manual for information on jumpering the Serial A (J1) port connector so that it matches with the handshake protocol you select.

If ETX/ACK or DC3/DC1 protocols are selected, the device cannot be used for input.

```
Use the following to select a handshake protocol

A.  No Handshaking
B.  ETX/ACK
C.  DC3/DC1
D.  RTS Positive (Pin 4)
E.  RTS Negative (Pin 4)
F.  DTR Positive (Pin 20)
G.  DTR Negative (Pin 20)

Enter one of the handshake values:
```

The stop bits value to be used:

```
Use one of the following stop bit values

A. 1   stop bit
B. 1.5 stop bits
C. 2   stop bits

Enter one of the stop bit values:
```

Should parity be used — (if so, CONFIGUR asks if it should be odd or even parity):

```
Do you wish to use parity?  (Y/N) <N>
```

The word length in bits exclusive of stop bits and parity:

```
Use one of the following word length selections
NOTE: Word length is exclusive of stop bits and parity

A.  5 bit words
B.  6 bit words
C.  7 bit words
D.  8 bit words

Enter one of the word length values:
```

After all information regarding the protocol has been defined, the Z-100 Rear Panel Diagram is displayed, when appropriate, to show where the device is to be connected according to the configuration.

Application

The CONFIGUR program provides a way for you to change peripherals as your needs change. If you need to save time with a printer, you may choose a high-speed dot- matrix line printer, which might use an RTS positive handshake at 4800 baud (such as the Z-25/H-25). Or, your printer needs might require a cleaner, crisper look for correspondence and contracts, so you may choose a daisy- wheel printer that uses ETX/ACK handshaking and operates at 1200 baud (such as the Diablo 1640).

CONFIGUR makes it possible for you to connect both at the same time, or alter the configuration quickly so that you are able to use one printer, run CONFIGUR, connect a different printer and immediately start to use that printer.

Certain applications require a modem. CONFIGUR sets up your Z-100's system configuration so that you are able to use the modem easily.

Disks that contain applications for specialized use can be quickly modified with different system configurations, without changing the configuration of the system in your computer.

COPY COMMAND

Brief

Format: **COPY [</x>]<filespec> [d:][<filespec>]**
or
COPY <filespec>[d:]

Command Location: System

Purpose: To transfer (copy) a file from one location (the source) to another (the destination).

Details

The COPY utility copies the first filespec. If the second filespec parameter is not given, the copy will have the same name as the original (first filespec parameter).

NOTE: If the first filespec references a file that is on the default drive and the second filespec is not given, the COPY will be aborted. (Copying a file to itself is not allowed.) If this is attempted, Z-DOS will return the error message:

```
. File cannot be copied onto itself
        0 File(s) copied
```

The Z-DOS prompt will reappear following the error message.

The second parameter ([<filespec>]) may take three forms.

- If the second parameter is a drive designation only (d:), the original file will be copied, with the same name, to the designated drive.

- If the second parameter is a filename only (filename.ext), the original file will be copied to a file with the name specified on the default drive.
- If the second parameter is a full filespec (d:filename.ext), the original file will be copied to a file with the name specified on the drive designated.

The COPY command also allows you to join separate files (file concatenation) while copying. Concatenation is invoked by simply listing any number of files as parameters to COPY, separated by "+" (plus sign).

For example:

A: **COPY A.XYZ+B.COM+B:C.TXT BIGFILE.CRP**

The above command concatenates the contents of A.XYZ, B.COM, and B:C.TXT and places them in the file on the default drive called BIGFILE.CRP.

The concatenation operation is normally carried out in text (or ASCII) mode, meaning a CTRL-Z (1A hex) in the file is interpreted as the end-of-file mark. To combine binary files, this interpretation of the end-of-file may be overridden with the /B switch, which forces the command to use the physical end-of-file as the end of file (that is, the file length seen in the DIR command).

For example:

A: **COPY/B A.COM+B.COM**

Also, in the above example, no resulting file name was given. In this case, COPY finds the end of A.COM and appends B.COM to it, leaving the result named A.COM.

ASCII and binary files may be arbitrarily combined by using /B on binary files and /A on ASCII files. A switch (/A or /B) takes effect on the file it is placed after and applies to all subsequent files until another switch is found.

An /A or /B switch on the destination file determines whether or not a CTRL-Z (End of File character) is placed at the end of the file. (Source files read while /A is in effect have CTRL-Z stripped off. If /A is in effect when the file is written, a single CTRL-Z will be put back.) Thus, an additional CTRL-Z would be appended with a command such as:

```
A: COPY A.ASM/B B.ASM/A
```

This occurs because the /B on the first file prevents the CTRL-Z from being stripped, and the /A on the second puts one on. The primary practical application may be the reverse, where a CTRL-Z is stripped from the file.

For example:

```
A: COPY PROG.COM/B + ERRS.TXT/A NEWPROG.COM/B
```

It is assumed here that ERRS.TXT is an ASCII file that was generated by an editor, though it is actually considered constant data (error messages) by the program to which it is being appended. Since the result is a .COM file, a CTRL-Z at the end is not needed.

Even when not concatenating files, the /A and /B switches are still processed. When not concatenating, the copy command defaults to binary copy. By using the /A switch, the result file may be truncated at the first end-of-file mark:

```
A: COPY  A.TXT/A  B.TXT
```

B.TXT may be shorter than A.TXT if A.TXT contained an embedded CTRL-Z. B.TXT will have exactly one CTRL-Z, the last character of the file.

Concatenation with wild card (ambiguous) file names is allowed, and the COPY command normally "does what you want". To combine several files specified with a wild card name into a single file, use a command like:

```
A: COPY  *.LST  COMBIN.PRN
```

All files matching ***.LST** are combined into one file called COMBIN.PRN. Another type of task is performing several individual concatenations:

```
A:COPY *.LST+*.REF *.PRN
```

In this example, for each file found matching *.LST, that file is combined with the corresponding .REF file, with the result given the same name but with the extension .PRN. Thus, FILE1.LST will be combined with FILE1.REF to form FILE1.PRN, then XYZ.LST with XYZ.REF to form XYZ.PRN, and so on. The following COPY command combines all files matching *.LST, then all files matching *.REF, into one file called COMBIN.PRN:

```
A:COPY *.LST+*.REF COMBIN.PRN
```

It is easy to enter a concatenation COPY command where one of the source files is the same as the destination, yet this often cannot be detected. For example, the following command is an error if ALL.LST already exists:

```
A:COPY *.LST ALL.LST
```

This is not detected, however, until it is ALL.LST's turn to be appended. At this point it could already have been destroyed.

COPY handles the problem this way: as each input file is found, its name is compared with the destination. If they are the same, that one input file is skipped, and the message `Content of destination lost before copy` is printed. Further concatenation proceeds normally. This allows "summing" files, with a command like

```
A:COPY ALL.LST + *.LST
```

This command appends all *.LST files, except ALL.LST itself, to ALL.LST. The error message is suppressed in this case, since this is produced by a true physical append to ALL.LST.

Application

This command allows you to: make duplicate copies of files; copy programs onto the disks where they will be needed for other purposes; do some file manipulation through concatenation; and can quickly copy a file onto your computer's screen or onto a line printer.

CREF (MACRO CROSS REFERENCE) COMMAND

Brief

Format: **CREF [<crffile>,<listing>]**

Command Location: File

Purpose: Creates an alphabetical cross reference listing of symbols from the cross reference file that is produced by MASM in response to a filename other than NUL to MASM's fourth prompt.

Details

CREF is covered extensively in Chapter Thirteen.

The CREF utility creates an expanded cross reference listing from the cross reference file that is generated by the macroassembler. CREF requires that a .CRF file is generated by MASM (MACRO-86) in order to build a listing. To use CREF, you must plan ahead and respond with a filename other than the default (NUL.CRF, which generates no .CRF file) to the fourth prompt that is generated by MASM:

```
Cross reference [NUL.CRF]:
```

If the cross reference file generated by MASM has an extension other than .CRF, you must be certain that you respond to CREF with the correct and complete filename.ext.

Invoking CREF

There are two methods to invoke CREF: 1) with no parameters, or 2) with the source (<crffile>) and the destination (<listing>) filespecs specified on the command line.

Method 1:

If you select Method 1 you will be prompted in turn for the source and destination. For example if you entered:

```
A:CREF  RETURN
```

Then, CREF prompts:

```
Cross reference [.CRF]:
```

Here you need only enter a filename, CREF assumes the default extension of .CRF, unless your .CRF file was created with another extension. The drive name of the file must also be specified if it is on a drive other than the default drive.

If you terminate the line to this prompt with a semicolon (;) before entering a **RETURN**, CREF assumes the default crffile, which has a .REF extension, for the next prompt.

The second CREF prompt is:

```
List filename [crffile.REF]:
```

Press **RETURN** if the .REF file is to have the same filename as the .CRF file. Otherwise, enter the correct filename (and <.ext>, if other than .REF) for the listing. Again, drive names must be specified if the files use other than the default drive.

Method 2:

If you select Method 2, enter:

A: **CREF <crffile>,<listing>**

where <crffile> is the [d:]<filename> of a .CRF file created by MASM; or where <crffile> is the [d:]<filename.ext> of a cross reference file; and where <listing> is the [d:]<filename> of the cross reference listing that will receive a .REF extension; or where <listing> is the [d:]<filename.ext> of the created file.

Error Message

If the <crffile> is not found, CREF returns the following error message:

```
Fatal I/O Error: 110
 in: <crffile>.CRF
```

Application

The CREF generated cross reference listing adds one more tool to the list of utilities available for the debugging and development processes under Z-DOS. The listing generated by the CREF utility gives you easy index to symbol occurence and definition, and can shorten search time.

DATE COMMAND

Brief

Format: **DATE [<mm>-<dd>-<yy>]**

Command Location: System

Purpose: Displays the date and then prompts the user to enter a new date.

Details

The DATE command may be invoked in two ways:

Invoking DATE

1) It may be entered from the command line followed immediately by the date that you select:

```
A:DATE 11/24/83 RETURN
```

In this case, no message will appear and you will see the command mode prompt A: again.

2) The date command may be entered at the command line by itself.

```
A: DATE RETURN
```

This method displays the current date and then asks if you want to change the current date. The DATE command will return the message:

```
Current date is <day> <mm>-<dd>-<yyyy>
Enter new date: _
```

- If you do not want to change the date shown, press **RETURN**.

- If you do want to change the date, the new date must be entered in numerals only; letters are not permitted. The allowable parameters are:

  <mm> = 1-12
  <dd> = 1-31
  <yy> = 80-99 or 1980-2099

NOTE: The day is calculated automatically.

The date, month and year entries may be separated by hyphens (-) or slashes (/).

Z-DOS is programmed to change months and years correctly, whether the month has 31, 30, 29, or 28 days. Z-DOS handles leap years, too.

This command is used with bootable disks that contain an AUTOEXEC batch file. When this type of disk is booted and AUTOEXEC first comes up, the date prompt is not displayed. If you need the date, the DATE command provides the means to input it.

You may want to use the DATE command simply to change the date that will be recorded for any files that are edited. Z-DOS uses whatever date is entered as the new date as long as the parameters and separators are legal.

Error Messages

If the parameters or separators are not legal, Z-DOS returns the messages:

```
Invalid date, enter as mm-dd-yy
Enter new date:_
```

and waits for the user to enter a legal date.

Application

The date stamp that appears with the DIR (Directory) command is a direct result of the DATE function. If you generate several versions of a file, the date stamp tells you which file is most current.

The date stamp is also useful when it comes to "housekeeping", when you want to go through previously used disks and erase files that are no longer needed or are outdated.

DEBUG COMMAND

Brief

Format: **DEBUG [<filespec>]**

Command Location: File

Purpose: A tool for software development that provides a controlled environment to load, examine and change binary code files.

Details

DEBUG is covered extensively in Chapter Seven.

All commands given to DEBUG may be aborted at any time by pressing **CTRL-C**. **CTRL-S** will suspend the display so that the user can read it before the output scrolls away. Any key other than CTRL-C or CTRL-S will restart the display.

To invoke DEBUG, enter:

Invoking DEBUG

A: **DEBUG [<filespec>]**

The filespec is an optional entry with the invocation command. The ability to enter the invocation command without a filespec allows absolute disk addresses to be used.

DEBUG loads a specified file starting at 100 hex in the lowest available segment. The CX register will be loaded with the number of bytes loaded.

If filespec is not entered COMMAND.COM will load DEBUG, which will in turn display its prompt (>).

If a syntax error occurs in a DEBUG command, DEBUG reprints the command line and indicates the error with an up-arrow and the word error. For example:

```
<dcs:100 cs:110
         ^ error (not a valid hex digit)
```

Any combination of upper and lower case may be used in DEBUG commands. Spaces or commas are legal delimiters for parameters. A delimiter is required only between two consecutive hexadecimal values.

DEBUG Command Descriptions

C(Compare) Command

Format: **C<range> <address>**

Compares the portion of memory specified by <range> to a portion of memory starting at <address>that is the same size.

D (Dump) Command

Format: **D[<address>[L <value>]]**
or
D[<range>]

Displays the memory contents of either a single address, a range of addresses, or the number of locations specified by <value> beginning at the <address> specified.

E (Enter) Command

Format: **E<address>[<list>]**

Enter an <address>, display its contents, and wait for the user's input.

F (Fill) Command

Format: **F<range> <list>**

Fill the addresses in the <range> with the values in the <list>.

G (Go) Command

Format: **G[=<address>[<address>...]]**

Execute the program.

H (Hex Arithmetic) Command

Format: **H<address> <address>**

Perform hexadecimal arithmetic on the two parameters.

I (Input) Command

Format: **I<value>**

Input and display one byte from the port specified by <value>.

L (Load) Command

Format: **L[<address>[<drive> <record> <record>]]**

Load a file into memory.

M (Move) Command

Format: **M<range> <address>**

Move the block of memory specified by <range> to the location beginning at the <address> specified.

N (Name) Command

Format: **N<filename>[<filename>...]**

Set up filename in proper format of an unopened file control block at DS:5CH, and set up parameters for program.

O (Output) Command

Format: **O<value> <byte>**

Send the <byte> specified to the output port specified by <value>.

Q (Quit) Command

Format: **Q**

Terminate the debugger.

R (Register) Command

Format: **R[<register name>]**

Display the contents of one or more CPU registers, and prompts for new value.

S (Search) Command

Format: **S<range> <list>**

Search the range specified for the <list> of bytes specified.

T (Trace) Command

Format: **T[=<address>][<value>]**

Execute one instruction and display the contents of all registers, flags, and the decoded instruction.

U (Unassemble) Command

Format: **U[<address>[L <value>]]**
U[<range>]

Disassemble the bytes and display the source statements that produced the bytes along with the addresses and byte's values.

W (Write) Command

Format: **W[<address>[<drive> <record> <record>]]**

Write the file being debugged to a disk file.

DEBUG Error Messages

ERROR CODE	DEFINITION
BF	Bad Flag. Attempt made to alter a flag, but the characters entered were not one of the acceptable pairs of flag values. See the Register command for the list of acceptable flag entries.
BP	Too many Breakpoints. More than ten breakpoints were specified as parameters to the G command. Re-enter the Go command with ten or fewer breakpoints.
BR	Bad Register. The R command was entered with an invalid register name. See the Register command for the list of valid register names.
DF	Double Flag. Two values for one flag were entered. You may specify a flag value only once per RF command.

DEL (DELETE FILES) COMMAND

Brief

Format: **DEL** <**filespec**>
or
ERASE <**filespec**>

Command Location: System

Purpose: Delete specified file(s) from disk

Details

Delete all the files with the filename specified on the default or designated drive.

If the filename is *.*, the prompt `Are you sure (Y/N)?` appears. If a **Y** or **y** is typed as a response, then all files are deleted as requested.

ERASE may be used instead of DEL to invoke this command.

Application

This is a command that you will use quite often. It frees up your disk space when files are no longer needed.

DIR (LIST DIRECTORY) COMMAND

Brief

Format: **DIR [d:] [</x>]**
or
DIR [<filespec>] [</x>]

Command Location: System

Purpose: Displays directory entries on a disk.

Details

If no parameters are present (DIR), all directory entries on the default drive are listed. If only the drive parameter is present (DIR d:), all entries on the disk in the specified drive are listed. If only the filename parameter is present DIR <filename>, all files with the filename specified on the disk in the default drive will be listed. If both parameters are present (DIR d:<filename>), all files with the filename specified on the disk in the drive specified will be listed.

In all cases, files are listed with their size in bytes and the date of the last edit.

The wild card characters ? and * may be used in the filename parameter. Refer to "Referencing Several Files at One Time" on Page 3.6, for information on the use of the wild card characters.

The following invocations of the DIR command are equivalent:

COMMAND	EQUIVALENT
DIR	DIR *.*
DIR FILE	DIR FILE.*
DIR .EXT	DIR *.EXT
DIR .	DIR *.

DIR Switches

Two switches may be given with DIR:

- The /**P** switch selects Page Mode. With /P, display of the directory pauses after the screen is filled. To resume display of output, type any key.

- The /**W** switch selects Wide Display. With /W, only file names are displayed without other file information. Files are displayed five per line.

Application

DIR gives you an immediate reference to the directory entries on a disk. If you have a printer connected to your Z-100, you may want to type <**CTRL-P**>**DIR**<**RETURN**>. This sends a copy of the disk's directory to the line printer, which you may then use as an index to the disk's contents.

DSKCOMP (COMPARE DISK) COMMAND

Brief

Format: **DSKCOMP [d:] [d:]**

Command Location: File

Purpose: Do a track-by-track comparison on two disks to test if the contents of the two disks are identical.

Details

DSKCOMP compares data on two disks (one is an original source disk, the other is a copy made with the DSKCOPY utility) to determine if the data is exactly identical on both disks. DSKCOMP also determines that the location of the data on the two disks are contained in the same respective disk locations. Because of this location check, DSKCOMP and DSKCOPY can only be used on disks that are both five-inch or both eight-inch. However, both disks must have the same number of sides, tracks per side and density. The DSK COMP utility is used to verify that the DSKCOPY utility made an exact duplicate of a source and did not introduce any errors into the copy that it made.

NOTE: At the time a disk copy is created with DSKCOPY, the /V switch may be used to verify that the data is copied correctly.

Invoking DSKCOMP

There are two methods available for you to use DSKCOMP. One method is to invoke the command and sources on the command line. The other method is to invoke DSKCOPY, which then prompts you for the sources. For example:

A: **DSKCOMP RETURN**

produces:

```
                DSKCOMP version 1.02
        Copyright(C) 1982 Zenith Data Systems

Source1 drive name?  (A-D)  _:
```

After responding, you will be asked:

```
Source2 drive name? (A-D) _:
```

NOTE: Source1 and Source2 need not be different disk drives, if you only have a single drive system. Though in this case you should specify different drive names (See Appendix G for details regarding single-drive systems.) However, the disks must be the same size, density, tpi, and sides.

After entering Source2, DSKCOMP will ask you to:

```
Place source1 diskette in drive d: and source2 diskette in d:
Hit RETURN when ready.
```

After placing the correct disks in the drives and pressing **RETURN**, the utility tells you that it is working with the message:

```
Verifying...
```

After a few moments you will see the message `Compare another? (Y/N) <N>`. This indicates that the disk verified is identical. Press **RETURN** for **N** to return to the system prompt. Press **Y** (and then **RETURN**) if you want to compare additional disks. If you get a different message, it means that there was some verification error, which means that the disks were not identical.

The second method of beginning DSKCOMP requires some command line options for the Source1 and Source2 drives. If you should only enter one drive on the command line, that drive is assumed to be the Source1, and you will then be prompted for the Source2 drive. For example:

```
A:DSKCOMP A:   RETURN
```

produces:

```
                    DSKCOMP version 1.01
Copyright(C) 1982 Zenith Data Systems

Source2 drive name?  (A-D) _:
```

After entering Source2, DSKCOMP will ask you to:

```
Place source1 diskette in A: and source2 diskette in d:
Hit RETURN when ready.
```

NOTE: In this message the d: would show the drive name that you entered as Source2.

If you enter:

```
A:DSKCOMP A: B:   RETURN
```

DSKCOMP displays:

```
                    DSKCOMP version 1.02
Copyright(C) 1982 Zenith Data Systems

Place source1 diskette in  A: and source2 diskette in B:
Hit RETURN when ready.
```

Application

The DSKCOMP command is useful after you have used DSKCOPY to create a backup. DSKCOMP double checks what DSKCOPY has copied to make certain that the copy it made is exactly correct. If you store a master and a backup copy of data disks, and the backup was created with DSKCOPY, you can use DSKCOMP to verify that the disks are still identical and that their data content has not degraded after weeks, months, or years. The key to this, of course is that the backup was created from the master with DSKCOPY. Otherwise, disk files may need to be verified separately with FILCOM (see Page 6-67 for details regarding FILCOM).

DSKCOPY (COPY DISK) COMMAND

Brief

Format: **DSKCOPY[</x>] [d:] [d:]**

Command
Location: File

Purpose: Copy one disk's content onto another disk so that the contents of both are identical.

Details

The DSKCOPY program may be requested in two ways. The command line can include the source and destination drives, or the source and destination can be omitted. In the latter case, DSKCOPY prompts you for the source and destination. For example:

```
A:DSKCOPY   RETURN
```

produces:

```
                    DSKCOPY version 1.01
Copyright(C) 1982 Zenith Data Systems

Source drive name? (A-D)    _:
```

Type the letter name of the drive that contains the disk that you want to copy. DSKCOPY continues with:

```
Destination drive name? (A-D)    _:
```

Type the letter name of the drive that contains a blank disk (any information on the destination disk will be overwritten by the format program, effectively erasing all data on the disk).

NOTE: The two disks used for DSKCOPY must both be of the same size, and must be formattable to the same density, with the same number of tracks and sides.

DSKCOPY next tells you to place your source in the drive you specified and to place the destination disk in its specified drive.

```
Place source diskette in d: and destination disk in d:
Hit RETURN when ready.
```

Press the **RETURN** key when you have placed the correct disks in the drives specified. When DSKCOPY begins each of its operations, it tells you. For instance, you see:

```
Formatting destination...
Copying...
```

appear on your screen. `Formatting destination...` is displayed at the start of the formatting operation. `Copying...` is displayed at the start of the copying operation. When all operations have finished, DSKCOPY asks you:

```
Copy another? (Y/N) <N>
```

Pressing any key except Y or y (followed by a **RETURN**) returns you to the system prompt. Pressing Y or y causes DSKCOPY to request the source and destination again.

If the source and destination drives are specified in the command line:

```
A:DSKCOPY A: B:   RETURN
```

DSKCOPY does not ask for source and destination but continues and requests:

```
                DSKCOPY version 1.01
Copyright(C) 1982 Zenith Data Systems

Place source diskette in A: and destination disk in B:
Hit RETURN when ready.
```

When you have placed the correct disks in the correct drives, and pressed **RETURN**, DSKCOPY informs you of its current operation:

```
Formatting destination...
Copying...
```

When DSKCOPY has completed its operations, it will return to the system prompt. (DSKCOPY only asks if you want to "Copy another?" when you do not specify source and destination on the command line).

If you specify only a single drive name on the command line (e.g., DSKCOPY C:), DSKCOPY assumes that drive is the source and then prompts you for the destination.

DSKCOPY Switch

A /V switch is optionally available. The /V switch must be entered directly after DSKCOPY and before the drive designations (if entered). /V causes DSKCOPY to verify that the disk has been copied correctly. When the /V switch has been selected, DSKCOPY will verify the copied data after it formats and copies. The corresponding message is:

```
Verifying...
```

Without the /V switch, you may use DSKCOMP after DSKCOPY if you want to verify that the copy is identical to the original. DSKCOMP is covered on Page 6.53.

NOTE: Because DSKCOPY creates an exact duplicate of the original the data in each physical disk sector is identical. DSKCOPY format operation does not "gather" the bad sectors of the disk up so that they cannot be used. This means that DSKCOPY can not copy properly if the destination disk has a bad sector. If a bad sector error does occur, it is wise to reformat the disk with the FORMAT program (with /V switch) and then use the newly reformatted disk only with programs other than DSKCOPY.

CAUTION: Make certain you have a backup copy or an original copy of any disk that you plan to reformat. All data on a disk is effectively erased by the FORMAT program.

Application

This program helps making a backup copy of your valuable disks easier. Backups are very important and you should get into the habit of making them.

EDLIN (LINE EDITOR) COMMAND

Brief

Format: **EDLIN [<filespec>]**

Command Location: File

Purpose: Edits line of text files.

Details

EDLIN is extensively covered in Chapter Eight.

Invoking EDLIN

To invoke EDLIN, enter:

EDLIN <filespec>

If the file specified exists, EDLIN loads the file into memory. If the whole file is loaded, EDLIN returns the message `End of input file` and an asterisk (*) prompt.

If the file is larger than EDLIN's work area (buffer), EDLIN loads as much as it can; then returns the asterisk (*) prompt, but not the `End of input file` message. You may now edit the existing file.

NOTE: When creating a new file, you must consider on what drive a new file should be saved. The command which ends the editing session and saves the file does not allow parameters. Therefore, if the drive is not designated during EDLIN invocation, the file will be saved on the default drive.

Commands

EDLIN commands belong to two types: Intraline and Interline. A brief description of the interline commands is covered below. See Page 8.5 for intraline command information.

Append Lines (A) Command

Format: **[<n>]A**

Append lines from the input file to the editing buffer.

Delete Lines (D) Command

Format: **<line>,<line> D**

Delete the specified lines and all lines in between.

Edit Line (<line>) Command

Format: **<line>**

When the line number <line> is entered, EDLIN displays the line number and text, then, on the line below, reprints the line number. That line is ready for editing.

End Editing (E) Command

Format: **E**

Save the edited file on disk, rename the original file as filename.BAK, then exit EDLIN to the Z-DOS operating system.

Insert Text (I) Command

Format: <**line**> **I**

Insert line(s) of text immediately before the specified <line>. EDLIN remains in this insert mode until a CTRL-Z or a CTRL-C is entered.

List Text (L) Command

Format: <**line**>,<**line**> **L**

List the specified range of lines, including the two lines specified.

Quit Editing (Q) Command

Format: **Q**

Quit the editing session. Do not save any editing changes. Exit to the Z-DOS operating system. No .BAK file will be created.

Replace Text (R) Command

Format: <**line**>,<**line**> **[?] R**<**string**><**CTRL-Z**><**string**>

Replace all occurrences of the first <string> in the specified range with the second <string>.

Search Text (S) Command

Format: <**line**>,<**line**> **[?] S**<**string**>

Search the specified range of lines for the specified string.

Write Lines (W) Command

Format: **[<n>]W**

Write lines from the editing buffer to the output file.

EXE2BIN (.EXE TO BINARY FILE CONVERSION) COMMAND

Brief

Format: **EXE2BIN <filespec> [d:][<filename>][<.ext>]**

Command Location: File

Purpose: Convert .EXE files to binary form.

Details

The first parameter is the input file. If no extension is given, it defaults to the .EXE extension. The second parameter is the output file. If no drive is given, the drive of the input file is used. If no filename is given, the filename of the input file is used. If no extension is given, .BIN is used for the extension.

The input file must be in a valid .EXE format produced by the linker. The "resident", or actual code and data part of the file, must be less than 64K. There must be no STACK segment. Two kinds of conversion are possible depending on the specified initial IP:

1. If IP is not specified, a pure binary conversion is assumed. If segment fix-ups are necessary, the following prompt appears:

   ```
   Fix-up needed - base segment (hex):
   ```

   By typing a legal hexadecimal number and then **RETURN**, execution will continue.

2. If IP is specified as 100H, then it is assumed the file is to be run as a .COM file ORGed at 100H, and the first 100H of the file is to be deleted. No segment fix-ups are allowed, as .COM files must be segment relocatable.

   If IP does not meet one of these criteria or meets the .COM file criterion, but has segment fix-ups, the following error message is displayed:

   ```
   File cannot be converted
   ```

   To produce standard .COM files with the MACRO-86 assembler, you must both ORG the file at 100H and specify the first location as the start address (this is done in the END statement).

   For example:

   ```
       ORG     100H
   START:
       .
       .
       .
       END     START
   ```

FILCOM (COMPARE FILES) COMMAND

Brief

Format: **FILCOM [<s1>[,<s2>][,<list>][</x>...]]**

Command Location: File

Purpose: To compare one file (source) to another to see if they match and to create a file that contains a list of differences between the two files

Details

FILCOM is covered extensively in Chapter Nine.

Invoking FILCOM

The FILCOM (file compare) utility compares two files. The differences between the two files are output to a third file for inspection later.

FILCOM can be invoked one of two ways.

Method 1:

Enter:

A: **FILCOM RETURN**

Method 2:

Enter:

A: **FILCOM <s1>[,<s2>][,<list>][</x>...] RETURN**

Commands

Commands to FILCOM consist of responses to three prompts for file specifications, plus optional switches. The file specifications may be entered one at a time as the prompts appear, or all at once as part of the FILCOM invoke command (Method 2).

File Specifications

All file specifications take the form:

[d:]filename.ext

where **d:** is the letter of a disk drive.

where **filename** is a 1-8 character name of the file.

where **.ext** is a 1-3 character extension to the filename.

Prompts

If Method 2 is used to invoke the command, FILCOM displays no prompts. If Method 1 is used (or if Method 2 is invoked, but with an illegal filename or the name of a nonexistent file for the first source file), FILCOM displays its banner followed by a series of prompts:

```
Source1 filename [.ASM]:
```

Enter the name of one of the files you want compared. The default extension .ASM is assumed if no extension is given.

```
Source2 filename [source1.BAK]:
```

Enter the name of the other file you want compared to Source1. Pressing **RETURN** will default to the same <filename> as the Source1 file, but with a .BAK extension. A <filename> followed by a RETURN will cause a default compare of Source1 to <filename>.BAK (a .BAK default).

```
List filename [source1.DIF]:
```

Enter the name of the file to receive the list of differences. FILCOM defaults to the name given for Source1 with a default .DIR extension.

FILCOM recognizes the following default extensions:

Prompt	Extension	Effect
Source 1	.ASM	Default for Source1 filename. May be overridden.
	.EXE, .OBJ, .COM	Causes default to binary compare
Source 2	.BAK	Default for Source2 filename. May be overridden.
List	.DIF	Default for List filename. May be overridden.

Semicolon (;)

The semicolon character (;) also selects the default responses to the Source2 and List prompts. Unlike the RETURN key, once you enter the semicolon, none of the remaining prompts are given and the comparison begins.

Switches

FILCOM supports five switches. A switch must always be preceded by a slash. FILCOM switches are one of two types: source compare or binary compare.

Source Compare Switches

/A Force a source compare of files with filename extensions .OBJ, .EXE, and .COM. FILCOM defaults to binary compare on files with these filename extensions. Files with any other filename extensions default to source compare.

/C Include comments in compare. A comment in one of the files is considered to start with a semicolon (;), and end with an end-of-line character. Default is exclude comments from source compare.

/<n> (where <n> is a number from 1 through 9). <n> specifies how many consecutive lines in the two files must be the same before FILCOM considers that the two files match at that point. <n> defaults to 3.

/S Include spaces and tabs in compare. Default is exclude spaces and tabs from source compare.

/B Force binary compare of files that default to source compare (files without the filename extensions .OBJ, .EXE, or .COM). Sources are compared byte-by-byte, instead of line-by-line as in a source compare. Differences are output to the List file as an offset location and with the differing bytes in hexadecimal.

Binary Switch Compare

Application

FILCOM gives you a method to see which of several files is your most current copy by pointing to the place in the file where information begins to differ. It can tell you that two files are different. You should check to see what those differences are in order to determine which of the files you need to save.

FORMAT (FORMAT DISK) COMMAND

Brief

Format: **FORMAT [d:][</x>...]**

Command Location: File

Purpose: Prepare a disk by writing a map on it that Z-DOS uses to locate places on the disk.

Switches:
- /C (rewrites reserved sectors, clear FATs and directory)
- /M (format media as single-sided)
- /N (no prompts—suppress prompts)
- /S (copy system files during format)
- /V (verifies the format and gathers up bad sectors so they cannot be used)

Details

The FORMAT utility writes a magnetic map on the recording surface of the disk in the drive specified. It initializes the directory and file allocation tables. The boot loader is copied onto track 0, starting at Sector 1. (This occurs whether or not the /S is given.)

If the disk's format were visible it would appear as:

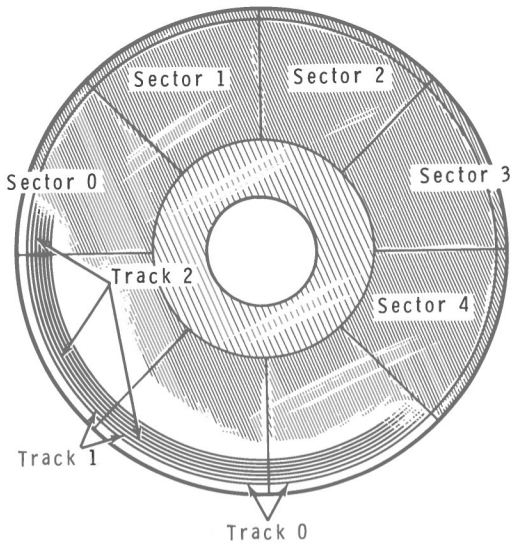


Figure 6.3
Layout of Disk Sectors and Tracks

The number of tracks per disk side would vary depending on the type of disk being used – 5.25-inch 48 tpi gives 40 tracks per disk side, 5.25-inch 96 tpi gives 80 tracks per side, and 8-inch disks are fixed to an industry standard of 77 tracks per side.

Different types of disks require different formats. FORMAT assumes that the disk is to be formatted on both sides. If the disk needs to be formatted on only one side, FORMAT is told to do so with the /M switch. Double-density writes more data patterns (recorded signals) on each track of the disk than single-density, which makes these patterns closer together than single-density.

All Z-DOS disk formats have an essential structure in common:

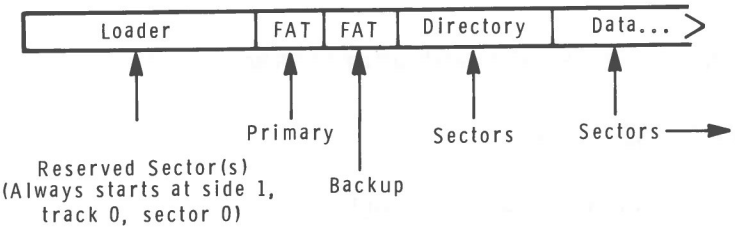


Figure 6.4

Basic Primary Sector Structure for Initialized Disks

This figure shows how the information on a disk would appear to be organized if you could see it all in consecutive order.

For all options selected, except the /C switch (see Page 6-71), FORMAT first initializes each track on the disk. It next goes back to sector 0, track 0, side 1, and writes the Boot Loader code to the first 512 data byte locations on the disk.

FORMAT next writes one dummy FAT. Actually, only the first byte (which is reserved) of the FAT is written. This FAT byte tells the system how many sides are to be formatted on the disk (FORMAT determines tpi from the DIP switch on the Z-207 controller card). It writes:

FFH—for 48 tpi single-sided
FEH—for 48 tpi double-sided
FDH—for 96 tpi double-sided

For 8-inch disks, the FAT ID byte is ignored. FORMAT assumes double-sided unless the /M switch is used. FORMAT formats 8-inch single-sided disks as single-density and formats 8-inch double-sided disks as double-density.

Invoking FORMAT

FORMAT may be entered to the system prompt with a variety of options; or FORMAT may be entered without selecting any options and assumes defaults.

The FORMAT command line without options appears as:

A: **FORMAT RETURN**

When this has been entered, FORMAT responds with:

```
Format version 1.5

Insert new diskette for drive d:
and strike any key when ready
```

When FORMAT is entered in this fashion, it assumes that the disk to be formatted is in the default drive.

To format a disk on a drive other than the current default, the drive name should be specified in the command line:

A: **FORMAT <d:> RETURN**

will cause the disk in the drive named by <d:> to be formatted.

Several options are available in the form of switches.

The /C switch causes FORMAT to rewrite the boot loader, FATs and directory. With the /C switch selected the disk is not initialized (the tracks are assumed to have already been initialized), therefore the disk must have been previously formatted at some earlier time.

The /M switch causes FORMAT to format a disk as single-sided instead of double-sided, (the default).

The /N switch causes FORMAT to skip all of the prompts. When selected, FORMAT displays its banner `Format version 1.5`, formats the disk in the destination specified, and then returns the system prompt when it finishes.

The /S switch causes FORMAT to copy all of the operating system files on the disk in the default drive to the newly formatted disk. The files that are copied are:

IO.SYS
Z-DOS.SYS
COMMAND.COM
ALTCHAR.SYS

These four files are copied in the order shown.

The /V switch causes FORMAT to check the format to see that it was written correctly. Any bad sectors that are found are mapped out of the available disk space so that they cannot be used for data storage by the operating system. At the end of the format operation, if bad sector(s) are found, a message is displayed telling the byte count for the bad sectors found.

Completion Messages

When FORMAT has finished formatting a disk it displays a short message that briefly reports what it has done and then asks if you have more disks to format. Some examples of FORMAT's completion message might be:

for 5.25-inch (48 tpi) with no switches selected:

```
322560 bytes total disk space
322560 bytes available on disk
```

for 5.25-inch (48 tpi) with the /S switch selected:

```
322560 bytes total disk space
 20992 bytes used by system
301568 bytes available on disk
```

for 5.25-inch (48 tpi) with the /M switch:

```
160256 bytes total disk space
160256 bytes available on disk
```

for 5.25 (48 tpi) with both /M and /S switches:

```
160256 bytes total disk space
 20992 bytes used by system
139264 bytes available on disk
```

for 8-inch with no switches:

```
1250304 bytes total disk space
1250304 bytes available on disk
```

for 8-inch with the /S switch:

```
1250304 bytes total disk space
 22528 bytes used by system
1227776 bytes available on disk
```

All of the completion messages in FORMAT take this form, though the byte count given may vary widely.

Immediately after the completion message, FORMAT asks:

```
Format another (Y/N)?
```

Pressing **N** or **n** for "no" returns you to the system prompt. Responding with a **Y** or **y** for "yes" will cause the program to run again (using the same options previously selected).

The space allocated for the Directory and Data Sectors varies with the type of disk format. In Figures 6.5 through 6.9, the different sector allocations for the initial disk sectors on side 1, track 0, sector 0 (up to their first data sector) are shown for each of the current Z-DOS formats.

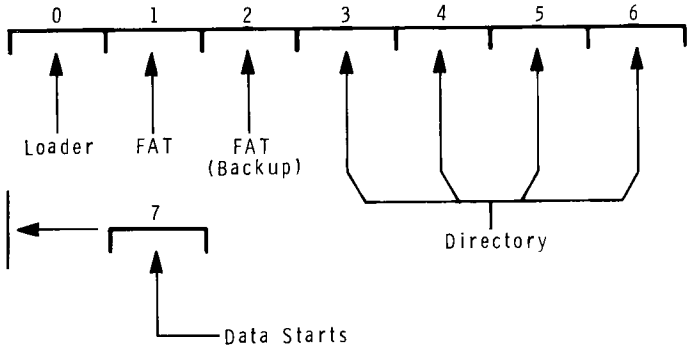


Figure 6.5

5.25-inch Single-sided Double-density 48 tpi Disk Format

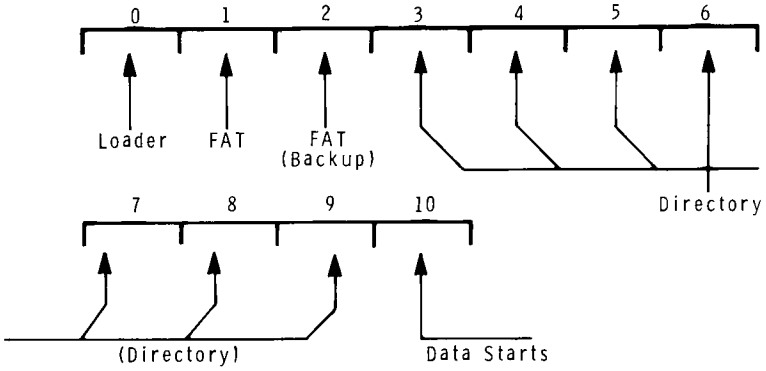


Figure 6.6

5.25-inch Double-sided Double-density 48 tpi Disk Format

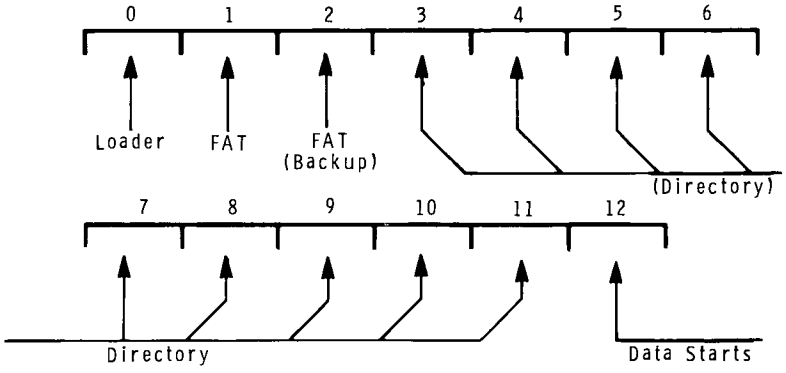


Figure 6.7

5.25-inch Double-sided Double-density 96 tpi Disk Format

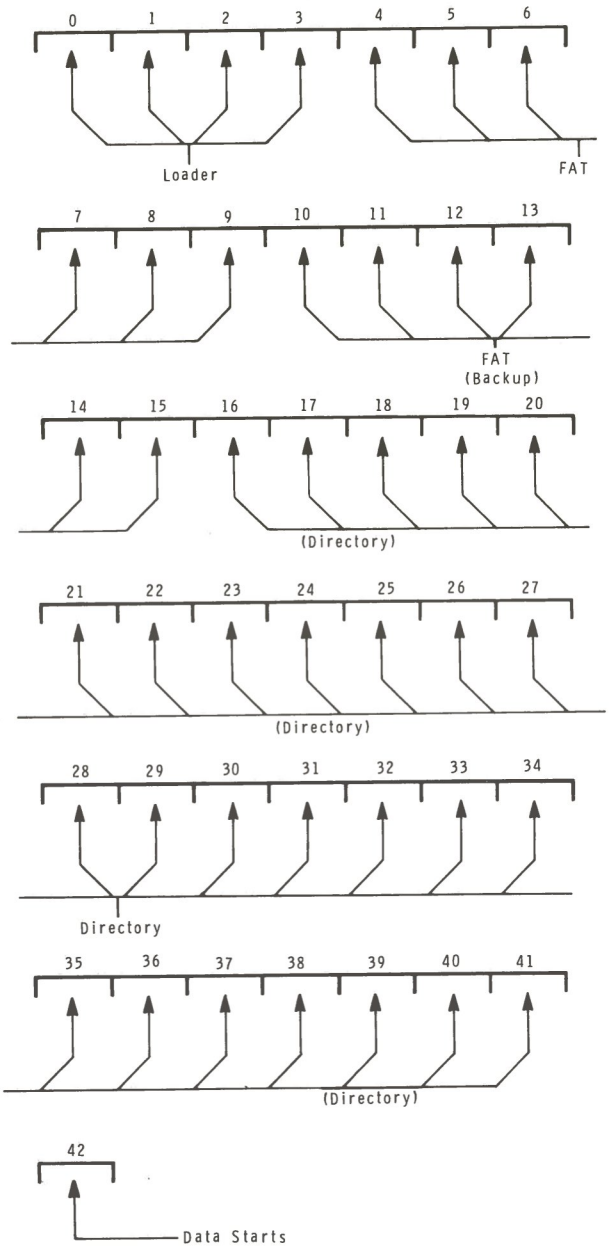


Figure 6.8

8-inch Single-sided Single-density 77 Track Disk Format

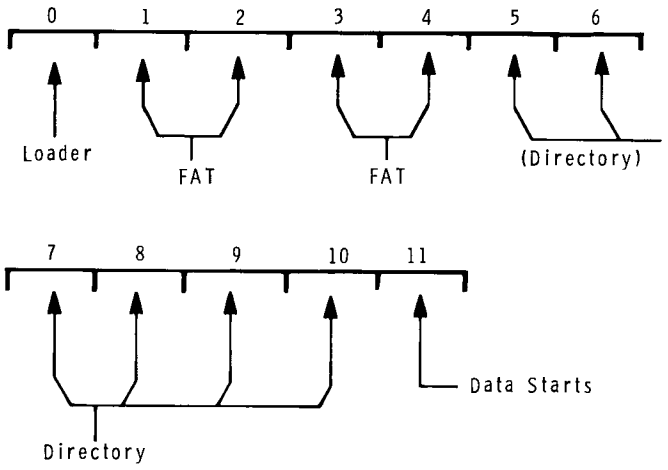


Figure 6.9

8-inch Double-sided Double-density 77 Track Disk Format

Application

FORMAT is one of the most basic programs in Z-DOS. The map that it magnetically draws on the disk is used by Z-DOS to determine where to read or write information. It allows you the option of single- or double-sided disks and places system files on the disks that you want to make bootable. FORMAT is the first program that you run before using a brand new disk.

LIB (MACRO LIBRARY MANAGER) COMMAND

Brief

Format: **LIB [<library><operations>,<listing>]**
or
LIB @<filespec>

Command Location: File

Purpose: Creates and maintains an indexed master file, or library, of code modules that may be deleted, extracted, appended or replaced by a new module from that master file. Also, new library files may be created.

Details

LIB is covered extensively in Chapter Twelve.

There are three methods that can be used to invoke LIB:

1) **LIB** can be entered alone at the command line and you will be prompted for a library file, the operation to be performed, and the name of a cross reference list file.

2) LIB can be invoked by specifying a library file, the operation to be performed, and the name of a cross reference list file in the form:

A: **LIB <library><operations>,<listing>**

3) LIB can be invoked at the command line followed by an at-sign (@) and filespec for a response file, in the form:

A: **LIB @<filespec>**

TYPE (Prompt)	DEFINITION (Response)
<library> (Library File)	Filename of library file to be manipulated (default: filename extension .LIB)
<operations> (Operations)	Command character(s) followed by module name(s) or object filename(s) (default action: no changes. Default object filename extension: .OBJ)
<listing> (List file)	Filename for a cross reference listing File (default: NUL; no file)
@<filespec>	Response filespec, preceded by an at- sign, where the response file contains a list of responses, one for each LIB prompt. If the number of responses is less than the number of LIB prompts, LIB assumes the defaults for the missing responses. If only a <library> filename followed immediately by a semicolon (;) is contained in the response file, LIB performs a consistency check on the library file.
NOTE: The essential difference between the object modules handled by LIB and an object file, is that an object module has no drive specification (<d:>) and no extension (<.ext>). Because of this, distinct object modules require that they be given filenames that are unique to the library file where they are contained.	

Table 6.2

Summary of Command Types (Prompts)

KEY	ACTION
+	Append an object file as the last module
–	Delete a module from the library
*	Extract a module and place in an object file
;	Use default responses to remaining prompts
&	Extend current physical line; repeat command prompt
@	Designates following filespec as a response file
CTRL-C	Abort library session.

Table 6.3
Summary of Command Characters

Application

The LIB utility can manage all of the object code modules that you need on a regular basis (up to 500). This keeps them handy but still out of the way. It is difficult to read a directory that contains many small files. LIB makes the quantity manageable. You just ask for the module that you need, when you need it.

Different libraries may be built by the LIB utility which means that you can have a different library of modules for different types of work.

LINK (MACRO LINKER) COMMAND

Brief

Format: **LINK [<filenames>[</x>]]**
or
LINK @<filespec>

Command Location: File

Purpose: To combine object files (and object modules from a library, if desired) into a relocatable Run file.

Details

LINK is covered extensively in Chapter Eleven.

Invoking LINK

There are three methods that may be used to invoke LINK:

1) LINK may be invoked by itself from the system command line, which causes it to prompt for the necessary information:

```
A:LINK  RETURN
```

LINK's prompts are:

PROMPTS	RESPONSES
`Object Modules [.OBJ]:`	.OBJ files to be linked, separated by blank spaces or plus signs (+). If plus sign is last character entered, prompt reappears. (no default — response required)
`RunFile [Object-file.EXE]:`	Filename for executable object code. (default: first-Object- filename.EXE)
`List File [NUL.MAP]:`	Filename for listing (default: NUL.MAP)
`Libraries [.LIB]:`	Filenames to be searched, separated by blank spaces or plus signs (+). If plus sign is last character entered, prompt reappears. (defaults: no search)

Table 6.4
LINK Prompts

The allowed switches for LINK are:

SWITCH	ACTION
/DSALLOCATE	Load data at high end of Data Segment. Required for Pascal and FORTRAN programs.
/HIGH	Place Run file as high as possible in memory. Do not use with Pascal or FORTRAN Programs.
/LINENUMBERS	Include line numbers in List file.
/MAP	List all global symbols with definitions.
/PAUSE	Halt linker session and wait for carriage return key.
/STACK:<n>	Set fixed stack size to <n> in Run file.

Table 6.5
Link Switches

LINK's command characters are:

CHARACTER	DESCRIPTION
+	The plus sign (+) separates entries and extends the current physical line after the "Object Modules" and "Libraries" prompts. (Blank spaces may be used to separate object modules.) Enter a large number of responses by entering a plus sign/RETURN at the end of the physical line (to extend the logical line).
;	A single semicolon (;), followed immediately by RETURN at any time after the first prompt, selects the default responses to the remaining prompts.
CTRL-C	Use CTRL-C at any time to abort the link session. If you enter an erroneous response, such as the wrong filename or an incorrectly spelled filename, press **CTRL-C** to exit.

Table 6.6
LINK Command Characters

2) LINK may be invoked on the system command line followed by the filenames to be affected and a series of optional switches:

A: **LINK <filenames>[</x>] RETURN**

where <filenames> expands to: <object-list>,<runfile>,<listfile>, <lib-list>; where <object-list> is a list of object modules, separated by plus signs; where <runfile> is the name of the file to receive the executable output; where <listfile> is the name of the file to receive the listing; where <lib-list> is a list of library modules to be searched; and where </x> are optional switches, which may be placed following any of the response entries (just before any of the commas or after the <lib-list>).

3) LINK may be invoked from the command line followed by a response file's name:

A: **LINK @<filespec> RETURN**

where <filespec> is a response file that contains answers to the prompts shown under method one, in the form:

<obj1> <obj2> <obj3>...
</x></x>...
<runfile>
<listfile>
<library1> + <library2>...

MAKE (NEW SOFTWARE INSTALLATION) COMMAND

Brief

Format: **MAKE [<filespec>] [d:]**

Command Location: File

Purpose: Create backup copies of new software.

Details

MAKE is a utility that makes it easier for you to start using a new software package. The MAKE program is an intelligent batch processing facility with a limited command set to provide a variety of task handling requirements.

Invoking MAKE

To invoke MAKE, Enter:

A: **MAKE RETURN**

or

A: **MAKE <filename.ext> RETURN**

from any of your disk drives (with a Z-DOS system disk in the A drive). The system disk in drive A: is necessary so that MAKE can use required system files such as: IO.SYS, Z–DOS.SYS, COMMAND.COM and ALTCHR.SYS if the %SYSTEM command is used.

MAKE begins by displaying information and then requesting:

```
Do you wish to continue? (Y/N)<N>
```

MAKE then asks:

```
Are the destination diskettes single sided? (Y/N)<N>
```

NOTE: There are more files present on the ZDOS distribution disks than can fit on a single sided diskette. To backup the distribution disks you will need double sided disks and answer no (N) to the above question.

MAKE continues:

```
Place Distribution Disk I in drive A:
Press RETURN when ready...
```

Remove any disk that is currently in your A drive and place Distribution Disk I in that drive. Close the drive door and press **RETURN**.

If no <filename.ext> was specified on the command line, MAKE attempts to open a file named COPYFILE.DAT on the A drive. If A does not contain COPYFILE.DAT, MAKE continues as if it found a COPYFILE.DAT and it contained the command COPY *.* (copy all files).

For both methods, there is one additional option. You may specify the destination drive (<d:>) where the MAKE created copy is to be placed. If no <d:> is given as MAKE's destination, MAKE assumes the B drive.

Z-DOS COMMANDS

Command Reference Guide

The following is a list of commands that may be used in a COPYFILE.DAT that is used by MAKE.

%CONCISE This suppresses all of the filenames during copying. %CONCISE resets the verbose flag (also see %VERBOSE).

%DOC Used to insert Remarks or Comments into COPYFILE.DAT. %DOC is ignored by MAKE.

%ECHO Echos all commands read from COPYFILE.DAT to the console. Normally no commands are echoed. This is primarily a debugging tool for this utility.

%ESC[<c>] Sets any ASCII character (<c>) except a space equal to ASCII 27H (Escape) for a %TYPE command. %ESC ignores all spaces in its command line and looks for the first occurence of a character up to the RETURN <LF>. If no character is found, it defaults to backslash (\).

When %ESC is not included in the data file, or when no <c> is specified, the backslash (\) is the default and has a 27H value. For example:

%TYPE \E

is interpreted as ESC E, which clears the screen.

%NEXT <filename> Closes the current copy file, executes a %WAIT, and then opens <filename> on the disk in drive A. If no filename is specified, it opens COPYFILE.DAT. This command is used when multiple processes are to be performed or when a new disk needs to be placed in the A drive.

%NOECHO Resets the %ECHO command. No commands are echoed to the console.

%NOSYS Prompts you to replace the disk (if any) in drive B with a blank disk, and then formats the disk as a data disk (produces the same result as FORMAT with no switches, see Page 6.63).

%SYSTEM Prompts you to replace the disk (if any) in drive B with a blank disk and then formats that disk as a system disk (produces the same result as FORMAT/S/V, see Page 6.75).

%TYPE <message> Types the message following the command onto the screen.

%VERBOSE This causes the name of each file that gets copied to be sent to the console as it is being copied. The message appears like:

```
Copying <filename> <ext>
```

where <filename> and the <ext> of the file being copied are separated by one or more spaces (this has the same effect as COPY *.*, see Page 6.35). This sets the verbose flag. (Also see %CONCISE.)

%WAIT [<message>] Executes like %TYPE but displays the message `Type RETURN when ready...` after the <message> and waits until you press **RETURN**.

Entries in the COPYFILE that are not valid commands (those listed above) are assumed to be valid filenames that exist on that disk. Files are copied one at a time onto the current disk in the B drive.

Only one filename is allowed per line. The filename may consist of any valid combination of wild cards that Z-DOS allows (e.g., B?S?C.*, or FORM*.??M, etc.).

All commands must be separated from other text by a space or RETURN.

If a read or write error occurs anytime during the MAKE procedure, the appropriate error message will be displayed on the console screen and MAKE will exit to the system prompt.

The following is an example of a COPYFILE.DAT that copies files from two distribution disks.

COPYFILE.DAT on distribution disk one would contain:

```
%SYSTEM
*.BAS
%TYPE Insert distribution diskette 2
%NEXT
```

COPYFILE.DAT on distribution disk two would contain:

```
*.BAS
%NOSYS
*.DAT
```

During execution of these two files, you would see the following on the screen:

```
A: MAKE
Place distribution disk 1 in drive A:
Hit RETURN when ready...
Place a blank diskette in drive B:
Hit RETURN when ready...
Insert distribution disk 2
Hit RETURN when ready...
Place a blank diskette in drive B:
Hit RETURN when ready...
A:
```

A further (and more elaborate) example is the two files used for duplicating your Z-DOS Distribution Disks. On Disk I, the SYSCOPY.DAT file is used by MAKE. On Disk II, the DATCOPY.DAT file is also used by MAKE.

Application

MAKE takes you through all of the most important steps that you need to perform after receiving new software from Zenith Data Systems. It does not require you to read lengthy descriptions of what to do each time you receive a new software package. Because of its prompts, even first-time users can protect their investments by creating backup copies of their software with little or no instruction.

MAP (PHYSICAL TO LOGICAL DRIVE NAME MAPPING) COMMAND

Brief

Format: **MAP [</x>][d:n]**

or

MAP ?

Command Location: File

Purpose: Allows you to reassign the logical drive names to physical drive units.

Details

The MAP utility allows you to reassign the logical drive names so that they refer to different physical drives. This alters the default one-to-one mapping that exists from physical (both real and imaginary) to logical drives within the operating system. The term *real* refers to a logical drive name that is assigned to a drive that is physically connected to your computer; and *imaginary* refers to a logical drive name that does not have its corresponding drive actually connected. (If you have only one 5-inch drive, it is assigned as drive A. A is a real drive, and B refers to an imaginary drive since there is no corresponding drive.)

Invoking MAP

MAP will probably not be used very often, so an additional option has been provided:

A: **MAP ? RETURN**

returns a brief description of MAP's use, including the command line syntax:

```
MAP [/Z;/I] [d: n]

     /Z-Reset table to default values

     /I-Make drive imaginary

     d:-Logical drive name

     n- Physical drive number
```

To invoke MAP in order to see the current logical to physical assignments, you can type:

A:**MAP RETURN**

MAP with no options specified causes a table of assignments to display:

```
                    MAP Version 1.01
          Copyright(C) 1982 Zenith Data Systems

Physical Drive  Kind         Logical Names   Type  Assigned
 0              5 Inch drive  A:             Real  A:
 1              5 Inch drive  B:             Real  B:
 2              8 Inch drive  C:             Real  C:
 3              8 Inch drive  D:             Real  D:
```

The display shown here reflects the default name assignments for a system having two 5-inch and two 8- inch disk drive units connected to the Z-207 controller board.

In the example, "Physical Drive" is the number of the drive; "Kind" is the device type; "Logical Name" is the default drive names in the range from A through D; "Type" is either Real or Imaginary (an imaginary type occurs when only one of the two possible drives is connected; e.g., only one 5-inch drive is connected, the other is shown as imaginary); and "Assigned" shows the current logical name to physical drive assignments.

To reassign drive names with MAP, you type:

A:**MAP <d:> <n> RETURN**

where <d:> is the logical name for the drive; and <n> is the number of the drive.

For example, if you typed:

A:**MAP B: 0 RETURN**

you would in effect be naming the drive that boots as drive A with the name of B. If you have not made any other name assignments, your drive responds as both A: and B:. The map table would appear like:

```
                MAP version 1.01
        Copyright(C) 1982 Zenith Data Systems

Physical Drive  Kind          Logical names  Type      Assigned
 0              5 Inch drive  A:,B:          Real       A:
 1              5 Inch drive                 Real       B:
 2              8 Inch drive  C:             Real       C:
 3              8 Inch drive  D:             Imaginary  D:
```

Notice that the type for physical drive 3 shows as imaginary. An imaginary type drive occurs when only one of two possible drives is connected to a device. Therefore, the table shown previously would only occur on a Z-100 having two 5.25-inch drives and one 8-inch drive (the C: drive) connected to the Z-207 controller.

If you typed the command:

A: **MAP/Z RETURN**

the logical to physical assignments are reset to the original boot condition. If you issued this command on the same machine that produced the previous table, MAP would display:

```
                    MAP Version 1.01
            Copyright(C) 1982 Zenith Data Systems

Physical Drive  Kind          Logical Names   Type      Assigned
 0              5 Inch drive  A:              Real      A:
 1              5 Inch drive  B:              Real      B:
 2              8 Inch drive  C:              Real      C:
 3              8 Inch drive  D:              Imaginary D:
```

Physical drive 0 has a logical name of A and physical drive 1 would again be named as the B drive. The /Z switch always resets the drive name assignments to their initial values (the names that the drives are assigned when an initial boot is performed).

Application

When you boot your system, the logical drive names (A, B, C, and D) are assigned to the default device unit. If, however, one of your drives becomes unreliable, it might be necessary to reassign the drives' names. This might be necessary for an application program that requires a particular drive to be present in order to operate. If the required drive is faulty, the logical name could be reassigned to a drive that could be used instead.

MASM (MACRO-86 ASSEMBLER) COMMAND

Brief

Format: **MASM [<filenames>[/x>...]]**

Command Location: File

Purpose: To assemble macrocode source documents into a file that may be relocated and loaded by LINK, in order to create an executable file.

Details

MACRO–86 and MASM are covered extensively in Chapter Ten.

MASM (MACRO-86) may be invoked by two methods. The first invocation uses no command line parameters or arguments, the second specifies arguments and optional parameters.

Invoking MASM Method One:

When MASM is invoked with no arguments or parameters:

A: **MASM RETURN**

A series of prompts appear in the order that is shown in the Command Prompt Summary Table on Page 6.99. Several options are available that may be entered as switches. The switches are shown in the Command Switches Summary Table on Page 6.99. There are also two command characters that may be used during MASM's operation. These two command characters are described in the Command Characters Table on Page 6.100.

PROMPT	RESPONSES
Source filename [.ASM]:	List .ASM file to be assembled. (No default: filename response required)
Object filename [source.OBJ]	List filename for relocatable object code. (Default: source-filename.OBJ)
Source listing [NUL.LST]:	List filename for listing file. (Default: no listing file)
Cross reference [NUL.CRF]	List filename for cross-reference file (used with CREF to created cross-reference listing). (Default: no cross-reference file)

Table 6.7

MASM Command Prompt Summary

SWITCH	ACTION
/D	Produce a listing on both assembler passes.
/O	Show generated object code and offsets in octal radix on listing.
/X	Suppress the listing of false conditionals. Also used with the .TFCOND directive.

Table 6.8

MASM Command Switches Summary

CHARACTER	DESCRIPTION
;	Use a single semicolon (;), followed immediately by RETURN at any time after responding to the first prompt (from Source filename on) to select default responses to the remaining prompts. This feature saves time and overrides the need to enter a series of RETURNs.
NOTE: Once the semicolon has been entered, you can no longer respond to any of the prompts for that assembly. Therefore, do not use the semicolon to skip over some prompts. For this, use RETURN.	
CTRL-C	Use CTRL-C at any time to abort the assembly. If you enter an erroneous respose, such as the wrong filename or an incorrectly spelled filename, you must press **CTRL-C** to exit MACRO-86. Then reinvoke MACRO-86 and start over. If the error has been typed and not entered, you may delete the erroneous characters, but for that line only.

Table 6.9
MASM Command Characters

Invoking MASM Method Two:

The second method that may be used to invoke MASM is:

A: **MASM <filenames>[/x...] RETURN**

where <filenames> is defined as: <source>,<object>,<listing>, <crossref> <source> is the source filename; <object> is the name of the file to receive the relocatable output; <listing> is the name of the file to receive the listing; <crossref>is the name of the file to receive the cross reference output; and [</x>...] are optional switches, which may be placed following any of the response entries (just before any of the commas or after the <cross-ref>, as shown.

To select the default for a field, simply enter a second comma without space in between (the defaults are shown in the preceding tables).

Directives and information on the macro codes and macro language are given in Chapter Ten.

Application

The number of uses for assembly language is almost limitless. The MASM command (MACRO-86) is an effective tool in the development process. Thorough study of Chapter Ten as well as a study of the 8088 (8086) instruction set and architecture, can make the Z-100 an even more effective tool should you choose to do software development of your own.

The MACRO-86 information contained in this manual is not meant to be a guide on "How to Program", nor is it a manual of the 8088 (8086) opcodes. However, this manual in conjunction with one of the many sources now available should make assembling code for the Z-100 as easy as possible.

PAUSE COMMAND

Brief

Format: **PAUSE [remark]**

Command Location: System

Purpose: PAUSE suspends execution of a BATCH file.

Details

During the execution of a batch file, if you need to change disks, the PAUSE command suspends execution until you type any key, except CTRL-C.

When COMMAND encounters PAUSE, it prints:

```
Strike a key when ready . . . _
```

Pressing any key except CTRL-C resumes execution of the batch file. If you type CTRL-C, another prompt is displayed:

```
Terminate batch job (Y/N)?
```

If you type "Y" in response to this prompt, execution of the remainder of the batch command file is aborted and control returns to the operating system command level. Therefore, PAUSE can be used to break a batch file into pieces, allowing you to end the batch command file at an intermediate point.

An optional comment may be entered on the same line as PAUSE. You may want to enter some meaningful message to be displayed at some point in the execution of the batch file. For example, you may want to change disks in one of the drives. The PAUSE command will suspend execution, and the optional comment will prompt you for the appropriate action. The prompt will be displayed before the `Strike a key when ready...` message.

Application

If you begin to make your own batch files PAUSE is indispensible for some procedures. PAUSE can allow you to make many useful procedures that might otherwise require more than one batch file to complete, because some type of break might be required to allow you to change disks, look-up information, or turn switches on or off between certain steps.

PRINT (PRINT ASCII FILES) COMMAND

Brief

Format: **PRINT <filespec>[</x>...]**
or
PRINT ?

Command Location: File

Purpose: Print ASCII files, automatically expanding tabs to eight columns.

Details

Invoking PRINT

The PRINT utility is invoked followed by either a question mark, or by the filespec of the ASCII file to be printed. With the question mark:

A: **PRINT ? RETURN**

a brief description of PRINT is displayed on the screen. This description includes the command syntax and the switches that are available for use with PRINT.

When PRINT is followed by the filespec of the ASCII file that is to be printed:

A: **PRINT <fielspec> RETURN**

PRINT sends the content of the ASCII file to the peripheral device that is configured as the PRN device (further details regarding CONFIGUR are covered on Page 6.22). PRINT expands all tabs in the ASCII file to eight columns.

PRINT Switches

There are several optional switches that may be used individually or in combination with PRINT. The switch options available for use with PRINT are:

/C<n>— where <n> is an integer and /C causes PRINT to print <n> copies of the ASCII file. The default that is used when this switch is not selected is one copy.

/F — causes PRINT to print an FF at the end of the file that is invoked with PRINT. Without this swtich, no FF is printed at the end of the file.

/L<n> — where <n> is an integer and /L causes PRINT to use a left margin set to the <n>th column. The default that is used when this switch is not selected sets the left margin to zero.

/M — forces (maps) lowercase characters contained in the ASCII file to uppercase characters. When this switch is not selected, characters are printed as they are written in the file(i.e., no mapping occurs).

/P<n> — where <n> is an integer and /P causes PRINT to print <n> lines of text per page before inserting a form feed. The default that is used when this switch is is not selected is set to sixtyone lines per page.

/R<n> — where <n> is an integer and /R causes PRINT to use a right margin set to the <n>th column. The default that is used when this switch is not selected sets the right margin to 131.

/W — causes PRINT to pause (wait) between pages. When this switch is not selected, PRINT does not pause after each form feed but instead prints the file without stopping.

RDCPM (READ CP/M FILE) COMMAND

Brief

Format: **RDCPM <filespec> [d:][filename.ext]**
or
RDCPM DIR <d:>[<filespec>]
or
RDCPM ?

Command Location: File

Purpose: Read a CP/M formatted file and copy that file to a Z-DOS formatted file.

Details

Invoking RDCPM

RDCPM has two primary functions: read a CP/M formatted disk's directory, and copy CP/M formatted files into a Z-DOS format.

RDCPM may be invoked in several ways. If you type:

A: **RDCPM RETURN**

RDCPM returns the message:

```
          RDCPM version 1.13
Copyright(C) 1982 Zenith Data Systems

Usage:  RDCPM source [destination]
-or-    RDCPM DIR drive
```

If you would like more detailed instructions about RDCPM's function, type:

A: **RDCPM ? RETURN**

A description of RDCPM appears on your screen, along with the message shown above.

Reading a CP/M Disk's Directory

To read a directory of a disk that has been formatted with CP/M, you should type:

A: **RDCPM DIR <d:>[<filespec>] RETURN**

where <d:> is the drive name where the CP/M disk is located; and where [<filespec>] may be any legal CP/M filename and extension or wild cards that represent the file(s) that you want to see in the directory. The RDCPM banner will display on the screen. Below that, the directory of the CP/M formatted disk will appear in CP/M directory form (with filenames and extensions in four columns) on your screen: with filenames and extensions displayed in four columns.

Copying a CP/M File

To copy a CP/M formatted file or several files (wild cards are acceptable), you type:

A: **RDCPM <filespec> [d:][filename.ext] RETURN**

where <filespec> is the CP/M source file; [d:] is the optional drive name for a Z-DOS formatted disk; and [filename.ext] is an optional name for the file that is being copied to the Z-DOS destination.

NOTE: RDCPM used in this manner (to copy a CP/M file from a source disk that has been formatted with CP/M, to a destination disk that has been formatted with Z-DOS) is exactly like the Z-DOS COPY command. It does require that the source is CP/M format and that the destination is Z-DOS format.

During the copy operation, RDCPM will display:

```
          RDCPM version 1.13
Copyright(C) 1982 Zenith Data Systems
```

directly above the list of filenames that are copied. This list appears similar to the way COPY displays the names that it is copying.

Application

RDCPM expands the usefulness of Z-DOS by allowing you to use the features of Z-DOS while taking advantage of documents and files created with CP/M.

REM (REMARK) COMMAND

Brief

Format: **REM [remark]**

Command Location: System

Purpose: To place remarks inside a batch file which will be displayed on the monitor's screen when encountered.

Details

REM displays comments entered on the same line as the command "REM" when encountered during execution of batch file.

The REM command has no other effect. The only delimiters for the comment are any one of the three legal delimiters to start the comment (space, tab, comma).

Application

Creating a batch on a computer and remembering what it does later on is no problem when you can annotate the steps with REM. It is also useful for leaving an audit trail for others. Someone could open the batch file with EDLIN, read the entries along with the remarks, and if they are well written, be able to come away knowing just what the batch file is supposed to do.

It is also very useful when a batch file can instruct you or display messages about some ongoing batch operation. The REM command displays text on the screen that can guide or prompt you with information during the command execution.

REN (RENAME FILE) Command

Brief

Format: **REN** <**filespec**> <**filespec**>
or
RENAME <**filespec**> <**filespec**>

Command Location: System

Purpose: Change the name of the first parameter (<filespec>) to the second parameter (<filespec>).

Details

The first parameter (<filespec>) must be given a drive designation if the file's disk resides in a drive other than the currently logged (default) drive. Any drive designated for the second parameter (<filename>) will be ignored. The file will remain on the disk where it currently resides.

The wild card characters, question mark (?) and asterisk (*), may be used in both parameters. Use of a wild card in one parameter, but not in the other, is illegal. Use of a wild card character produces a one-to-one correspondence between files specified in the first parameter (filespec) and those specified by the second parameter (filename).

For example, the following command changes the names of all files with the .LST extension to similar names with the .PRN extension:

```
A:REN *.LST *.PRN  RETURN
```

The next example causes the file ABODE on drive B to be renamed ADOBE:

```
A:RENAME B:ABODE ?D?B?  RETURN
```

This renames the file ABODE on drive B to ADOBE. The file remains on drive B.

RENAME is a synonym for the REN command, and executes in exactly the same way.

Application

Whether you want to rename your backup files with a name to instantly show you that they are backups (such as by using a .BAK extension), or if you have made a mistake in naming a file, REN is a quick way to make the change. You will probably find that a different name may be more suitable for your filing system or while manipulating a file. REN can fix either quickly.

SYS (COPY SYSTEM FILES) COMMAND

Brief

Format: **SYS [d:]**

Command Location: File

Purpose: Replace the system files on the specified drive with the system files on the default drive.

Details

SYS is normally used to update or replace the system on a formatted disk. An entry for d: is required.

The files transferred are copied in the following order:

Z-DOS.SYS
IO.SYS

Neither COMMAND.COM or ALTCHAR.SYS are transferred. IO.SYS and Z-DOS.SYS are both set as hidden files that do not appear when the DIR command is executed.

Application

At some point in time if you are a seasoned programmer, the SYS command can place a new IO.SYS on a disk that you want because you need different I/O functions than those that are standard. It also can be of use to rewrite the system files onto a disk that is formatted.

TIME COMMAND

Brief

Format: **TIME [<hh>[:<mm>[:<ss>]]]**

Command Location: System

Purpose: Display and change the time.

Details

If the TIME command is entered without any parameters, then the following message is displayed:

```
Current time is <hh>:<mm>:<ss>.<cc>
Enter new time:_
```

Type **RETURN** if you do not want to change the time shown. When Z-DOS first requests the time after the system has been booted, the default time is used.

Optionally, a new time may be given as a parameter to the TIME command as in:

```
A:TIME  8:20:00  RETURN
```

The new time must be entered using numerals only: letters are not allowed. The allowable parameters are:

<hh> =00-23
<mm> =00-59
<ss> =00-59

The hour, minute, and second entries must be separated by colons.

<cc>, which is not an enterable parameter, shows the time to the nearest hundredths of a second.

TIME controls the rollover of date. If the system is left on for a period of time, the data will be correct, even if a new month starts. Z-DOS is programmed to change dates correctly, whether the month has 31, 30, 29, or 28 days. (Z-DOS handles leap years, too).

Error Messages

Z-DOS uses whatever time is entered as the new time as long as the parameters and separators are legal. If the parameters or separators are not legal, Z-DOS returns the message:

```
Invalid time, enter as hh:mm:ss (24 hour clock)
Enter new time:_
```

and waits for you t enter a legal time.

Application

Like DATE, TIME helps you keep track of files or archive purposes. It defines the creation time down into smaller sequences than DATE does. This helps you distinguish which file is which if you have been working all day developing a file such as a program. At the end of the day you have several copies on different disks with each of the files in a different state of development. The time stamp in conjunction with the date stamp, gives you a distinct reference as to when a file was created.

Additionally, it can help you to keep track of the time you have spent working on a project.

TYPE (DISPLAY FILE) COMMAND

Brief

Format: **TYPE** <**filespec**>

Command Location: System

Purpose: Displays a file's contents on the console without altering that file.

Details

Type displays the contents of the file on the console screen. The contents of the file are copied from the disk to the CRT, but the file is not loaded into memory. The only formatting performed is that the tab stops are expanded to eight blank columns.

Application

This command can be used to examine a file without modifying it. Use DIR to find the name of a file and EDLIN to alter the contents of a file. The only formatting performed by TYPE is that tabs are expanded to spaces consistent with tab stops every eight column. Displaying binary files causes control characters — including bells, formfeeds, and escape sequences — to be sent to your computer.

Part 3

Comprehensive Utility Details

The DEBUG Command

DEBUG

Brief

Format: **DEBUG [<filespec>]**

Command
Location: File

Purpose: A tool for software development that provides a controlled environment to load, examine, and change a program.

DEBUG Commands:

Compare	C<range> <address>
Dump	D[<address>[L <value>]]
	D[<range>]
Enter	E<address>[<list>]
Fill	F<range> <list>
Go	G[=<address>][<address>...]
Hex	H<address> <address>
Input	I<value>
Load	L[<address>[<drive><record><record>]]
Move	M<range> <address>
Name	N<filename>[<filename>]
Output	O<value> <byte>
Quit	Q
Register	R[<register name>]
Search	S<range> <list>
Trace	T[=<address>][<value>]
Unassemble	U[<address>[L <value>]]
	U[<range>]
Write	W[<address>[<drive><record><record>]]

DEBUG

The DEBUG Command

Details

DEBUG is a resident debugger program used to provide a controlled testing environment for binary code files (executable object files) that may not function correctly. EDLIN is used to alter source files (program files that are not yet assembled); DEBUG is EDLIN's counterpart for binary files. DEBUG eliminates the need to reassemble a program each time to see if problems have been fixed or if other changes are working properly. DEBUG allows you to alter the file at an address or in a CPU register, then execute the altered instructions as an immediate check of the validity of the changes.

All commands given to DEBUG may be aborted at any time by pressing **CTRL-C. CTRL-S** will suspend the display so that you can read it before the output scrolls away. Any key other than CTRL-C and CTRL-S will restart the display.

Invoking DEBUG

To invoke DEBUG, enter:

A: **DEBUG [<filespec>] RETURN**

The filespec is an optional entry with the invocation command. However, you may eventually need to specify a file to be debugged. The ability to enter the invocation command without a filespec permits single-drive users to use DEBUG and allows absolute disk addresses to be used.

If filespec is specified:

A: **DEBUG <filespec> RETURN**

COMMAND.COM will load DEBUG: DEBUG will load the specified file starting at 100 hex in the lowest available segment. The CX register will be loaded with the number of bytes loaded. DEBUG will return the banner:

```
DEBUG Version n.nn Copyright Microsoft, Inc.   1981
>
```

The DEBUG prompt is a > rather than the d: of COMMAND, or the * of EDLIN.

You may now review and alter the file.

If filespec is not entered, you would type:

A: **DEBUG RETURN**

COMMAND.COM will load DEBUG, and DEBUG will return the banner as described above. DEBUG is now ready to accept your commands. Since no filename was specified, you will have to work with current memory. You can optionally bring in files or sectors from the disk system.

DEBUG Commands

The DEBUG commands consist of a single letter followed by parameters. Additionally, the control function characters and special editing commands described on Pages 4.15 and 4.19 apply in DEBUG program operation.

If a syntax error occurs in a DEBUG command, DEBUG reprints the command line and indicates the error with an up-arrow and the word "Error." For example:

```
>dcs:100 cs:110
          ^ Error (not a valid hex digit)
```

All commands and parameters may be entered in either upper or lower case. Any combination of upper and lower case may be used in commands.

The DEBUG commands are summarized in the following table and are described in detail with examples following the description of command parameters.

COMMAND	FORMAT
Compare	C<range> <address>
Dump	D[<address>[L <value>]]
	D[<range>]
Enter	E<address>[<list>]
Fill	F<range> <list>
Go	G[=<address>][<address>...]
Hex	H<address> <address>
Input	I<value>
Load	L[<address>[<drive><record><record>]]
Move	M<range> <address>
Name	N<filename>[<filename>...]
Output	O<value> <byte>
Quit	Q
Register	R[<register name>]
Search	S<range> <list>
Trace	T[=<address>][<value>]
Unassemble	U[<address>[L <value>]]
	U[<range>]
Write	W[<address>[<drive><record><record>]]

Table 7.1
DEBUG Commands

DEBUG Command Parameters

All DEBUG commands, as Table 7.1 shows, accept parameters, except the Quit command. Parameters are always entered after the command. They may be separated by delimiters (spaces or commas), but a delimiter is required *only* between two consecutive hexadecimal values. Thus, the commands

dcs:100 110

d cs:100 110

and **d,cs:100,110**

are equivalent. Also, entries may be either upper or lower case or any combination.

PARAMETER	DEFINITION
drive	A one digit hexadecimal value to indicate which drive a file will be loaded from or written to. The valid values are 0-3. These values designate the drives as follows: 0=A, 1=B, 2=C, 3=D.
byte	A two digit hexadecimal value to be placed into or read from an address or register.
record	A one to three digit hexadecimal value used to indicate the logical record number on the disk and the number of disk sectors to be written or loaded. Logical records correspond to sectors, however their numbering differs since sectors represent the entire disk space.
value	A hexadecimal number of up to four digits that is used to specify a port number or the number of times a command should repeat its functions.

The DEBUG Command

PARAMETER	DEFINITION
address	A two part designation consisting of either an alphabetic segment register designation or a four digit segment address plus an offset value. The segment designation or segment address may be omitted, in which case the default segment will be used. DS is the default segment for all commands except G, L, T, U, and W, for which the default segment is CS. All numeric values are hexadecimal. For example: **CS:0100** or **04BA:0100** The colon is required between the segment designation (whether numeric or alphabetic) and the offset.
range	Either two addresses: e.g., <address> <address>; or one address, an L, and a value: e.g., <address> L <value> where <value> is the number of locations the command should operate on. Examples: **CS:100 110** or **CS:100 L 11** not **CS:100 CS:110** ^ Error The limit for <range> is 10000 hex. To specify the <value> of 10000 hex with four digits, enter 0000 (or 0).
list	A series of byte values or of strings. List must be the last parameter on the command line. Example: **fcs:100 42 45 52 54 41**

PARAMETER	DEFINITION
string	Any number of characters enclosed in quote marks. Quote marks may be either single (') or double("). Within strings, the opposite set of quote marks may be used freely as literals.

If the delimiter quite markes must appear within a string the quite marks must be doubled.

For example:

```
        'This as a ' 'string' ' is okay.'
or      'This as a "string" is okay.'
but     'This as a 'string' is not.'

        "This as a 'string' is okay."
or      "This is a ""string"" is okay."
but     "This as a "string" is not."

but     "This as a ' 'string' ' is not necessary."
and     'This as a ""string"" is not necessary.'
```

The ASCII values of the characters in the string are used as a list of byte values.

DEBUG Command Descriptions

C(COMPARE) COMMAND

Brief

Format: **C<range> <address>**

Purpose: Compare the portion of memory specified by <range> to a portion of the same size beginning at <address>.

Details

If the two areas of memory are identical, there is no display and DEBUG returns with the DEBUG prompt. If there *are* differences, they are displayed as:

```
<address1> <byte1> <byte2> <address2>
```

The following commands have the same effect:

CCS:100,200 CS:300

or

CCS:100L101 CS:300

Each command compares the block of memory from 100 to 200H with the block of memory from 300 to 400H of the CS segment.

D (DUMP) COMMAND

Brief

Format: **D[<address>[L <value>]]**
or
D[<range>]

Purpose: Displays the memory contents of either a single address, a range of addresses, or the number of locations specified by value beginning at the address specified.

Details

If only a single address is specified, the contents of 80 (hexadecimal) bytes are displayed on 80 column format screens.

If a range of addresses is specified, the contents of the range are displayed.

If the D command is entered without parameters, the result is the same as if you specified a single address, except that the display begins at the current location in the DS (Data Segment).

The dump is displayed in two portions:

1) A hexadecimal dump. Each byte is shown in hexadecimal value.

2) An ASCII dump. The bytes are shown in ASCII characters.

Non-printing (invisible) characters are shown by a period (.) in the ASCII portion of the display.

If your display is an 80 column format, each display line shows sixteen bytes with a hyphen between the eighth and ninth bytes.

Each displayed line begins on an even boundary (a 16-byte boundary for the 80 column format).

NOTE: The first line may display fewer than 16 bytes if the address or start address of the range is not on a boundary. In this case, the second line will begin on a boundary, and the first line will contain fewer bytes than the rest of the displayed lines.

Example:

With an 80 column format, if you enter the command:

```
>dcs:100 110
```

DEBUG would display this same information in the format:

```
04BA:0100  42 45 52 54 41 20 54 00-20 42 4F 52 4C 41 4E 44 BERTA T. BORLAND
04BA:0110 20
```

If you enter the command:

```
>D
```

the display will be formatted as described above, but 80H bytes will be displayed. Each line of the display will begin with an address; incremented from the address on the previous line by 10H (80 column display). Each subsequent D (entered without parameters) displays the bytes immediately following those last displayed.

If you enter the command:

>**DCS:100 L 20**

the display will be formatted as described above, but all of the bytes for 20H lines will be displayed.

If you enter the command:

>**DCS:100 115**

the display will be formatted as described above, but all the bytes in the range of lines from 100H to 115H in the CS segment will be displayed.

DEBUG Command Descriptions

E (ENTER) COMMAND

Brief

Format: **E<address>[<list>]**

Purpose: Enter an <address>, display its contents, and wait for input.

Details

If the optional list of hexadecimal values is entered, the replacement of byte values occurs automatically. (If an error occurs, no byte values will be changed.)

If the address is entered without the optional list, DEBUG displays the address and its contents. Then it repeats the address on the next line and waits for your input.

The E command accepts four types of input:

1) Replace a byte value with a value you type in. Simply type the value after the current value.

   If the value typed in is not a legal hexadecimal value or if more than two digits are typed, the illegal or extra character(s) will not be echoed.

2) Press the **SPACE BAR** to advance to the next byte.

   If you want to change the value, simply enter the new value (as described in 1 above).

   If you space beyond an eight-byte boundary, DEBUG will start a new display line with the address displayed at the beginning.

3) Type a hyphen (-) to return to the preceding byte.

   If you decide to change a byte previous to the current position, typing the hyphen will return the current position to the previous byte. When the hyphen is typed, a new line is started with the address and its byte value displayed.

4) Press the **RETURN** key to terminate the E command. The RETURN key may be pressed at any byte position.

For example, if you enter:

```
>ECS:100
```

DEBUG displays in this format:

```
04BA:0100 EB._
```

Now, if you wanted to change this value to 41:

```
>04BA:0100EB.41_
```

If you want to step through the subsequent bytes, press the space bar and you see:

```
04BA:0100  EB.41  10.  00.  BC._
```

If you decide to change BC to 42:

```
04BA:0100  EB.41  10.  00.  BC.42_
```

Realizing that 10 should be 6F, enter the hyphen as many times as needed to return to byte 0101 (value 10), and then replace 10 with 6F:

```
04BA:0100  EB.41 10. 00. BC.42-
04BA:0102  00.-_
04BA:0101  10.6F_
```

Pressing the RETURN key ends the E command and gives the DEBUG ">" prompt.

F (FILL) COMMAND

Brief

Format: **F<range>[<list>]**

Purpose: Fill the addresses in the <range> with the values in the <list>.

Details

If the range contains more bytes than the number of values in the list, the list will be used repeatedly until all bytes in the range are filled.

If the list contains more values than the number of bytes in the range, the extra values in the list will be ignored.

If any of the memory in the range is not valid (bad or nonexistent), the error will be propagated into all succeeding locations. The F command does not abort as the E command does. The F command is a multiple version of the E command in that it allows you to change more than one address at a time.

Example:

If you enter the command:

F04BA:100 L 100 42 45 52 54 41

DEBUG fills memory locations 04BA:100 through 04BA:1FF with the bytes specified. The five values will be repeated until all 100H bytes are filled. When the fill is finished, DEBUG simply displays the ">" prompt.

G (GO) COMMAND

Brief

Format: **G[=<address>][<address>...]**

Purpose: Execute the program.

Details

If the G command is entered alone, the program will execute as if you had run the program outside DEBUG.

If =<address> is set, execution begins at the address specified.

If the segment designation is omitted from =<address>, only the instruction pointer is set.

If the segment designation is included in =<address>, both the CS segment and the instruction pointer are set. The equal sign (=) is required so that DEBUG can distinguish the start address from the breakpoint addresses.

With the other optional address(es) set, execution stops at the first address encountered, regardless of that address's position in the list of addresses (the breakpoint address will not be executed).

If the program contains jump or branching instructions, you may want to set more than one breakpoint to halt execution no matter which branch the program takes. When program execution reaches a breakpoint, the registers, flags, and decoded instruction will be displayed for the last instruction executed. (The result is the same as if you entered the **R** (Register) command for the breakpoint address.) You may set up to ten breakpoints. Breakpoints should only be set at addresses containing the first byte of an 8086 opcode.

If more than 10 breakpoints are set, DEBUG returns the `BP Error` message.

DEBUG Command Descriptions

Your stack pointer must be valid and have six bytes available for this command. The G command uses an IRET instruction to cause a jump to the program under test. Your stack pointer is set, and your flags, code segment register, and instruction pointer are pushed on your stack.

NOTE: If your stack is not valid or is too small, the system will crash.

An interrupt code (0CCH) is placed at the specified breakpoint address(es). When an instruction with the breakpoint code is encountered, all breakpoint addresses are restored to their original instructions.

If execution is not halted at one of the breakpoints, the interrupt codes will not be replaced with the original instructions.

Example:

If you enter the command:

>**GCS:7550**

The program will execute up to the address 7550 in the CS segment. Then DEBUG will display the registers and flags. The G command is then terminated.

If you enter the command:

>**G**

the program will execute just as if you had entered the filename at COMMAND.COM command level. The difference is that program execution will begin at the current instruction rather than at the usual starting point.

H (HEX ARITHMETIC) COMMAND

Brief

Format: **H<address> <address>**

Purpose: Perform hexadecimal arithmetic on the two parameters.

Details

The H command causes DEBUG to first add the two parameters, then subtract the second parameter from the first. The results of the arithmetic will be displayed on one line; first the sum, then the difference.

Example:

If you entered:

```
H10A 19F
```

DEBUG will perform the calculations and return the results:

```
02A9    FF6B
```

I (INPUT) COMMAND

Brief

Format: **I<value>**

Purpose: Input and display one byte from the port specified by <value>.

Details

The I command inputs a single byte. A 16-bit port address is allowed.

Example:

Enter the command:

```
>I2F8
```

Assume also that the byte at the port is 42H. DEBUG will input the byte and display the value:

```
>I2F8
42
```

L (LOAD) COMMAND

Brief

Format: **L[<address>[<drive> <record> <record>]]**

Purpose: Load a file into memory.

Details

Set CX to the number of bytes read. The file must have been named either with the DEBUG invocation command or with the N command (see Name on Page 7.24). Both the invocation and the N commands format a file name properly in the normal format of a file control block at CS:5C.

If the L command is given without parameters, DEBUG loads the file into memory beginning at address CS:100 and sets CX to the number of bytes loaded.

If L is entered with <address> only, the file named by the N command is loaded.

NOTE: If the file named by the N command or the invocation is a .HEX file, the L command entered by itself will load the file beginning at the address specified in the .HEX file. If the L command includes the option address, DEBUG will add the address specified in the L command to the address found in the .HEX file to determine the start address for loading the file.

If the L command is given with parameters, the loading begins at the memory address specified. This invocation does not require a filename since it reads sectors directly from the disk. When the L command is invoked with parameters, BX:CX is set to the number of bytes read.

If L is entered with all parameters, absolute disk sectors are loaded, not a file. The records are taken from the drive specified (the drive designation is numeric here — 0=A, 1=B, 2=C, 3=D); DEBUG begins loading with the first record specified; and continues until the number of sectors specified in the second record have been loaded.

Example:

If you enter the commands:

```
A:DEBUG
>N<filename>
>L
```

DEBUG loads the file and simply returns the ">" prompt.

If you want to load only portions of a file or certain records from a di k, you enter:

```
>L04BA:100 2 0F 6D
>
```

DEBUG will load 109 records beginning with logical record number 15 into the portion of memory beginning at address 04BA:0100. When the records have been loaded, DEBUG will simply return the > prompt.

M (MOVE) COMMAND

Brief

Format: **M<range> <address>**

Purpose: Move the block of memory specified by <range> to the location beginning at the <address> specified.

Details

Overlapping moves (moves where part of the block overlaps some of the current addresses) are always performed without loss of data.

Addresses that could be overwritten are moved first. The sequence for moves from higher addresses to lower addresses is to move the data beginning at the block's lowest address and work towards the highest. The sequence for moves from lower addresses to higher addresses is to move the data beginning at the block's highest address and work towards the lowest.

NOTE: If the addresses in the block being moved will not have new data written to them, the data there before the move will remain. The M command really copies the data from one area into another, in the sequence described, and writes over the new addresses. That is why the sequence of the move is important.

Example:

If you enter the command:

>MCS:100 110 CS:500

DEBUG will first move address CS:110 to address CS:510, then CS:10F to CS:50F, and so on until CS:100 is moved to CS:500. DEBUG will display the ">" prompt. You should enter the **D** command, using the address entered for the M command, to review the results of the move.

N (NAME) COMMAND

Brief

Format: **N<filename>[<filename>...]**

Purpose: Set filenames.

Details

The Name command performs two distinct functions, both having to do with filenames. First, Name is used to assign a filename for a later Load or Write command. Thus, if you invoke DEBUG without naming and file to be debugged, then the N<filename> command must be given before a file can be Loaded. Second, Name is used to assign filename parameters to the file being debugged. In this case, Name accepts a list of parameters that are used by the file being debugged.

These functions overlap. Consider the following set of DEBUG commands:

```
>NFILE1.EXE  RETURN
>L  RETURN
>G  RETURN
```

Because of the two-pronged effect of the Name command, the following happens:

1. **N** assigns the filename FILE1.EXE to the filename to be used in any later Load or Write commands.

2. **N** also assigns the filename FILE.EXE to the first filename parameter to be used by any program that is later debugged.

3. **L** loads FILE.EXE into memory.

4. **G** causes FILE.EXE to be executed with FILE.EXE as the single filename parameter (that is, FILE.EXE is executed as if FILE FILE.EXE had been typed at the command level.)

A more useful chain of commands might appear as:

```
>NFILE1.EXE  RETURN
>L  RETURN
>NFILE2.DAT FILE3.DAT  RETURN
>G  RETURN
```

Here, Name sets FILE1.EXE as the filename for the subsequent Load command. The Load command loads FILE1.EXE into memory, and then the Name command is used again, this time to specify the parameters to be used by FILE1.EXE. Finally, when the Go command is executed, FILE1.EXE is executed as if FILE1 FILE2.DAT FILE3.DAT had been typed at the Z-DOS command level. Note that if a Write command were executed at this point, then FILE1.EXE--the file being debugged--would be saved with the name FILE2.DAT! To avoid such undesired results, you should always execute a Name command before either a Load or a Write.

There are four distinct regions of memory that can be affected by the Name command:

```
CS:5C   FCB for file 1
CS:6C   FCB for file 2
CS:80   Count of characters
CS:81   All characters entered
```

A File Control Block (FCB) for the first filename parameter given to the Name command is set-up at CS:5C. If a second flename parameter is given, then an FCB is setup for it beginning at CS:6C. The number of characters typed in the Name command (exclusive of the first character, "N") is given at location CS:80. The actual stream of characters of the letter "N") begins at CS:81. Note that this stream of characters may contain switches and delimiters that would be legal in any command typed at the Z-DOS command level.

Application

A typical use of the Name command would be:

```
DEBUG PROG.COM
>NPARAM1 PARAM2/C
>G
>
```

In this case, the Go command executes the file in memory as if the following command line had been entered:

```
PROG PARAM1 PARAM2/C
```

Testing and debugging therefore reflect a normal runtime environment for PROG.COM.

O (OUTPUT) COMMAND

Brief

Format: **O<value>** <**byte**>

Purpose: Send the <byte> specified to the output port specified by <value>.

Details

The O command causes a single byte to be output. A 16-bit port address is allowed.

Example:

If you enter the command:

```
>O2F8  4F
```

DEBUG will output the byte value 4F to output port 2F8.

Q (QUIT) COMMAND

Brief

Format: **Q**

Purpose: Terminate the debugger.

Details

The Q command takes no parameters. The file in memory is not saved. You will see the COMMAND program prompt, d:.

Example:

If you want to end the debugging session, enter:

```
>Q  RETURN
```

DEBUG will terminate, and COMMAND.COM will return the system prompt d:. Actually, the system will return to the drive that was logged when DEBUG was invoked. Therefore, the prompt could be A:, B:, C:, D:.

```
>Q
A:
```

R (REGISTER) COMMAND

Brief

Format: **R[<register name>]**

Purpose: Display the contents of one or more CPU registers.

Details

If no register name is entered, the R command dumps the register save area and displays the contents of all registers and flags.

If a register name is entered, the 16-bit value of that register will be displayed in hexadecimal, then a colon will appear as a prompt. Next you either enter a value to change the register, or press the **RETURN** key if no change is wanted.

The only valid register names are:

AX	BP	SS	
BX	SI	CS	
CX	DI	IP	(IP and PC both refer to
DX	DS	PC	the instruction pointer.)
SP	ES	F	

Any other entry for the register name results in a `BR Error` message.

If you enter **F** as the register name, DEBUG displays a series of two character alphabetic codes.

To alter the flag, you must enter the opposite two letter code. The flags are either set or clear. The Flags with their codes for set and clear are:

FLAG NAME	SET	CLEAR
Overflow	OV	NV
Direction	DN Decrementing	UP Incrementing
Interrupt	EI Enabled	DI Disabled
Sign	NG Negative	PL Plus
Zero	ZR	NZ
Auxiliary Carry	AC	NA
Parity	PE Even	PO Odd
Carry	CY	NC

Table 7.2
Flag Set and Flag Clear Codes

Whenever you enter the command RF, the flags will be displayed in the order shown in a row at the beginning of a line. At the end of the list of flags, DEBUG displays a static hyphen (-). You may enter new flag values as alphabetic pairs. The new flag values may be entered in any order. You are not required to leave spaces between the flag entries.

To exit the R command, press the **RETURN** key. Flags for which new values were not entered remain unchanged.

If more than one value is entered for a flag, DEBUG returns a `DF Error` message.

If you enter a flag code other than those shown above, DEBUG returns a `BF Error` message. In both cases, the flags up to the error in the your list will be changed. Flags at and after the error are not changed.

Example:

If you enter the command:

>**R**

DEBUG will display all registers, flags, and the decoded instruction for the current location.

Assume the location is CS:11A; then DEBUG might display:

```
AX=0E00  BX=00FF  CX=0007  DX=01FF  SP=039D  BP=0000  SI=005C  DI=0000
DS=04BA  ES=04BA  SS=04BA  CS=04BA  IP=011A   NV UP DI NG NZ AC PE NC
04BA:011A  CD21          INT     21
```

If you enter the command:

>**RF**

DEBUG will display the flags:

```
NV UP DI NG NZ AC PE NC -
```

You may now enter any valid flag designation, in any order, and with or without spaces. For example:

```
NV UP DI NG NZ AC PE NC - PLEICY   RETURN
```

DEBUG will respond with the hyphen prompt only. To see the changes, you must enter either the R or RF command:

>**RF**
NV UP EI PL NZ AC PE CY -

You may either press **RETURN** to leave the flags this way or enter different flag values.

S (SEARCH) COMMAND

Brief

Format: **S<range> <list>**

Purpose: Search the range specified for the <list> of bytes specified.

Details

The S command causes a search for a list throughout a specified <range>.

The list may contain one or more bytes, each separated by a space or comma.

If the list contains more than one byte, only the first address of the byte string is returned.

If the list contains only one byte, all addresses of the byte in the range will be displayed.

Example:

If you enter:

```
>SCS:100 110 41
```

DEBUG might return the response:

```
04BA:0104
04BA:010D
>_
```

T (TRACE) COMMAND

Brief

Format: **T[=<address>][<value>]**

Purpose: Execute one instruction and display the contents of all registers, flags, and the decoded instruction.

Details

If the optional =<address> is entered, tracing occurs at the address specified.

If the address includes the segment designation, then both CS and the instruction pointer are specified.

If the address omits the segment designation, only the instruction pointer is specified.

The optional value causes DEBUG to execute and trace the number of steps specified by the value. The T command uses the hardware trace mode of the 8088 (or 8086) microprocessor. Consequently, you may also trace instructions stored in ROM.

Example:

If you enter the command:

```
>T
```

DEBUG will return a display of the registers, flags, and decoded instruction for that one instruction. Assume that the current position is 04BA:011A. DEBUG might return the display:

```
AX=0E00  BX=00FF  CX=0007  DX=01FF  SP=039D  BP=0000  SI=005C  DI=0000
DS=04BA  ES=04BA  SS=04BA  CS=04BA  IP=011A   NV  UP  DI  NG  NZ  AC  PE
04BA:011A  CD21            INT     21
```

If you enter the command:

>T=011A 10

DEBUG will execute ten instructions beginning at 011A in the current segment and display all registers and flags for each instruction as it is executed. The display scrolls away until the last instruction is executed. Then the display stops, and you can see the register and flag values for the last few instructions performed. Remember that CTRL-S will suspend the display at any point so that you can study the registers and flags for any instruction.

U (UNASSEMBLE) COMMAND

Brief

Format: **U[<address>[L <value>]]**
or
U[<range>]

Purpose: Disassemble the bytes and display the source statements that produced the bytes along with the addresses and bytes values.

Details

The display of disassembled code looks like a listing for an assembled file.

If you enter the U command without parameters, the number of bytes depends on the format of the system display.

DEBUG will disassemble 20 (hexadecimal) bytes. The number of disassembled lines displayed will vary depending on the number of bytes generated by each instruction.

If the U command is entered without parameters, disassembly begins at the current location counter in the CS (program) segment.

If you enter the U command with the range parameter, DEBUG will disassemble all bytes in the range up to 20 (hexadecimal) bytes.

If there are fewer bytes in the range than the number displayed when U is entered without parameters, only the bytes in the range will be displayed.

If you enter the U command with the address parameter, DEBUG will disassemble the default number of bytes (20H), beginning at the address specified.

If you enter the U command with the <address> L <value> parameters, DEBUG will disassemble all bytes beginning at the address specified for the number of locations specified by value. Entering the U command with the <address> L <value> parameters overrides the default limit (20H or 10H bytes).

Example:

If you enter the command:

>**U04BA:100**

DEBUG disassembles 16 bytes beginning at address 04BA:0100:

```
04BA:0100  206472      AND   [SI+72],AH
04BA:0103  69          DB    69
04BA:0104  7665        JBE   016B
04BA:0106  207370      AND   [BP+DI+70],DH
04BA:0109  65          DB    65
04BA:010A  63          DB    63
04BA:010B  69          DB    69
04BA:010C  66          DB    66
04BA:010D  69          DB    69
04BA:010E  63          DB    63
04BA:010F  61          DB    61
```

If, however, you entered the command:

>**UCS:100 L 20**

all of the bytes in the twenty lines of instructions beginning at address CS:100 through address CS:11F would be disassembled and displayed.

If you alter the bytes in some addresses, the disassembler alters the instruction statements. You can enter the U command for the locations changed, see the new instructions, and use the disassembled code to edit the source file.

W (WRITE) COMMAND

Brief

Format: **W[<address>[<drive><record><record>]]**

Purpose: Write the file being debugged to a disk file.

Details

If <drive><record><record> is not present, BX:CX should be set to the number of bytes to be written.

If a G or T command was used, CX must be reset before using the W command without parameters.

NOTE: if a file is loaded and modified, the name, length, and starting address are all set correctly to save the modified file (as long as the length has not changed.)

The file must have been named either with the DEBUG invocation command or with the N command (see Name on Page 7.23). Both the invocation and the N commands format a file name properly in the normal format of a file control block at CS:5C.

If the W command is given without parameters, CX must be set to the number of bytes to be written. Then, DEBUG writes the CX number of bytes to the disk file. The debugged file will be written to the disk from which it was loaded. This means that the debugged file will be written over the original file that was loaded into memory.

If the W command is given with parameters, the write begins from the memory address specified. The file is written to the drive specified (the drive designation is numeric here — 0=A, 1=B, 2=C, 3=D). DEBUG writes the file beginning at the logical record number specified by the first record and continues until the number of sectors specified in the second record have been written.

NOTE: Writing to absolute sectors is EXTREMELY dangerous because the process bypasses the file manager.

Example:

If you enter the command:

```
>W
```

DEBUG writes out the file to disk then displays the > prompt:

```
>W
>_
```

If you enter:

```
>WCS:100 1 37 2B
```

DEBUG writes out the contents of memory beginning with the address CS:100 to the disk in drive B:. The data written out will start in disk logical record number 37H and will consist of 2BH records. When the write is complete, DEBUG will display the > prompt:

```
WCS:100 1 37 2B
>_
```

DEBUG Error Messages

During the DEBUG session, you may receive any of the following error messages. All errors terminate the DEBUG command, but not DEBUG.

ERROR CODE	DEFINITION
BF	Bad Flag. You attempted to alter a flag, but the characters entered were not one of the acceptable pairs of flag values. See the Register command for the list of acceptable flag entries.
BP	Too many Breakpoints. You specified more than ten breakpoints as parameters to the G command. Reenter the Go command with ten or fewer breakpoints.
BR	Bad Register. You entered the R command with an invalid register name. See the Register command for the list of valid register names.
DF	Double Flag. You entered two values for one flag. You may specify a flag value only once per RF command.

The EDLIN (Line Editor) Command

EDLIN

Brief

Format: **EDLIN** <**filespec**>

Command
Location: File

Purpose: Create or edit text files.

EDLIN Commands:

Append Lines	<n>A
Delete Lines	<line>,<line> D
Edit Line	<line>
End Editing	E
Insert Text	<line> I
List Text	<line>,<line> L
Quit Editing	Q
Replace Text	<line>,<line> [?] R<string><CTRL-Z><string>
Search Text	<line>,<line> [?] S<string>
Write Lines	<n>W

Details

EDLIN is a line text editor. The text of a file created or edited with EDLIN is divided into lines. Each line may contain up to 255 characters, one of which is an end-of-line character (carriage return, 00DH).

The line numbers are not actually present in the saved text. The numbers are dynamic. When a file is created or edited, the line numbers begin at one and are incremented by one through the end of the file.

When new lines are inserted between existing lines, all line numbers following the inserted text are incremented automatically by the number of lines inserted. When lines are deleted between existing lines, all line numbers following the deleted text are decremented automatically by the number of lines deleted.

Consequently, line numbers always run one through n (last number), incremented by one. For example:

Assume the following file exists:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: Compare the before and after
4: See what happens when you
5: Delete and Insert
6: Line numbers
```

Now, delete line 3; enter:

3 D

Now, list the file; enter:

L

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: *See what happens when you
4: Delete and Insert
5: Line numbers
```

Now, insert two lines between lines 4 and 5; enter:

5 I

5 **(The D and I commands)**
6 **(Use CTRL-C to exit insert mode)**
7 <**CTRL-C**>

Now, list the file again; enter L:

L

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: Delete and Insert
5: (The D and I commands)
6: (Use CTRL-C to exit insert mode)
7: Line numbers
```

To invoke EDLIN, enter:

Invoking EDLIN

EDLIN <filespec>

If the specified file exists, EDLIN loads the file into memory.

If the whole file is loaded, EDLIN returns the message `End of input file` and an asterisk (*) prompt.

If the file is larger than workspace or file buffer, EDLIN loads as much as it can and returns the asterisk (*) prompt but not the End of input file message.

You may now edit the existing file. When you want to edit the part of a larger than memory file not in memory, you must first write out to disk some of the file that is in memory and then append lines into memory. These commands are simply nW, where n is the number of lines written out; and nA, where n is the number of lines appended.

When the editing session ends, the file is saved on the same drive where it was found.

If the file specified does not exist, EDLIN creates the file and returns the message `New file`. Then EDLIN displays the asterisk (*) prompt. You may now create a new file.

EDLIN Command Types and Parameters

These are two types of EDLIN commands: Intraline and Interline.

Intraline Commands

Intraline commands perform edit functions within a single line. The commands used to perform intraline editing are the control function commands and the special editing (escape code) commands.

Interline Commands

Interline commands perform edit functions on whole lines at a time. The interline commands are summarized in the following table and are described in detail with examples following the description of command parameters and the intraline commands.

COMMAND NAME	FORMAT
Append Lines	<n>A
Delete Lines	<line>,<line> D
Edit Line	<line>
End Editing	E
Insert Text	<line> I
List Text	<line>,<line> L
Quit Editing	Q
Replace Text	<line>,<line> [?] R<string><CTRL-Z><string>
Search Text	<line>,<line> [?] S<string>
Write Lines	<n>W

Table 8.1
Interline Commands

The EDLIN (Line Editor) Command

Command Parameters

Each interline command accepts some optional parameters. The following list of the parameters indicates their form. The effect of a parameter depends on the command it is used with. See the discussions of each command in "Interline Commands" for a description of the effect of the parameters on the command.

PARAMETER	DEFINITION
<line>	This indicates a line number you should enter. Line numbers must be separated from other line numbers, other parameters, and the command. Use a comma or space to separate.

It is specified one of three ways:

Number	any integer less than 65534. If a number larger than the largest existing line number is specified, then <line> indicates the line after the last line number.
Period (.)	If a period is specified for the line, it indicates the current line number. The current line is the last line edited, and not necessarily the last line displayed. The current line is always marked by an asterisk (*) between the line number and the first character.
Pound (#)	The pound sign indicates the line after the last line number. Specifying # for the line has the same effect as specifying a number larger than the last line number.

PARAMETER	DEFINITION
RETURN	A RETURN entered without any of the line specifiers listed above directs EDLIN to use a default value appropriate to the command.
?	The question mark parameter directs EDLIN to ask you if the correct string has been found. The question mark is used only with the Replace and Search commands. EDLIN waits for you to respond with either a Y or RETURN for a yes response, or with any other key for a no response before continuing.
<string>	This string represents text to find, text to replace, or text to replace other text. The string parameter is used only with the Search and the Replace commands. Each string is terminated by a **CTRL-Z** or a **RETURN** (see the Replace command for details). No spaces should be left between strings or between a string and its command letter, unless you want the spaces as part of the string.

EDLIN Commands

INTRALINE COMMANDS

<F1> (Copy One Character) Command

Brief

Key: **F1**

Purpose: Copy one character from the template to the input buffer.

Details

Press the **F1** key to copy one character from the template to the input buffer. When the F1 key is pressed, one character is inserted in the buffer and insert mode is automatically turned off if it was on. Use the F1 key to advance the cursor one column across the line.

Application

Assume the screen shows:

```
1: *This is a sample file.
1: *_
```

At the beginning of the intraline edit, the cursor is positioned at the beginning of the line (indicated by the underline). Pressing the F1 key copies the first character (T) to the second of the two lines displayed:

```
        1: *This is a sample file.
<F1>    1: *T_
```

Each time the F1 key is pressed, one more character appears:

```
<F1>    1: *Th_
<F1>    1: *Thi_
<F1>    1: *This_
```

<F2> (Copy Up To Character) Command

Brief

Key: **F2**

Purpose: Copy multiple characters up to a given character.

Details

Press the **F2** key to copy all characters up to a given character from the template to the input buffer. The given character is the next character typed and is not copied or shown on the screen. Pressing the F2 key causes the cursor to move to the single character which is this command's only parameter. If the template does not contain the specified character, nothing is copied. Pressing F2 also automatically turns off insert mode if it is on.

Application

Assume that the screen shows:

```
1: *This is a sample file.
1: *_
```

At the beginning of the intraline edit, the cursor is positioned at the beginning of the line (indicated here by the underline). Pressing the F2 key copies all characters up to the character that is pressed immediately after the F2 key.

```
          1: *This is a sample file.
<F2>p     1: *This is a sam_
```

<F3> (Copy Template) Command

Brief

Key: **F3**

Purpose: Copy template to input buffer.

Details

Press **F3** to copy all remaining characters from the template to the input buffer. Regardless of the cursor position at the time the F3 key is pressed, the rest of the line appears, and the cursor is positioned after the last character on the line.

Application

Assume the screen shows:

```
1:*This is a sample file.
1:*_
```

At the beginning of the intraline edit, the cursor is positioned at the beginning of the line (indicated by the underline). Pressing the F3 key copies all characters from the template (shown in the upper line displayed) to the line with the cursor (the lower line displayed):

```
        1:*This is a sample file.
<F3>    1:*This is a sample file._
```

Also, insert mode is automatically turned off if it was on.

<F4> (Skip One Character) Command

Brief

Key: **F4**

Purpose: Skip over one character in the template.

Details

Press the **F4** key to skip over one character in the template. Each time you press the F4 key, one character is deleted (not copied) from the template. The action of the F4 key is similar to the F1 key, except that F4 skips a character in the template rather than copying it to the input buffer.

Application

Assume the screen shows:

```
1:*This is a sample file.
1:*_
```

At the beginning of the intraline edit, the cursor is positioned at the beginning of the line (indicated by the underline). Pressing the F4 key skips over the first character (T).

```
        1:*This is a sample file.
<F4>    1:*_
```

The cursor position does not move. Only the template is affected. To see how much of the line has been skipped over, press the **F3** key, which moves the cursor beyond the last character of the line.

```
        1:*This is a sample file.
<F4>    1:*_
<F3>    1:*his is a sample file._
```

<F5> (Skip Up To Character) Command

Brief

Key: **F5**

Purpose: Skip multiple characters in the template.

Details

Press the **F5** key to skip over all characters up to a given character in the template. The given character is the next character typed, and is not copied and not shown on the screen. If the template does not contain the specified character, nothing is skipped over. The action of the F5 key is similar to the F2 key, except that F5 skips over (deletes) all the characters in the template up to the character pressed after the F5 key:

```
      1:*This is a sample file.
<F5>  1:*_
```

The cursor position does not move. To see how much of the line has been skipped over, press the **F3** key to copy the template. This moves the cursor beyond the last character of the line:

```
          1:*This is a sample file.
<F5>p     1:*_
<F3>      1:*ple file._
```

SHIFT-<F0> (Quit Input) Command

Brief

Key: **SHIFT-F0**

Purpose: Quit input and flush the input buffer.

Details

Press the **SHIFT-F0** key to flush the input buffer while leaving the template unchanged. SHIFT-F0 also prints a back slash (\), RETURN, and line feed, and turns insert mode off if it was on. The cursor is positioned at the beginning of the line. Press the **F3** key to copy the template to the input buffer just as the line was before SHIFT-F0 was pressed.

Application

Assume the screen shows:

```
1:*This is a sample file.
1:*_
```

At the beginning of the intraline edit, the cursor is positioned at the beginning of the line (indicated by the underline). Assume you want to replace the line by typing:

```
                  1:*This is a sample file.
Sample File       1:*Sample File_
```

Now, to re-edit the line, press **SHIFT-F0:**

```
                  1:*This is a sample file.
<SHIFT-F0>        1:*Sample File\
```

RETURN can now be pressed to keep the original line or to perform any other intraline editing functions. If F3 is pressed, the original template is copied to the input buffer:

```
<F3> This is a sample file._
```

<F6> (Insert Mode) Command

Brief

Key: **F6**

Purpose: Enter insert mode.

Detail

Press the **F6** key to cause entry into insert mode. The current position in the template is not changed. The cursor does move as each character is inserted. However, when you have finished inserting characters, the cursor is positioned at the same character as it was before the insertion began. Thus, characters are inserted **before** the character the cursor points to.

Application

Assume the screen shows:

```
1:*This is a sample file.
1:*_
```

At the beginning of the intraline edit, the cursor is positioned at the beginning of the line (indicated by the underline). Assume you press the **F2** and "**p**" keys:

```
                1:*This is a sample file.
<F2>p           1:*This is a sam_
```

Now press the **F6** key and insert the three characters "s", "o", and "n".

```
                1:*This is a sample file.
                1:*This is a sam_
<F6>son         1:*This is a samson_
```

If you now press the **F3** key, the rest of the template is copied to the line:

```
          1:*This is a samson_
<F3>      1:*This is a samsonple file...
```

If you were to press the **RETURN** key instead, the remainder of the template would be truncated, and the input buffer would end at the end of the insert:

```
<F6>son   RETURN   1:*This is a samson
```

<F7> (Replace Mode) Command

Brief

Key: **F7**

Purpose: Enter Replace Mode.

Details

Press the **F7** key to cause exit from insert mode and entry into replace mode. All characters entered overstrike and replace characters in the template. (Replace mode is the default.) When you start to edit a line, this mode is in effect. Each character typed replaces a character in the template. If the **RETURN** key is pressed, the remainder of the template is truncated.

Application

Assume the screen shows:

```
1:*This is a sample file.
1:*_
```

At the beginning of the intraline edit, the cursor is positioned at the beginning of the line (indicated by the underline). Assume you then press <**F2**>**p,** <**F7**>**son**, and then <**F3**>:

```
                1:*This is a sample file.
<F2>p           1:*This is a sam_
<F7>son         1:*This is a samson_
<F3>            1:*This is a samson file._
```

If you type in characters that extend beyond the length of the template, the remaining characters in the template are automatically appended when you type **F3**.

\ <F0> (New Template) Command

Brief

Key: **F0**

Purpose: Create New Template.

Details

Press the **F0** key to copy the current contents of the input buffer to the template. The contents of the old template are then destroyed. Press **F0** to output an at sign character (@), a RETURN, and a line feed. The input buffer is also emptied and insert mode is turned off.

NOTE: F0 performs the same functions as the SHIFT-F0 key, except that the template is changed and an at-sign character (@) is printed instead of a backslash (\).

Application

Assume the screen shows:

```
1:*This is a sample file.
1:*_
```

At the beginning of the intraline edit, the cursor is positioned at the beginning of the line (indicated by the underline). Assume that you enter <**F2**>**p,** <**F7**>**son,** and then <**F3**>:

```
                1:*This is a sample file.
<F2>p           1:*This is a sam_
<F7>son         1:*This is a samson_
<F3>            1:*This is a samson file._
```

At this point, assume that you want this line as the new template, so you press the <**F0**> key:

```
<F0> 1:*This is a samson file.@
                _
```

Additional editing can now be done using the above new template.

INTERLINE COMMANDS

<line> (Edit Line) Command

Brief

Format: <**line**>

Purpose: The <line> command displays the specified line for editing.

Details

When the line number is entered, EDLIN displays the line number and text. On the line below, it reprints the line number. The line is ready for editing. You may now use any of the intraline commands to edit the line. The existing text of the line serves as the template until the RETURN key is pressed.

If no line number is entered (that is only the RETURN key is pressed), the line after the current line, marked with an asterisk (*), is edited.

If no changes are needed and the cursor position is at the beginning or end of the line, simply press the **RETURN** key.

NOTE: If the RETURN key is pressed while the cursor is in the middle of the line, the remainder of the line is truncated.

Application

Assume the following file exists and is ready to edit.

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: Delete and Insert
5: (The D and I commands)
6: (Use CTRL-C to exit insert mode)
7: *Line numbers
```

Edit line 6; enter:

6 RETURN

```
6: * (Use CTRL-C to exit insert mode)
6: *
```

Move the cursor to the first lowercase "t"; enter <**F2**> **t**

The result is:

```
6: * (Use CTRL-C_
```

Enter insert mode; press <**F6**>, then enter the text:

6 <F6>Careful! RETURN causes truncation of the line! <F7> <F3> RETURN

The result is:

```
6 (Use CTRL-C Careful! RETURN causes truncation of the line! to exit insert mode)
```

D (Delete Lines) Command

Brief

Format: <**line**>,<**line**> **D**

Purpose: The D command deletes the specified lines and all lines in between.

Details

If the first line is omitted, the first line defaults to the current line (the line with the asterisk next to the line number).

If the second line is omitted, the second line defaults to the first line; one line is deleted. When the lines are deleted, the next line after the deleted section becomes the current line and now has the same line number as the first line had before the deletion occurred.

Application

Assume the following file exists and is ready to edit.

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: Delete and Insert
.
.
.
25: (The D and I commands)
26: (Use CTRL-C to exit insert mode)
27:*Line numbers
```

To delete multiple lines, enter <**line**>,<**line**> **D** **RETURN**

5,24 D

The result is:

```
1: This is the sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: Delete and Insert
5:*(The D and I commands)
6: (Use CTRL-C to exit insert mode)
7: Line numbers
```

To delete a single line, enter <**line**>**[,] D**

6 D RETURN
or
6, D RETURN

The result is:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: Delete and Insert
5: (The D and I commands)
6:*Line numbers
```

To delete a range of lines, beginning with the current line, from the following file:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3:*See what happens when you
4: Delete and Insert
5: (The D and I commands)
6: (Use CTRL-C to exit insert mode)
7: Line numbers
```

Enter ,<**line**> **D**

,6 D RETURN

The result is:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: *Line numbers
```

A (Append Lines) Command

Brief

Format: **[<n>]A**

Purpose: Append lines from input file to editing buffer

Details

Use this command for extremely large files that will not fit into memory all at one time. By writing out part of the editing buffer to the output file with the Write command, room is made for lines to be appended with the Append lines command. If A is typed without a parameter, lines are appended to the part of the file currently in memory until available memory is 3/4 full or until there are no more lines to append.

Use the W command to write out lines to the output file. If the parameter <n> lines are appended to that part of the file that currently is in memory. If <n> is not given, then as much of the input file as possible is read into the editing buffer until the editing buffer is three quarters full.

L (List Text) Command

Brief

Format: <**line**>,<**line**> **L**

Purpose: The L command lists the specified range of lines, including the two lines specified.

Details

If the first line is omitted (the comma must still appear to indicate that the first line was omitted) EDLIN displays from eleven lines before the current line through the given line.

If the second line is omitted, 23 lines are listed, starting with the given line.

If the current line is one of the lines listed, it contains an asterisk between the line number and the first character.

If no parameters are given and only L is entered, 23 lines are displayed: eleven lines before the current line, the current line and eleven lines after the current line.

Application

Assume the following file exists and is ready to edit.

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: Delete and Insert
5: (The D and I commands)
.
.
.
15:*The current line contains an asterisk.
.
.
.
26: (Use CTRL-C to exit insert mode)
27: Line numbers
```

To list a range of lines without reference to the current line, enter <**line**>, <**line**> **L**

2,5 L RETURN

The result is:

```
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: Delete and Insert
5: (The D and I commands)
```

To list a range of lines beginning with the current line, enter: ,<**line**> **L**

,26 L RETURN

The result is:

```
4: Delete and Insert
.
.
.
15:*The current line contains an asterisk.
.
.
.
26: (Use CTRL-C to exit insert mode)
```

To list a range of 23 lines beginning with around a specified line, enter <**line**>, **L**

13, L RETURN

The result is:

```
13: The specified line is listed first in the range.
14: The current line remains unchanged by the L command.
15:*The current line contains an asterisk.
.
.
.
35: CTRL-C exits interline insert command mode.
```

To list a range of 23 lines centered around the current line, enter only **L.**

The result is:

```
L RETURN

 4: Delete and Insert
 5: (The D and I commands)
 .
 .
 .
13: The current line is listed in the middle of the range.
14: The current line remains unchanged by the L command.
15:*The current line contains an asterisk.
 .
 .
 .
26: CTRL-C exits interline insert command mode.
```

I (Insert Text) Command

Brief

Format: <**line**> **I**

Purpose: The I command inserts a line (or lines) of text immediately before the specified <line>.

Details

If you are creating a new file, the I command is given before text is inserted. In this case, the insert begins with line number one.

EDLIN remains in this insert mode until a <**CTRL-Z**> <**RETURN**>, or a <**CTRL-C**> is entered. Successive line numbers appear automatically each time the **RETURN** key is pressed. When the insert is finished and insert mode is exited, the line, which now immediately follows the inserted lines, becomes the current line. All line numbers following the inserted section are incremented by the number of lines inserted.

If the line is not specified, the default is the current line number (the lines are inserted immediately before the current line).

If the line is an integer larger than the last line number, or if # is specified as <line>, the inserted lines are appended to the end of the file. In this case, the last line inserted becomes the current line. (This is the same as when the file is created.)

Application

Assume the following file exists and is ready to edit:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: Delete and Insert
5: (The D and I commands)
6: (Use CTRL-C to exit insert mode)
7: Line numbers
```

To insert text before a specific line (not the current line), enter <**line**> **I**

4 I RETURN

The result is:

```
4: *_
```

Now, enter the new text:

```
4 fool around with
5 those very useful commands that
6 <CTRL-Z>  RETURN
```

If the file is listed now, the result is:

L RETURN

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: fool around with
5: those very useful commands that
6:*Delete and Insert
7: (The D and I commands)
8: (Use CTRL-C to exit insert mode)
9: Line numbers
```

To insert lines immediately before the current line, enter:

I RETURN

The result is:

```
6:*_
```

Now, insert the new text.

```
6:* perform the two major editing functions,
7:* <CTRL-C>
```

List the file to see the result:

L RETURN

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: fool around with
5: those very useful commands that
6: perform the two major editing functions,
7:*Delete and Insert
8: (The D and I commands)
9: (Use CTRL-C to exit insert mode)
10: Line numbers
```

To append new lines to the end of the file, enter either:

11 I RETURN
or
#I RETURN

The result is:

```
11:*_
```

Now, enter the new lines.

```
11: The insert command can place new lines
12: anywhere in the file; there's no space problems.
13: because the line numbers are dynamic;
14: They'll slide all the way to 65533.
15: <CTRL-Z>  RETURN
```

These lines will appear at the end of all previous lines in the file. Enter the list command **L**; the result is:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: fool around with
5: those very useful commands that
6: perform the two major editing functions,
7: Delete and Insert
8: (The D and I commands)
9: (Use CTRL-C to exit insert mode)
10: Line numbers
11: The insert command can place new lines
12: anywhere in the file; there's no space problems.
13: because the line numbers are dynamic;
14: They'll slide all the way to 65533.
```

S (Search Text) Command

Brief

Format: <**line**>,<**line**> **[?] S**<**string**>

Purpose: The S command searches the specified range of lines for the specified string.

Details

The search string that you input terminates with a RETURN. The first line that matches the string is displayed and becomes the current line. The Search command terminates when a match is found.

If no line contains a match for the string, the message `Not found` is displayed. The Search command also terminates when no match is found.

If the optional parameter ? is included in the command, EDLIN displays the first line with a matching string then prompts you with the message `O.K.?`.

If you press either the **Y** or the **RETURN** key, the line becomes the current line and the search terminates.

If you press any other key, the search continues until another match is found, or until all lines are searched (then the `Not found` message is displayed).

If the first line is omitted (,<line> S<string>), the first line defaults to the line after the current line.

If the second line is omitted (<line> S<string> or <line>, S<string>), the second line defaults to # (same as <line>,# S<string>).

If the string is omitted, no search is made and the command terminates immediately.

Application

Assume the following file exists and is ready for editing.

```
1:* This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: fool around with
5: those very useful commands that
6: perform the two major editing functions,
7: Delete and Insert
8: (The D and I commands)
9: (Use CTRL-C to exit insert mode)
10: Line numbers
11: The insert command can place new lines
12: anywhere in the file; there's no space problems.
13: because the line numbers are dynamic;
14: They'll slide all the way to 65533.
```

To search for the next occurrence of a string, enter **<line>,<line> S<string>:**

2,12 Sand

The result is:

```
5: those very useful commands that
*_
```

If you really wanted the "and" in line 7, the <F3> key causes the S command to display again. Pressing the RETURN key causes the S command to execute again. The result of this sequence is:

2,12 Sand RETURN

```
5 those very useful commands that
*6 <F3>,12 Sand RETURN
7 Delete and Insert
*
```

To Search through several occurrences of a string until the correct string is found, enter <**line**>,<**line**> **? S**<**string**>**:**

, ? Sand RETURN

The result is:

```
5: those very useful commands that
O.K.?
```

If you press any key except **Y** or a **RETURN**, the search continues:

```
O.K.?  N
7: Delete and Insert
O.K.?
```

If you press **Y** or a **RETURN**, the search terminates.

```
O.K.?  Y
*_
```

R (Replace Text) Command

Brief

Format: <**line**>,<**line**> **[?] R**<**string**><**CTRL-Z**><**string**>

Purpose: The R command replaces all occurrences of the first <string> in the specified range with the second <string>.

Details

As each occurrence of the first string is found, it is replaced by the second string. The line is displayed with the second string. Only the lines in which strings are replaced are displayed.

If the first line is omitted (,<line>[?]R<string>), the first line defaults to the line after the current line.

If the second line is omitted (<line>[?]R<string>), the second line defaults to #.

If a line contains two or more matches with the first string, the line displays once for each occurrence of the first string. Each successive display of the line shows one more occurrence of the first string replaced with the second string. When all occurrences of the first string in the specified range are replaced by the second string, the Replace terminates automatically, and the asterisk prompt appears again.

The first string is terminated with a **CTRL-Z** if a second string is given as a replacement.

If the second string is omitted, the first string is terminated with either a combination **CTRL-Z RETURN**, or simply **RETURN**. The second string is terminated with **RETURN**.

If the first string is omitted (R<CTRL-Z><string>), the replace is terminated immediately.

If the second string is omitted (R<string><CTRL-Z> RETURN, or R<string>RETURN), the first string is deleted from all lines in the range.

If the optional ? parameter is given, the Replace command stops at each line with a string that matches the first string, displays the line with the second string in place, then displays the prompt `0.K.?`.

If you press **Y** or the **RETURN** key, the second string is left in place of the first string, and the next occurrence of the first string is found and replaced.

Again, the `0.K.?` prompt is displayed. This process continues until the end of the range or until the end of the file. After the last occurrence of the first string is found, EDLIN returns the asterisk prompt.

If you press any key besides **Y** or **RETURN** after the `0.K.?` prompt, the first string is left as it was in the line, and the Replace goes to the next occurrence in the file of the first string.

If the first string occurs more than once in a line, each occurrence of the first string is replaced individually, and the `0.K.?` prompt is displayed after each replacement. In this way, only the desired string is replaced, and you can prevent replacement of embedded strings.

Application

Assume the following file exists and is ready for editing.

```
1:* This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: fool around with
5: those very useful commands that
6: perform the two major editing functions,
7: Delete and Insert
8: (The D and I commands)
9: (Use CTRL-C to exit insert mode)
10: Line numbers
11: The insert command can place new lines
12: anywhere in the file; there's no space problems.
13: because the line numbers are dynamic;
14: They'll slide all the way to 65533.
```

To replace all occurrences of the first string with the second string in a specified range, enter <**line**>,<**line**> **R**<**string**><**CTRL-Z**> <**string**>

For example:

2,12 Rand CTRL-Z or **RETURN.**

The result is:

```
    5: those very useful commors that
    7: Delete or Insert
    8: (The D or I commands)
    8: (The D or I commors)
   11: The insert commor can place new lines
```

To replace only certain occurrences of the first string with the second string, enter <**line**>, <**line**>**? R**<**string**><**CTRL-Z**><**string**>**RETURN**

For example:

2 ? Rand CTRL-Z or **RETURN**

The result is:

```
    5: those very useful commors that
O.K.?N
    7: Delete or Insert
O.K.?Y
    8: (The D or I commands)
O.K.?Y
    8: (The D or I commors)
O.K.?N
   11: The insert commor can place new lines
O.K.?N
*
```

Now, enter the List command **L** to see the result of all these changes.

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: See what happens when you
4: fool around with
5: those very useful commands that
6: perform the two major editing functions,
7: Delete or Insert
8: (The D or I commands)
9: (Use CTRL-C to exit insert mode)
10: Line numbers
11: The insert command can place new lines
12: anywhere in the file; there's no space problems.
13: because the line numbers are dynamic;
14: They'll slide all the way to 65533.
```

E (End Edit) Command

Brief

Format: **E**

Purpose: The E command saves the edited file on disk.

Details

When the edited file is saved with the E command it renames the original file as filename.BAK, then exits EDLIN to the Z-DOS operating system.

If the file was created during the editing session, no .BAK file is created.

NOTE: The E command takes no parameters. Therefore, you cannot direct EDLIN on which drive to save the file. The drive is selected when the editing session is invoked.

If the drive is not designated when EDLIN is invoked, the file is saved on the disk in the default drive. (It is still possible to copy the file to a different drive. However, this is done automatically if the drive is designated during invocation.)

You must be sure that the disk contains enough free space for the entire file to be written.

If the disk does not contain enough free space, the write out is aborted and the edited file is lost.

Application

The only possible command is **E RETURN**. When the exit is complete, COMMAND displays the default drive prompt d:

Q (Quit Editing) Command

Brief

Format: **Q**

Purpose: The Q command quits the editing session.

Press **Y** to quit the editing session. Press **N** (or any other key except CTRL-C) if you decide to continue the editing session.

Details

The Q command does not save any editing changes but exits to the Z-DOS operating system. No .BAK file is created.

The Q command takes no parameters. It is simply a fast means of exiting an editing session.

As soon as the Q command is given, EDLIN displays the message:

```
Abort edit (Y/N)?_
```

Press **Y** to quit the editing session. Press **N** (or any other key except <**CTRL-C**>) if you decide to continue the editing session.

Application

Assume the following file exists and is ready to edit:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3: Compare the before and after
4: See what happens when you
5: Delete and Insert
6: Line numbers
```

Now, delete line 3; enter:

3 D RETURN

Now, list the file; enter:

L RETURN

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3:*See what happens when you
4: Delete and Insert
5: Line numbers
```

Now, if you decide not to keep the changes and to quit the editing session; enter:

Q RETURN

The result is:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3:*See what happens when you
4: Delete and Insert
5: Line numbers
```

Q RETURN

```
Abort edit (Y/N)?_
```

Enter **Y**. The result is:

```
1: This is a sample file.
2: Use: to demonstrate dynamic line numbers
3:*See what happens when you
4: Delete and Insert
5: Line numbers
```

Q RETURN

```
Abort edit (Y/N)?Y
A:_
```

W (Write Lines) Command

Brief

Format: **[<n>]W**

Purpose: Write lines from the editing buffer to the output file

Details

The Write Command is used when editing files that are larger than available memory. By executing the Write, lines are written out to the output file and room is made in the input buffer for more lines to be appended from the input file. If W is typed with no <n> parameter, then lines are written until memory is 1/4 full.

If the n parameter is given, then n lines are written out. Note that lines are written out beginning with the *start* of the file; subsequent lines in the editing buffer are renumbered beginning with one. A later Append command will append lines to any remaining lines in the editing buffer.

EDLIN Error-Messages

Errors When Invoking EDLIN

EDLIN error messages occur either when you try to invoke EDLIN or during the actual editing session.

Message: `Cannot edit .BAK file--rename file`

Cause: You attempted to edit a file with the filename extension .BAK.

.BAK files cannot be edited because the extension is reserved for backup copies.

Cure: If you need the .BAK file for editing purposes, you must either RENAME the file with a different extension or copy the .BAK file but give the copy a different filename extension.

Message: `No room in directory for file`

Cause: When you attempted to create a new file, either the file directory was full or you specified an illegal disk drive or an illegal filename.

Cure: Check the EDLIN invocation command line for illegal filename and illegal disk drive entries.

If the command is no longer on the screen and if you have not yet entered a new command, the EDLIN invocation command is recovered by pressing the <F3> key.

If the invocation command line contains no illegal entries, run the CHKDSK program for the specified disk drive.

If the status report shows the disk directory full, remove the disk and insert and format a new disk.

If the CHKDSK status report shows the disk directory is not full, check the EDLIN invocation command for illegal filename or illegal disk drive designation.

EDLIN

EDLIN Error-Messages

Message: `No end-of-file mark found in file`

Cause: EDLIN found no end-of-file mark (1A hex). After loading the file, EDLIN scans from the last line of the file for the end-of-file mark. When it finds the mark, EDLIN truncates anything after the mark.

If EDLIN finds no end-of-file mark, it assumes that there is nothing to edit.

Cure: DEL the file then recreate it with the command EDLIN filespec.

Errors While Editing

Message: `Entry Error`

Cause: The last command entered contained a syntax error.

Cure: Re-enter the command with the correct syntax.

Message: `Line too long`

Cause: During Replace command mode, the string given as the replacement causes the line to expand beyond the limit of 254 characters. EDLIN aborts the Replace command.

Cure: Divide the long line into two lines, then retry the Replace command.

Message: `Disk full--file write not completed`

Cause: You gave the E (End of edit) command, but the disk did not contain enough free space for the whole file. EDLIN aborts the E command and returns you to the operating system. Some of the file may have been written to the disk.

Cure: Only a portion (at most) of the file is saved. You should probably delete whatever file was saved and restart the editing session. The portion of the file that was not written out is not available after the error. Always make certain that the disk has sufficient space free for the file to be written.

The FILCOM (Compare Files) Command

FILCOM

Brief

Format: **FILCOM <s1>[,<s2>][,<list>][</x>...]**

Command Location: File

Purpose: To compare one file to another to see if they are identical and to create a file that contains a list of differences between the two files.

Switches:

/A	Force source compare
/C	Include comments in compare
/<n>	Required lines to be a match
/S	Include spaces and tabs in match
/B	Force binary compare

Details

The FILCOM (File Compare) Utility compares two files. The differences between the two files are output to a third file that you can inspect at leisure. The files compared are either source files (files containing source statements of a programming language) or binary files (files output by the MACRO-86 assembler, by the LINK Linker Utility, or by a high-level language compiler).

Limitations on Source Compares

FILCOM uses all available memory as buffer space to hold the source files.

If the source files are larger than available memory, FILCOM compares what it was able to load into the buffer space.

The FILCOM (Compare Files) Command

If no matches are found within the portions of the files in the buffer space, FILCOM outputs only the message:

```
FILES ARE DIFFERENT
```

and the compare ends. For binary files larger than available memory, FILCOM compares both files completely, overlaying the portion in memory with the next portion from disk. All differences are output the same as for binary compares of files that fit completely in memory.

FILCOM is invoked one of two ways.

Invoking FILCOM

Method 1:

Enter:

A: **FILCOM RETURN**

FILCOM responds with a banner and the first prompt and then waits:

```
Microsoft File Compare Utility (date)
Copyright (C) 1981 by Microsoft, Inc.
Source1 filename [.ASM]:
```

Method 2:

Enter:

A: **FILCOM <s1>[,<s2>][,<list>][</x>...]**

FILCOM and the filenames must be separated by commas. The slash mark is the only delimiter allowed between a filename and a switch letter. Switch(es) are placed after any of the entries in the invoke command line (but before the comma).

If you want to select the default for Source2 but not for List, enter two consecutive commas between the Source1 and List entries. For example:

A: **FILCOM ALPHA,,GAMMA RETURN**

When you use Method 2, FILCOM responds with a banner but no prompts, and performs the compare. Method 2 permits FILCOM commands in a batch file as well as permitting you to enter all commands on one line at one time. When FILCOM is finished, the operating system prompt reappears.

Commands

Commands to FILCOM consist of responses to three prompts for file specifications, plus optional switches. File specifications are entered one at a time as the prompts appear, or all at once as part of the FILCOM invoke command (Method 2).

File Specifications

All file specifications take the form:

[d:]<filename><.ext>

where **d:** is the letter of a disk drive; <**filename**> is a 1–8 character name of the file; and <**.ext**> is a 1–3 character extension to the filename.

If the drive designation is omitted, FILCOM defaults to the operating system's (current) default drive. See "Defaults and Shortcuts" on Page 9.5, for a list of the default filename extensions used under FILCOM and their effects.

PROMPTS

If you use Method 2 to invoke the command, FILCOM displays no prompts. It simply performs the compare and exits to the Z-DOS command prompt.

If you used Method 1 (or if you invoke Method 2 but with an illegal filename or the name of a nonexistent file for the first source file), FILCOM displays its banner followed by the first prompt:

```
Source1 filename [.ASM]:
```

Enter the name of one of the files you want compared.

The FILCOM (Compare Files) Command

If the filename extension for this file is .ASM, the extension may be omitted from the entry. Otherwise, include the extension. When a legal response is entered, FILCOM responds with the second prompt:

```
Source2 filename [source1.BAK]:
```

Enter the name of the file you want compared to Source1. FILCOM defaults to the backup file for the file named for the first prompt.

If the response to prompt 1 is TEST (meaning TEST.ASM), FILCOM displays as default for Source2 the filename TEST.BAK.

If you want to compare the Source1 file with its backup file, simply press **RETURN**. Otherwise, enter a filename.

Likewise, if the Source2 file has a filename extension of .BAK, the extension may be omitted. Otherwise, enter the extension.

When a legal response to the second prompt is entered, FILCOM responds with the third prompt:

```
List filename [source1.DIF]:
```

Enter the name of the file to receive the list of differences. FILCOM defaults to the name given for Source1 with a default filename extension of .DIF.

If the response to prompt 1 was TEST (meaning TEST.ASM), FILCOM displays as default for List the filename TEST.DIF.

If this default filename is acceptable to you for the List file, simply press the **RETURN** key.

NOTE: Any file already on the disk that is called TEST.DIF is overwritten by the new file.

Otherwise, enter the filename.

Likewise, if the filename extension .DIF is acceptable to you, you may omit the extension even if you specify a filename.

If .DIF is an unacceptable filename extension, enter an extension along with the filename.

When FILCOM is finished comparing the two source files and has output the differences to the List file, the operating system prompt reappears.

DEFAULTS

FILCOM recognizes the following default extensions:

Prompt	Extension	Effect
Source1	.ASM	Default for Source1 filename. May be overridden.
	.OBJ	Causes default to binary compare
	.EXE	
	.COM	
Source2	.BAK	Default for Source2 Filename. May be overridden.
List	.DIF	Default for List Filename. May be overridden.

SHORTCUTS

Two shortcuts for entering commands are supported. Both shortcuts use default responses for any prompts to which a response is not entered.

The FILCOM (Compare Files) Command

RETURN Key

The Source1 filename [.ASM] prompt requires at least a filename response.

The Source2 filename [source1.BAK] and List filename [source1.DIF] prompts show a default entry; the filename entered for Source1 and a default filename extension. To select the default entry, simply press the **RETURN** key.

Example:

```
Source1 filename [.ASM]: TEST  RETURN
Source2 filename [TEST.BAK]:  RETURN
List filename [TEST.DIF]:  RETURN
```

These responses cause FILCOM to compare TEST.ASM with TEST.BAK and to output any differences in the file TEST.DIF.

A RETURN by itself is an acceptable entry for either of these prompts, regardless of what you plan to enter for the other. For example, the RETURN is used to select default for Source2, yet allows a non-default entry for List.

Example:

```
Source1 filename [.ASM]: TEST  RETURN
Source2 filename [TEST.BAK]:  RETURN
List filename [TEST.LST]: PAST.PRN  RETURN
```

These responses cause FILCOM to compare TEST.ASM with TEST.BAK and to output any differences in the file PAST.PRN (default for Source2 was selected, but not for List).

The FILCOM (Compare Files) Command

Semicolon (;)

The semicolon character (;) also selects the default responses to the Source2 and List prompts.

If you enter the semicolon following the Source2 Filename prompt, the List Filename prompt will not appear. That is, the semicolon selects the default response for all remaining prompts. Unlike the RETURN key, once you enter a semicolon, comparing begins, and you have no chance to enter another response for that compare.

You may think of the semicolon as a message to FILCOM that you want to use all default responses only. This is especially useful when using Method 2.

Example:

```
A:FILCOM TEST;  RETURN
```

This entry causes FILCOM to display its banner and then perform the compare. FILCOM compares TEST.ASM with TEST.BAK and output the differences in TEST.DIF (the same as the example under RETURN Key above).

Also:

```
A:FILCOM  RETURN

Microsoft File Compare Utility ([dd-mmm-yy])
Copyright (C) 1981 by Microsoft, Inc.
Source1 filename [.ASM]:  TEST;
```

This set of commands and responses produces the same result as the two previous examples. Note that you have no chance to enter alternatives to the defaults for Source2 or List.

The FILCOM (Compare Files) Command

SWITCHES

FILCOM supports five switches. Switches are single letters appended to the (method 2) command line or to any of the prompt responses to control the file compare.

A switch is always preceded by a slash. FILCOM switches are one of two types: source compare or binary compare.

Source Compare Switches

/A Force a source compare of files with filename extensions .OBJ, .EXE, and .COM. FILCOM defaults to binary compare on files with these filename extensions. Files with any other filename extensions default to source compare. Therefore, the /A switch is not required for files that do not have one of these three filename extensions.

/C Include comments in compare. A comment in one of the files is considered to start with a semicolon (;) and end with an end-of-line character. Default excludes comments from source compare. This means that comments in the two files, even if the same, are ignored when FILCOM searches for consecutive lines that match (see the /n switch below for an explanation of "match"). The /C switch directs FILCOM to treat equal comments lines as matches.

/n Where n is a number from 1 through 9. Default is 3. N specifies how many consecutive lines in the two files must match before FILCOM considers that the two files are a match. (Refer to examples 1 and 2 below for a demonstration of the effects of the /n switch).

When FILCOM finds n lines that match, it outputs the lines that are different since the last n lines that matched, plus the first line of the current n lines that match. The first match line should help locate where differences occurred.

/S Include spaces and tabs in compare. Default excludes spaces and tabs from source compare. This means that spaces and tabs in the two files, even if the same, are ignored and are not used to find matches. The /S switch directs FILCOM to treat equal spaces and tabs as matches.

Binary Compare Switch

/B Force binary compare of files that default to source compare (files without the filename extensions .OBJ, .EXE, or .COM). Instead of a source compare of lines, FILCOM compares the two files byte by byte. For differences, FILCOM outputs the offset location into the files and the differing bytes in hexadecimal. (Refer to example 3 for a demonstration of this use of the /B switch.)

FILCOM Operation

Application

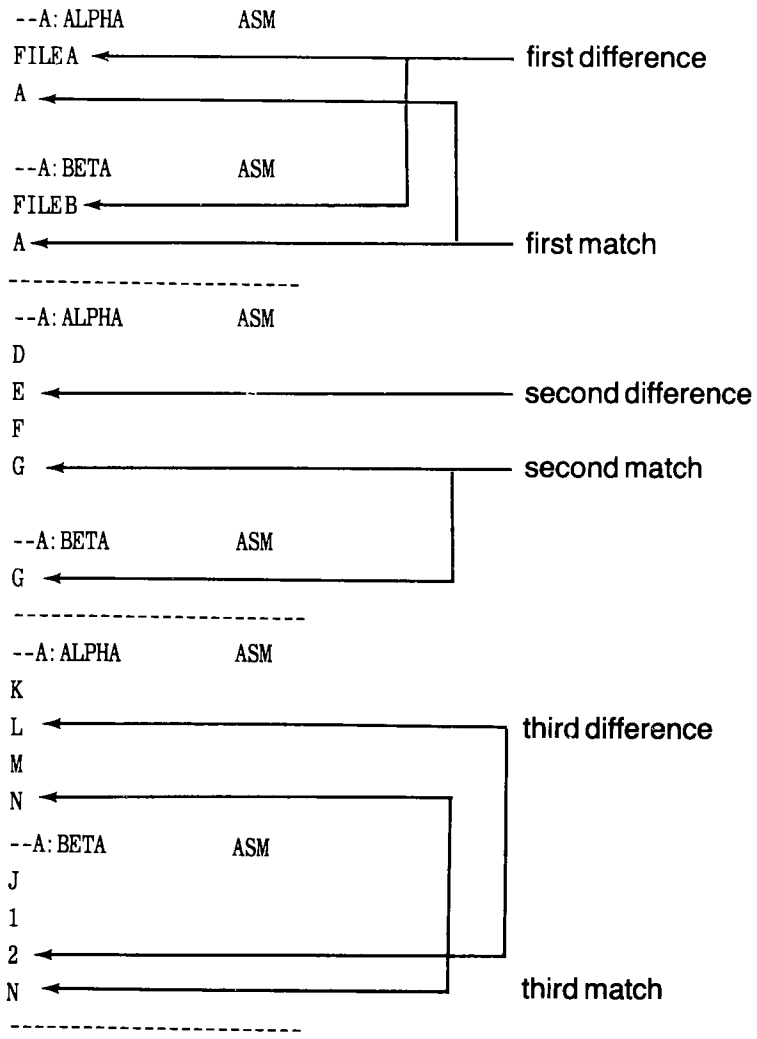
Assume these two ASCII files are on disk:

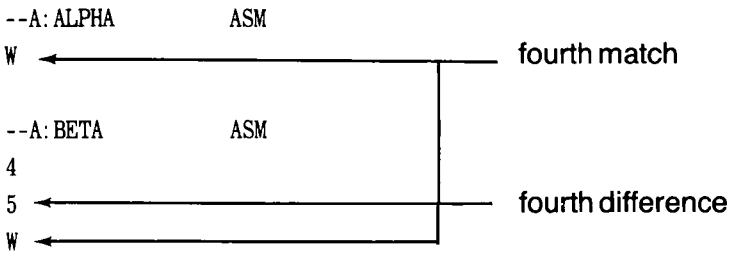
ALPHA.ASM	BETA.ASM
FILEA	FILEB
A	A
B	B
C	C
D	G
E	H
F	I
G	J
H	1
I	2
K	N
L	O
M	P
N	Q
O	R
P	S
Q	T
R	U
S	V
T	4
U	5
V	W
W	X
X	Y
Y	Z
Z	

To compare the two files and output the differences on the terminal screen, enter the following (Method 2) command line:

A: **FILCOM ALPHA,BETA.ASM,CON**

FILCOM is directed to compare ALPHA.ASM with BETA.ASM and output the differences on the terminal screen. All other defaults remain intact (do not use tabs, spaces, or comments for matches and conduct a source compare). The output should appear as follows on the terminal screen:

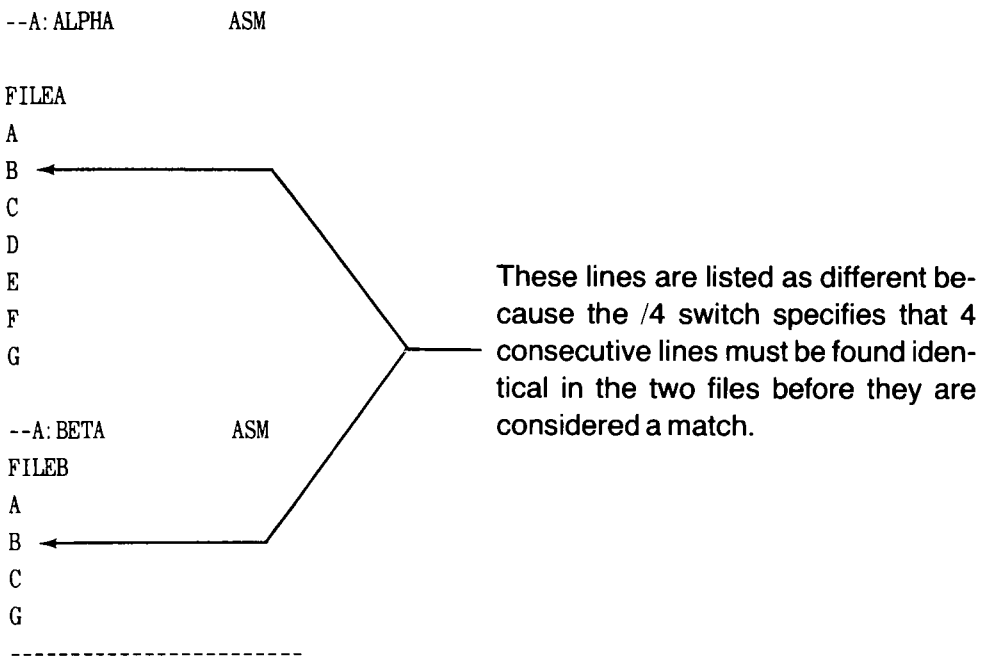




Using the same two source files, output the differences on the line printer. Also, require that four successive lines must be the same to constitute a match. Using Method 2 again, enter:

A: **FILCOM ALPHA,BETA.ASM,PRN/4**

The following output should appear on the line printer:



```
--A:ALPHA          ASM
K
L
M
N
--A:BETA           ASM
J
1
2
N
------------------------
--A:ALPHA          ASM
W
--A:BETA           ASM
4
5
W
```

Using the same two source files again, force a binary compare. Then output the differences on the terminal screen. Using Method 1, enter:

```
A:FILCOM
```

FILCOM responds:

```
Microsoft File Compare Utility ([dd-mmm-yy])
Copyright (C) 1981 by Microsoft, Inc.
Source1 filename [.ASM]:
```

Entries and responses should appear as follows:

```
Source1 filename [.ASM]:    ALPHA/B
Source2 filename [ALPHA.BAK]: BETA.ASM
List filename [ALPHA.DIF]: CON
```

The /B switch at the end of the Source1 line forces binary compare. This switch, and the others, is entered at the end of any of the response entries. The switches may be, but need not be, entered on the same response line.

The screen display should appear as follows:

ADDRESS	ALPHA ASM	BETA ASM
000004	41	42
000010	44	47
000013	45	48
000016	46	49
000019	47	4A
00001C	48	31
00001F	49	32
000022	4A	4E
000025	4B	4F
000028	4C	50
00002B	4D	51
00002E	4E	52
000031	4F	53
000034	50	54
000037	51	55
00003A	52	56
00003D	53	34
000040	54	35
000043	55	57
000046	56	58
000049	57	59
00004C	58	5A
00004F	59	
000050	LARGER	

HEATH | ZENITH data systems

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

4-29-93

595-2827
591-3931