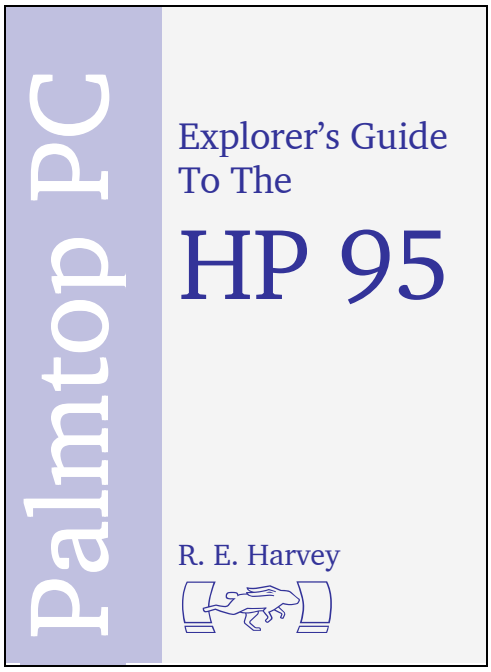


In 1991, while experimenting with one of the very first HP 95 portable computers, this book came together. Today, more than fifteen years later, that HP 95 computer still works great, though it's creaking and slowing down a bit and doesn't have the battery life it once had; I know the feeling.

I would like to invite you to wander through this book, to see what an eleven-ounce computer could do, not all that long ago. Rumor has it that HP later released very nice palmtop computers with 100 and 200 model numbers, even though this book knows nothing about them.

I've simplified some illustrations for this online version, but text and page layout still look like 1991. On the left is something like the original book cover, complete with the old rabbit logo.

One edition of this book was formatted with Microsoft Word for Windows 1.1. It was printed on an HP DeskJet 500 printer, which was a 300 d.p.i. B&W inkjet that scooted through about two pages per minute in best-quality mode, and cost more than $500. Body text in this book is still in Bitstream Charter font, which was designed for low-resolution laser and inkjet printing.

There are 112 pages and around 53,000 words. The peculiar page count begins with this title page. Maybe I was trying to make it sound like a lot of pages, while actually keeping the page count down; maybe it was something else.

The Binary section, beginning on page 12, may look familiar; it was later improved a bit, and used in an Assembly Language book, then improved some more and used on my home page. Page 108, the ASCII Codes table, was pasted-in from another file, so it isn't here. That simple page eventually evolved into something called ASCIIcat, which is a big project, all by itself. Instead of recreating the original table, I'd like to invite you to visit my Internet homepage for ASCIIcat, which is free (probably why it's so popular). Oh, things really have changed in fifteen years! Well, now everyone uses the Internet, and everybody downloads things. If you can't find my home page, I'm sure you can find ASCIIcat by asking google.com.

If you are one of the thousands who purchased this book when it was available in print, thank you, I truly do appreciate your support. This book was self-published, and it retailed at about $19.95. Over several years, most copies were sold through EduCALC mail store. I miss EduCALC, their catalogs, and its founder, Jim Carter. They provided an invaluable service to HP portable users from the 1970s through well into the '90s, when everything changed. Someone has surely written that history, and it's probably available on the Internet.

In addition to a couple of editions of this book, I also did some software packages for the HP 95. The final software took seven or eight very tough months to write, 80-100 hours per week, and then ended up selling fewer than seventy copies at $39.95 retail (I netted about half that amount). Well then, I tried one more HP calculator book, and then I moved on. Since that time I've written programming libraries, lots of code, documentation, and a few books. The latest book is mostly a novel, and it might actually be available in stores that don't sell calculators. I'm still working, writing, walking, enjoying life and little HP classic portables. I hope you are, too.

Happy Programming

R. E. Harvey
October 14, 2006

Implements of Instruction

This book is a companion to a software package, the *HP 95 Utility Pack,* which is likely available from where you purchased this book. The package includes more than 30 small, efficient programs and utilities, including features missing from ROM-based MS–DOS 3.22. All programs are written in assembly language.

A consistent, easy to use interface works with DOS' command line, redirected input, or prompts for data automatically. You can use it to view PCX files, change file attributes and dates, encrypt files, create sounds. It includes several filters and 95.COM, which combines 20 utilities in one program. Also included is a TSR to set the cursor size — and to make sure it stays that size — yet it lets programs turn off the cursor when they need to.

Expect other products for our HP 95. Contact your dealer or the author for information.

Copyright Acknowledgments

Notice

2006 Addendum

R.E. Harvey
P.O. Box 10086
Glendale, AZ 85318 U. S. A.

E-mail: r.e.harvey @ gmail.com
Homepage: http://ourworld.compuserve.com/homepages/r_harvey/

Contents

Introduction

Every now and again something changes our lives. Things like automatic transmissions, microwave ovens, video recorders and global warming. The palmtop PC is another milestone.

At about eleven ounces, the HP 95 Palmtop PC favors a salesman's sample. With a megabyte-worth of built-in programs including Lotus 1–2–3, a half-megabyte of RAM, and the expertise of Hewlett-Packard, Lotus Development and Microsoft behind it, we can't help taking it seriously. That's the impact of the HP 95: instant access to programs and data in a compact, unassuming, well-engineered package.

What This Book Will / Won't Do

The HP 95 Owner's Manual is really quite good at many things, like introducing Lotus 1–2–3. We won't explain all the details or include many examples of using Lotus 1–2–3; that's best left to a 1–2–3 book. The 1–2–3 furnished with the HP 95 is nearly identical to version 2.2, and Lotus 1–2–3 is one of the best known, most completely supported software products ever produced; nearly every book store has a Lotus section. We will, however, show how the HP 95 makes use of some of Lotus' unique capabilities.

The Owner's Manual is mostly a 1–2–3 book, and it misses some important tools completely. We'll spend some time talking about those forgotten features.

Your HP 95 is likely not your first experience with handhelds or personal computers. We won't belabor points like cursors and programs and data, though we will spend quite awhile talking about MS–DOS. Many think of DOS as a program launcher and a way to make back-up copies of files, but it's much more. The power of MS–DOS is not as Byzantine as it may seem.

To make DOS more comfortable on the HP 95 and to illustrate some examples, we've included about ten sample programs written in assembly language.

How to Use This Book

The HP 95 is two computers in one box: a dedicated ROM-based personal information manager, and a multi-purpose PC-compatible microcomputer. We'll talk about these unique personalities separately, keeping in mind where they overlap or collide.

The ROM personality is consistent and easy once you've mastered it. The next chapter, HP 95 *Guided Tour,* is an "Executive summary" of some special keys and the ROM-based software. It's probably all you will need to get started.

If you're a casual user (aren't we all?), you might be tempted to skip *Binary Survival Guide,* the most textbook-like part of this book. Give it a look, though, it's a painless look at binary without the math anxiety.

Near the end is a chapter named *More Than You Probably Want To Know.* It's aptly titled. Once you've settled-in and would like to know where your HP 95 came from and how it was made, take a look at this chapter.

This book is mostly sequential. Each chapter is a unique subject, building on understandings from earlier chapters, but at the same time separate and usually unrelated. Think of each chapter as an essay.

At the end of most sections, a topic called *Notes about ¼* reviews salient issues and maybe a few caveats, as well as offering additional information not mentioned in the text. If after reviewing these points an item or two seems murky, feel free to pop back to the appropriate headline for another round.

Sample programs include listings that you can type directly in your HP 95 using the Memo program. The DEBUG section explains how to turn the text files into programs.

Don't limit yourself to typing the keystrokes in the examples (you already know what those keystrokes do). Try other combinations, for when we experiment, the little light goes on faster.

Symbols and Conventions

We'll show keystroke combinations joined with the plus character, for instance **CTRL+S**, in bold, small-capital letter type. This means you press and hold down **CTRL**, then press the **S** key, then release both keys. The HP 95 Owner's Manual uses a minus sign instead of a plus. Some manuals even represent the same keystroke as [Ctrl][S]. We've chosen the plus because it is a standard followed by much of the industry, including Microsoft. We often see small capital letters, but they can be difficult to recognize, which is why they are in bold type.

Cursor keys and the backspace key are shown as arrow characters. "←" is left cursor, "↑" is up, and so forth. The backspace key is represented as the "⇐" character.

When we introduce computer terms or want to emphasize a point we'll usually show text in *italic type* the first time, then in conventional, Roman type. It's a subtle italic style, hopefully not too distracting.

Symbols in the left margin offer suggestions (☝), warnings (✋) or keystroke shortcuts. The circular arrow icon points to other places to look for related information. Occasional icons are just for novelty value.

The "*" symbol is correctly called an *asterisk.* In conversations about programming and MS–DOS, we usually call the character a *star.* So that's what we'll call it here, along with a half dozen other mispronounced words and symbols. Computer science traditionally has a hit-or-miss relationship with grammar and proper English. There's no sense bucking traditions.

Examples in this book are in upper case, fixed-pitch type. MS–DOS commands are not case-sensitive; you can type commands in upper- or lower-case.

We'll use the terms DOS and MS–DOS interchangeably. Both refer to the Microsoft MS–DOS operating system. Some program version numbers are shown as 2.*x,* where the 2 (or whatever is appropriate) stands for the major revision and *x* stands for any minor revision number.

MS–DOS files usually have distinct file name *extensions.* These are the one- to three-character last names of the files, following a dot (another name for *period).* For instance, programs usually end in ".COM" or ".EXE;" something like COMMAND.COM. When we show a file extension without a file name, we'll show just the word, without the dot. For instance, we'll talk about specific files like DEBUG.EXE, and we'll also talk about the EXE file type.

Don't be deceived by the size of this book. There is a lot of information on each page (more than twice as much as other books of this type), presented in a simple, readable format.

Terminology: LCD, CPU and all

Computers are reflections of personal beliefs and styles of their creators and users. As such, new words are coined and meanings of existing words are reinterpreted, having little to do with the original. We often end up with words like *mnemonic* or *octal,* or obscure references to "Star Trek." There were *plugs* and *distributors* before there were automobiles, just as there were *bites* and *operators* before there were computers.

Occasionally there's a shortage of words. A case in point is the word *display.* We display numbers in a display format on the computer display. When you stop to think about it, it doesn't really matter what you call things, just so we consistently call them the same things. The Glossary on page 103 defines some of these terms.

They're just words.

HP 95 Guided Tour

Computers are about information, not computing. If a computer is to be truly useful — and personal — it has to be easy to get at that information. If you have used a desk bound computer, mastering the HP 95 will be a breeze: it's consistent, simple and powerful. It'll be even easier if you have used a Lotus program like Metro or the 1–2–3 spreadsheet.

The built-in applications were designed by Lotus to work consistently; once you've used Memo (the HP 95's text editor), you're primed and ready for the next application. When you add new programs designed for the HP 95, they will share the feel, consistency and power.

The best way to learn to use the HP 95 is to use the HP 95. This section, while not quite the "One Minute HP 95," is brief; it doesn't try to explain everything there is to know about the computer. Most everything the 95 does well is hiding in menus, and once you've found how to get to the menus, you're home free. Give it a try.

About Those Blue Keys

The blue application keys are shortcuts, offering one-key access to the built-in applications. Unique pictographs on key faces describe each program.

Each of these *hot* keys starts one of the HP 95 programs, and you can start a second or third or forth program without quitting the first. When a program is running but in the background, pressing the hot key again returns you to exactly where you left off.

Don't confuse these special keys with the function keys on other PCs. Function keys are general purpose, they do different things depending on which program is active. The blue keys always do the same thing, regardless of where you are.

The Filer

The Filer program is the backbone of the HP 95 file system. All of the other programs create and use files, but only Filer lets us copy, delete and rename files. It's also the portal to access other MS–DOS applications, but it's not the place to format RAM cards — that's done in the Setup program.

Data Communications

With the Comm program, you can connect your 95 to another computer or to a modem, and share data and files with X–Modem or Kermit protocol. Usually, we connect the two devices with a Serial Cable, part number HP 82222A, or with the cable included in the HP 95 Connectivity Pack.

Appointments, Stopwatch, World Clock

This is not just a calendar. The Appointment Book can set alarms and turn the computer on to remind you of repeating events. Hidden behind the **F9** function key is a second application program: a stopwatch, and within the stopwatch is a world clock.

Phone Book

The Phone Book application is a personal data base. It is *not* a phone dialer.

The Phone Book name list is sorted alphabetically beginning with the first character on each line. Be sure to enter last names first or the AA Milne entry will be at the start of your data base, before BB King, while the listing for ZZ Tops will be at the end.

You can search for information in two ways. Type the first character or two of a name to search the name field — you will want to enter last names first in this field. You can search for any keyword in any field by pressing **F7** (the Find key), then typing the name. Each time you press **F7** again, the computer will continue the search. Neither search method is case sensitive.

The Phone Book remembers names and numbers of up to about 500 of your closest friends. For more complex database tasks, consider using Lotus 1–2–3. If you use your HP 95 mainly for data collection or other dedicated data base functions, consider a program designed just the job.

When you add new entries to a data base, be sure to press **F8** (Insert) when you're done, instead of **ESC**. Edits to an existing entry are lost unless you immediately insert a new blank entry.

Memo Editor

This is the 95's simple, easy to use word processor that works with plain text files — no special formatting codes. It has insert and overtype modes, and automatically wraps text at the end of lines. Memo is wonderfully straightforward to use; it's a great place to start experimenting with your HP 95.

You can change the tab stop and word wrap settings in the menu. This changes how the program displays and edits files, but it does not change how it stores files on the disk drive. Memo only adds a true end of line marker when you press **ENTER** to start a new line. If you import the text files to your desk bound computer, be sure the text editor on your PC can handle the long lines. Be sure also to save files before exiting!

Lotus 1–2–3 Spreadsheet

This is the fully powered Lotus spreadsheet, version 2.2.

Financial Calculator

It's not just a financial calculator, but a versatile, full-featured program with equation solving and graphing. It even has an RPN mode, though it is not "keystroke programmable" like the classic HP-41.

The calculator automatically saves the contents of the registers each time you exit the program — the next time you run the calculator it restores the registers to their previous state.

The Setup Program

Setup is the place where you set the clock, speaker volume, RAM disk size and format RAM cards. The Setup program is the only built-in application without a separate menu, and the only application that requires the **SHIFT** key to start.

Be sure to pop-into Setup every few days to keep an eye on the battery level. When it dips to half or so, it's probably time to start packing spare batteries. We'll talk about batteries again later, and we'll introduce a new program, called BATTERY.COM, that displays the battery level a bit more accurately than Setup.

Some Other Important Keys

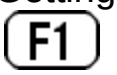
The built-in applications consistently do similar things with similar keys: **DEL** always deletes text, and **MENU** always opens the menu. Many MS–DOS applications also recognize these (or similar) keystrokes and share the same function keys, but alas there is no standardization in the MS–DOS world.

The HP 95 is patient, but after about five minutes with nothing to do it shuts-down to conserve battery power. The **ON** key does the same thing — you don't have to reboot each time you want to type a few lines of text or look-up a phone number — just turn it on and do your business then shut it off when you're done, and when you press **ON** it will come back to life where you left off.

The **ON** key is not part of the applications programs, but part of the operating system itself. You can put the computer to sleep while any program is running, even an MS–DOS application, by pressing **ON**.

If you have tried turning the computer off in a panic when something got out of control, you know that it will still be out of control when you turn the computer back on. We'll talk about how and why to reboot the computer later.

Getting Help



Wherever you are, there's at least a screen-full of information available about what to do next. The **F1** key is the de facto standard recognized by nearly all larger MS–DOS programs recognize for help (well, except for the WordPerfect word processor, which uses **F4** — there's always a holdout in every crowd).

The flashing cursor marks the current highlighted (inverse video) help topic, and cursor keys move you to other highlighted topics. Press **ENTER** to jump to the new topic, or **ESC** to return to your program.

- In Comm, press **CTRL**+**F1** for help. In Hearts&Bones, press the **F4** function key (see page 37 for information about this program). There is no on-line help for TigerFox or the MS–DOS command line editor.

Getting in Deeper

Many program functions are accessible through function keys, like **F1** for help. But the power of the HP 95 is in the **MENU** key. The **MENU** key opens a classic Lotus-style menu. The menu pointer, often called the menu bar, highlights the selected item. The apparent simplicity of the HP 95 is mostly due to hiding complexity (and features) in menus. Here's the Filer main menu:

```
File-Sort  Directory  Remote-Set
Options  System  Print  Quit
```

One menu often leads to another, and maybe even another. You can press **ESC** to back-up one level, or **CTRL**+⇐ (that's the **CTRL**+**BREAK** key) to quit the menu and return to your program.

- Most applications have a File sub-menu to load, name, and save files to disk. Follow the prompts carefully
- Lotus 1–2–3, HP Calc and the Appt program are very complex, full-featured programs. They hide much of their complexity behind menus.
- Until you press **ENTER** to accept the menu selection, no changes you have made are permanent.
- You can press **ESC** to back-up one level in heavily nested menus. If you change your mind and want all of the way out of the menu, press **CTRL**+**BREAK** (Press and hold **CTRL** while you press the ⇐ key).

Getting Out

The **ESC** key, short for *Escape,* most often cancels operations. It doesn't cancel programs.

Say you've popped-into the Phone book and would like to get out gracefully without changing anything. If you have changed a file, the editor (and most other programs) will ask if you would like to save the new version. Selecting Yes opens a series of confirmation menus, much like Lotus 1–2–3. Follow the prompts carefully, it's easy to press **Y** at the wrong time and accidentally quit the program without saving your file!

Quitting programs is more involved than it sounds. Here is one place where it gets more confusing — not complex, maybe annoying. Depending on which application is running:

- Lotus 1–2–3 asks for a confirmation before exiting.
- Setup doesn't use a menu bar, so you just press **Q**.
- TigerFox and Hearts&Bones use the **ESC** key to exit.

Don't press **ESC** too often in the games! The first keystroke will stop the current game, but the second time you press **ESC** will exit the program.

Notes About the HP 95

- The **MENU** key is your gateway to the power of the HP 95 built-in applications.
- **ESC** cancels operations or backs-out of menus. **CTRL**+**BREAK** is frequently used as a "Super" **ESC** key, to cancel or back out completely.
- You can turn the computer off and on in the middle of things without problems. The next time you turn the computer back on, it will continue from exactly where you stopped.

Files And Directories

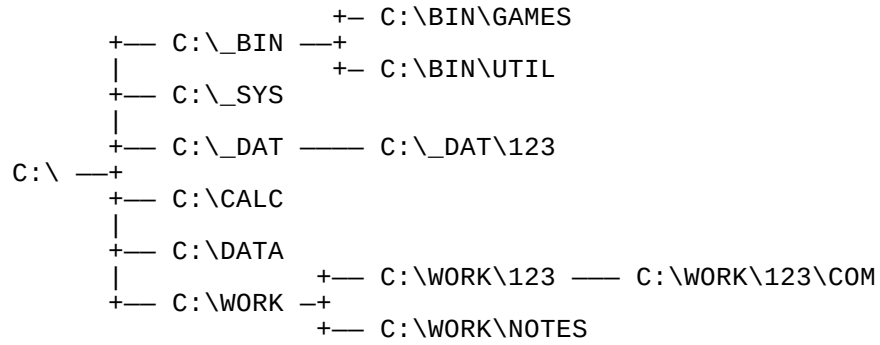
Our HP 95 is an MS–DOS compatible computer. This isn't a buzzword or sales hype, it's a way of life that defines how programs interact with one another and with disks and data.

The HP 95 RAM disk is the storage medium for data we create with the built-in applications and for new programs we add to the computer. Each disk drive is identified by a letter followed by a colon; the built-in RAM disk is C:, the plug-in card is A:. We tell programs which disk drive to use by specifying the drive letter.

The HP 95 built-in applications aren't stored on a disk drive; they are a permanent part of the computer. Other programs, such as CHKDSK.EXE (which is on the C: RAM disk), are stored in files. We usually run these programs from Filer by moving the selector bar to the program name then pressing **F4**. When the program ends, Filer starts again and returns us to the built-in programs.

Information used by both MS–DOS and the built-in applications is stored in *files.* When we create data with programs, they store the data in files on the disk. Think of a disk drive as analogous to a filing cabinet, and each file as a file in that cabinet.

Instead of throwing all the files into piles in the filing cabinet we split the cabinet into compartments with shelves or drawers. A disk directory is like a drawer in the filing cabinet. Here's a typical directory tree:



We call it a directory tree because directories and sub directories grow from the root directory. (Lean sideways and you'll recognize the tree metaphor a little more clearly.) There is always at least one directory in the tree: the root, which is C:\ in the example.

Each sub directory sprouts from the next directory up. We separate directory names with the backslash character ("\"). In the example, \NOTES\ is a sub directory of \WORK\, which itself is a sub directory from the root.

When we navigate through the directory tree with Filer, we can move freely between \NOTES\ and \WORK\ because they are on the same branch, but to move from C:\WORK\NOTES\ to C:_DAT\ we will first have to return to the root directory. We can only move between directories from a common root.

The HP 95 automatically created the C:_DAT\ directory when you turned the computer on the first time (and the hidden C:_SYS\ directory, but we'll talk about that later). The directory tree above is overkill for the HP 95, but you'll surely want to add one or two new directories.

You can create more directories with Filer, and that's probably a good idea. Instead of bunching-up all your files in the C:_DAT\ directory, you might, as we did above, create a directory for utility programs (called C:\BIN\, a commonly used name that's short for *binary* — program files are in binary format), another for work files, and perhaps others for types of spreadsheets. When you have separate folders for each group of files, it's easier to find the file you're looking for.

Binary Survival Guide

Here's a little background on memory, the hexadecimal numbering system, and binary — the root of it all. If you don't really care about this topic, skip to the next section. Remember where it is, you may want to come back later.

When you're done here, you may be looking for a hexadecimal calculator, like the classic HP–16C. Page 89 in the DEBUG section talks about the HP 95's built-in hexadecimal calculator.

Binary 101

Computers are by nature (and design) binary beasts, with chips containing millions of circuits, and each circuit representing one of two states: on or off. Here's a table showing the entire binary numbering system and the complete range:

0	1

No, you didn't miss something, that's it! Binary digits, like light switches and the flag on your mail box, can be either on or off. The word *bit* is a contraction of *binary digit*.

For more complex signals to the letter carrier, we could add more flags to the mail box: one for each class of message (now, what did that chartreuse flag mean?). The postal service usually only recognizes one flag, so I guess we can hang a note on the mail box.

Individual bits are about as useful in day to day life as subatomic particles. Tying notes on bits won't help; the computer can't read, anyway. We need to lump bits together to make useful quantities — and extend the maximum value beyond one. Doubling the number of bits doubles the range they can represent. The least significant digit is on the right — just like decimal numbers. In the computer world, two-bits is not a quarter dollar.

Two-bit Binary Numbers

0	0	=0. Both bits are false.
0	1	=1.
1	0	=2.
1	1	=3. Both bits are true.

Now we have a range from 0 through 3, still using only two binary digits. The only problem with this scheme is that even if the number we want is zero, we still need both bits. This is always true — memory is always reserved for the greatest possible magnitude allowed by the base — though sometimes we type the number "1" even though it's really "01".

But two bits still aren't enough for serious work, let's double it again.

Hexadecimal Numbers

Combining bits into groups of four increases the range from 0 through 15. As we move from right to left, each bit we add doubles the range. Representing it as powers of two, here are the possible values:

2^3	2^2	2^1	2^0
8	4	2	1

As we add values of set bits from each position, we find that 0101 binary is equivalent to 5 (2^2+2^0) decimal. When we get past 0111 things get out of hand: you can't represent these values as a single decimal digit.

We've progressed (out of need, not necessarily choice) to the base 16 numbering system, called *Hexadecimal,* where numbers don't stop at 9, but continue on through F, which is just big enough to represent four binary digits. Talking about numbers in binary is as fluid as talking about them in Roman numerals. Which is why we use hexadecimal: it's a compact way to express binary values.

Binary to Hexadecimal Conversion

Hex	Binary	Hex	Binary
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Hexadecimal is usually abbreviated as *Hex.* The fact that hex means six must have gone unnoticed in computer circles.

Larger Hexadecimal Numbers

By combining hex digits, we can create larger binary numbers. A four-bit quantity — a single hex digit — is called a *Nibble,* or occasionally *Nybble* (abbreviated as *Nib* or *Nyb).* The most common units are *Byte* and *Word:*

Common Data Sizes

Name	Size
Nibble	4-bits
Byte	8-bits
Word	16-bits
Double Word	32-bits

These data sizes usually reflect the size of microprocessor data registers. Internally, the HP 95 works easiest with 8-bit and 16-bit quantities — the 8-bit byte and the 16-bit word — because the 95 has a 16-bit microprocessor.

Note

While 16-bits is a common word size, it is by no means the only one. Hewlett–Packard, for instance, uses a 20-bit word size in their HP–48SX handheld, and other computers often use 32-bit words. The word size often reflects microprocessor memory address calculation units.

Bytes are tidy units of eight bits, the smallest useful unit for storage, representing values from 0 through 255.

7	6	5	4	6	2	1	0
0	0	0	0	0	0	0	0

The letter "A" takes one byte of memory. In memory, the letter "A" could be interpreted as the decimal number 65, the hexadecimal number 41, the binary number 01000001, or even part of a larger value. It all depends on how you look at it.

ASCII Codes

The ASCII (American Standard Code for Information Interchange) coding system uses an 8-bit number to represent letters, numbers, punctuation and special symbols. Each character in this sentence, including spaces, requires one byte for storage on the disk, and one byte will have to be allocated as each character is loaded into memory.

The ASCII standard, developed by the American National Standards Institute (ANSI), assures that the character represented by ASCII 65 is "A" regardless of the machine. Using ASCII codes for communications, the weather map computers on television, all-night teller machines, and even otherwise combative computer systems, could have reasonable conversations. Though it's hard to imagine what they'd have in common to talk about.

The ASCII standard applies only to codes 0 through 127 (using only the lower seven bits); the eighth bit is reserved for *parity checking*. That parity bit is as a checksum, tagged-onto the data to assure that when the byte gets to its destination, it is still intact. In the last twenty or so years, data transmission has become much more reliable, so the parity bit is seldom needed anymore. This left it open for the engineers at each computer company to decide to use the extra bit.

ASCII codes above 127, called high-order characters, are today defined by a few more standards. By far the most common is the character set used by most PC-compatibles. ANSI has stepped-in and assigned another, and the HP 95 lies somewhere in between. Computers are mostly English-speaking devices. Internationalization of computers means the 8-bit characters aren't enough (there are *lots* of Kanji characters), so expect another standard in a few years using 16- or even 32-bit characters.

See also Page 107 offers a look at the HP 95's character set.

It's not until we get to 16-bits that binary numbers are big enough to use as, er, numbers. Numbers of bytes in a file, memory addresses, distances, bank check numbers, and so forth begin to fit in binary words, with values up to 65535 decimal.

15	14	13	12	11	10	9	8	7	6	5	4	6	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16-bit words are often used to represent *signed* values. The leftmost (most significant) bit becomes the sign bit (true for negative, false for positive), leaving 15-bits for the mantissa. Thus, signed words can have values of -32768 (8000) through 32767 (7FFF). (You might think the smallest number would be -32767, but the HP 95 uses two's complement negation, where the number is NOTted then incremented by one — so FFFF in hex is -1 decimal).

Hexadecimal numbers frequently represent quantities like file sizes and number of bytes left on a disk. You cannot have negative file sizes, so there is frequently no sign bit.

These powers of two tables compare 8- and 16-bit data:

8-Bit Bytes

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

16-Bit Words

2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

For even larger values, we can combine bits into 32-, 64- and even 80-bit units. For text, we use long strings of ASCII codes in 8-bit units.

Beyond Bytes

Bytes and kilobytes are the ounces and pounds of the computer world — consistent, but not very intuitive. Computer companies sell memory chips by the *Kilobyte* (abbreviated KB or just K), not the pound, so kilobytes are the standard bartering unit. Instead of making the storage unit larger, we just use more of them.

Kilobytes are expressions of powers of two, so instead of 1000, a K is 1024 (that's 2^10 bytes). The next size up is *Megabyte* (MB). This is not a villain from an old shark movie, but a K's-worth of K's, or 1024*1024 bytes.

One common misconception is that a kilobyte is 1,000 bytes, and 512 kilobytes is 512,000 bytes. Since 1K is 1024 bytes, 512K is really 512*1024 bytes, or 524,288 bytes. It figures that a megabyte isn't one million bytes, but 1024*1024 bytes, or 1,048,576 bytes.

Name	Formula	Bytes
Kilobyte	1*1024	1,024
Megabyte	1*1024*1024	1,048,576

Then we have Gigabytes and Terabytes, but we'll leave those units for another story.

Memory Addresses

Members of the Intel 8086 family are *Segmented little endian* microprocessors. The HP 95 is no exception. This means that the computer stores multiple-byte values with the least significant bytes (the little ends) lower in memory, and it calculates memory addresses with segments and offsets.

The Apple Macintosh family uses Motorola microprocessors, which are *flat big endians.* This may explain some of the communications problems.

Address calculations are often difficult do grasp, even for experienced programmers. The segmented 8086 family is probably the reason most commercial applications (including the HP 95 built-in applications) are written in high level languages like C, instead of assembly language. C automatically takes care of segmentation hassles. When an otherwise civilized bit of code suddenly goes bonkers, it's usually the result of incorrect memory pointers. Let's have a go at sorting out pointers, anyway.

We express addresses as segment:offset, in the format 0000:0000. This address represents the first address (offset zero) of segment zero. Looking at these values in memory, the low byte of the offset will physically come first, followed by the high byte of the offset, followed again by the low then high bytes of the segment.

The offsets of addresses increase first, so 0000:0001 immediately follows 0000:0000.

Segments and offsets can have values of 0000 through FFFF, so each segment can be up to 64K bytes. The computer calculates addresses as twenty-bit (five hex digits) values, so the maximum address space is 16*64K, or 1024K (that's 1,048,576 bytes).

Shift the 16-bit segment address left by four bits (one hex digit); this is the same as multiplying by 16 decimal (10 hex), since every time you shift one bit to the left multiplies by two, and each shift to the right divides by two. The lower four bits of the twenty bit result are zeroes.

	1	2	3	4	16-bit segment.

Shift the segment register left 4-bits (multiply by 16):

1	2	3	4	0	20-bit segment.

Add the 16-bit offset:

1	2	3	4	0	20-bit segment.
	0	5	6	7	16-bit offset.
					+ Add segment and offset.
1	2	8	A	7	20-bit physical address.

Any combination of segment and offset can be combined, though since segments are multiplied by 16, they always begin on 16-byte boundaries (the lower nibble is always zero), called *pages.* Available memory is often rounded-down to pages — memory available in Lotus 1–2–3 is shown as a multiple of 16.

Let's look at a real HP 95 memory location. The scan code for the last key pressed is stored at address 0040:00F8; here's how to calculate the 20-bit address:

	0	0	4	0	Start with a 16-bit segment.
0	0	4	0	0	Shift into a 20-bit segment (multiply by 16).
	0	0	F	8	Add the 16-bit offset.
0	0	4	F	8	Which results in a 20-bit physical address.

The calculated physical address of 004F8 will always point to the same location. We can represent this address in any way that results in 004F8; for instance 0000:4F8 or even 004F:0008. We'll have a demonstration using this memory location on page 89 of the DEBUG section of this book.

Segment registers can hold values up to FFFF, with an offset just as large, much beyond the one megabyte address space, so the 8086 has do be able to deal with these odd numbers. It does this by a method known as segment wrap; when address calculations grow too large, the pointers wrap around to zero again. This is an interesting feature, but don't depend on it. Intel processors that can address more than one megabyte (like the Intel 80286) can disable segment wrap to address the next 64K beyond one megabyte limit — using 21-bit calculations and 21 address lines! Better yet, Intel's 80386/486 can use the full 32-bit address as one huge linear segment. Ah, life in flatland, no more segments to worry about.

Types of Memory

All these addresses point to real memory: the bits and bytes that make up the innards of the computer. This data storage takes three forms: *ROM, RAM* and *RAM Disk*.

ROM, an acronym for Read Only Memory, is permanently programmed at the computer factory. It contains the built-in applications, MS–DOS, and everything we need to get the HP 95 started and keep it running.

RAM means Random Access Memory. It's the main user memory — where spreadsheet data is held while we work with Lotus 1–2–3, and Memo keeps the current text file. When we finish editing, the computer saves the data to disk-based files. Most programs copy data to main RAM for editing because it's much faster to manipulate it there than on the disk drive.

The HP 95 handles memory with a twist: it allocates part of RAM as a disk, used just like a floppy or hard disk on your desk bound PC. When we save the 1–2–3 file to RAM disk, the computer copies it to another portion of memory or to a RAM card. Like a disk drive, HP 95 RAM cards retain their data when we turn off the computer.

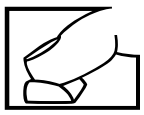
The internal C: RAM disk is part of main memory, designed to look like a disk drive to programs. RAM disks are more practical than physical disk drives on palmtop computers. A disk drive wouldn't run very long on two AA batteries. It's difficult to squeeze an inch-thick disk drive into an inch-thick computer.

There, that wasn't too bad.

Living With The HP 95

A personal computer isn't truly personal if it's inconvenient to use. The palmtop PC redefines the personal computer: powerful, versatile, approachable and handy. There's much to be said for the right tool for the job, but the right tool, in real world, is the tool you have with you.

Getting Started



If you haven't spent very much time with your HP 95 yet, experiment for awhile, try things, whether you think they'll work or not. We'll meet back here later.

You won't damage the computer by pressing keys. It may beep at you, lock-up, or even erase files, but you won't hurt the computer or lose any of the permanent files or built-in applications by trying things. If you aren't confident about what you are doing, make backup copies of important files you've created, then have at it — enjoy.

Even if the computer seems to go daffy or scrambles an important file and the computer refuses to run, there are several ways to recover. We'll talk about damage prevention and solutions in *Before Things Go Wrong,* starting on page 18.

The Big/Small Picture

Jaguar — that's Hewlett-Packard's internal name for our HP 95 — is much more than a palmtop information manager, it's a general purpose IBM PC compatible computer. The HP 95 can do business as a calculator — or just about anything else, for that matter.

As we said at the beginning, the HP 95 has a split-personality (see *How to Use This Book,* beginning on page 5). We have the built-in the applications: a personal information manager controlled by the ROM-based System Manager. The other personality is a traditional command line-driven, IBM–PC-compatible computer.

Most computers hide only the *BIOS* — the lowest-level utilities that connect software with hardware — in ROM. There's a lot more power in the HP 95's megabyte of ROM: the MS–DOS operating system and the built-in applications. Built-in languages include Lotus 1–2–3 functions and macros, as well as MS–DOS' batch language and the primitive assembler in DEBUG.EXE. The distinctions between these components blur in a computer as integrated as the HP 95.

MS–DOS is not the evil twin, but a partner to the built-in applications. MS–DOS handles disk drives and data files for both environments. The long-time standard MS–DOS file structure and Lotus 1–2–3's file format means compatibility. The built-ins replace most of the drudgery of MS–DOS — we can do most common tasks with just the built-in applications, without having to run (or buy) any extra programs.

What Happens After You Press ON?

Press the **ON** key and the computer boots; it's as simple as that.

The term *boot* evolved from *bootstrap-loader,* a method for a computer to "Pull itself up by the bootstraps." In the early days of computing (as in the early days of aviation, though more than a half-century later), starting was an arduous, time-consuming, hit-or-miss affair. Once you got it running, you really didn't want to shut it off.

It's still an arduous task, but at least we have a starter motor: the BIOS (an acronym of Basic Input/Output System). The BIOS is permanently programmed ROM containing essential programs to support the display, keyboard and disk drive — and to self-test and boot the computer.

Every time we power-up the computer, the BIOS checks that hardware, memory and the C: RAM disk are intact, and that battery power is at least fair to middling. Then, and here's where the HP 95 differs from other DOS-machines, it returns to whatever task the computer was performing when it shut down — exactly where it left off!

This is fine if the computer was in the middle of something productive — or maybe it was idle, doing nothing at all. However, what if the computer was, ahem, indisposed, or maybe this is the first time it's been started, so nothing was ever running.

There's no way for the computer to tell if all is well. One machine language instruction looks pretty much like another; and if it does freeze, well, it's too late for it to be able to do anything about it. It's up to us to cold-start the computer should things go wrong. To straighten it out, we'll have to stop it dead in its tracks and force it to reboot; this is explained in *Resetting The Computer,* on page 19.

Mostly though, we don't have to think about booting at all.

Before Things Go Wrong

Short of dropping it in a lake, leaving it on the bus, or loaning it to a friend, there's not much that can go wrong with the HP 95 hardware. Still, bad things do happen to nice computers. Most computer problems are the result of software errors.

Programs do crash, but most data loss is the result of user mistakes… like accidentally deleting files. For instance, in the preparation of this book, the author lost a file on the HP 95 by trying to save the file in the Memo program, then not paying attention to the prompts. There was a two day old back-up copy, but a lot can change in two days.

Avoiding Problems

It's devastating to see a whole day's, a week's or even more, work lost to a careless keystroke or misbehaving software. If you haven't lost data yet, you will. A rule of thumb is to back-up files before you lose files. It's amazing how often we lose data just before we didn't get around to making back-up copies.

Any back-up scheme, regardless of how casual, is better than none. A separate back-up disk for each project makes it easier to update only those files that have changed since the last back-up.

You could accidentally back-up files in the wrong direction and overwrite the new files with old ones. You can minimize the damage by maintaining two back-up copies, then overwrite the oldest copy each time.

RAM disks are fast and protect data reliably, but they are not permanent storage. Check the manufacturer's recommendations for changing batteries, then err on the conservative side. Always try to make an archival copy of important files on floppy disks.

The HP 95 appears to be less susceptible to static electricity crashes than some earlier handhelds, but RAM cards are an unknown. Static crashes can happen, and they can damage computers as well as data. On days when the humidity drops and every time you touch a door knob sparks arc from the ends of your fingers, consider leaving your HP 95 at home.

Identifying Problems

Programs beep at us all the time. The challenge is distinguishing the informational and casual warning messages from the warnings of impending doom. Here's a short crash course to help you tell when the sky really is falling.

It can be difficult to discriminate normal, albeit bizarre, behavior from trouble. Before assuming something is amiss, check for obvious keystroke combinations (like **F1** for help), then take a look at the documentation. If something has gone wrong, checking out the possibilities before rebooting can save data and time. If you can't turn the computer off or on, there is something wrong; skip down to the section titled *Resetting The Computer*.

The only real problems are loss of data, corrupted disks, and the inability to run programs. The HP 95 maintains a checksum to ensure that the C: RAM disk is intact. Each time we turn the computer on or do a warm or cold boot, the computer verifies the disk. If the computer doesn't complain about disk problems, there probably aren't any serious problems.

Not all MS–DOS programs run well on the HP 95, but very few cause problems. For instance, Microsoft's MSD diagnostic program (designed to test computers) appears to lock-up the HP 95, but it is actually working for about fifteen seconds, apparently trying to decide what kind of new-fangled computer the 95 is. Similar programs from Norton and QuarterDeck run quickly, without problems. While MSD runs strangely, it does not harm the computer.

Damage Control

When you've decided that something is wrong, don't start banging on keys! The more keys you hit when the computer is acting out of sorts, the greater the chance there is of doing the only real damage that's possible on the HP 95: lost or damaged files on the RAM disk.

Just because you can't see anything on the screen (or you see garbage), it doesn't mean the program has crashed. The HP 95 display is 16 lines of 40 characters, and a warning may be hidden on line 25 or over on the right, beyond the edges of the display window. Try pressing **ALT**+Cursor keys to find and visual clues. If you can, run the program on a computer with a full-size display to check for messages. Try the normal methods to exit a program (though you're flying in the dark): **ESC** or **CTRL**+**BREAK** (on the HP 95 that's **CTRL**+ ⇐) will often work.

If the display is blank, the program my not have correctly identified the 95's display, and if it's writing directly to the screen (many programs do this because it's much faster than going through MS–DOS or the BIOS), it may be writing to the wrong memory location. This is a bigger problem on the HP 95 than some other computers because those memory locations may be reserved for other program data. If this happens, it's probably safest to reboot the computer to avoid the chance that a program in memory has been corrupted.

- You can't accidentally erase programs in ROM.
- You can accidentally *trash* (that's computer lingo for "corrupt") the permanent part of the C: RAM drive.
- Computers and software can lose data through battery failure or electrical storms. Create permanent, back-up copies of important files, and make new back-ups whenever you make any significant changes.
- In the words of Douglas Adams' *Hitchhiker's Guide to The Galaxy:* "Don't panic!"

Resetting The Computer

When we reset the computer, we stop what's happening then reboot. This includes the self-checks, reloading MS–DOS, and re-reading CONFIG.SYS and (if your computer uses one…) AUTOEXEC.BAT. It is not necessary to turn the computer off before rebooting.

Two Different Ways to Boot

Keystrokes	Purpose
CTRL+**ALT**+**DEL**	Also known as a "Warm boot." Resets the computer and restarts the operating system.
CTRL+**SHIFT**+**ON**	This is a "Cold boot." All data in memory will be lost. There is some possibility that files on the C: RAM disk will be damaged.

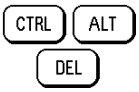
PC-compatibles are designed to be rebooted — desk bound PCs are booted at least once every day. You aren't damaging the computer by rebooting.

Rebooting the computer does not always mean something has gone wrong. Changing system configuration or loading TSRs often requires rebooting before MS–DOS will recognize the changes.

Caution

Before you reboot, be sure to exit from any running application programs if you can. If you don't (or can't) exit the programs, all un-saved data in the computer's main memory will be lost. If any application has a disk file open, the file may be corrupted.

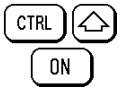
The Warm Boot



You can recover from most common problems by doing a warm boot. Press **CTRL**+**ALT**+**DEL** to restart MS–DOS. Main memory will be cleared, but there is little chance for problems with the C: RAM disk.

All MS–DOS computers recognize the **CTRL**+**ALT**+**DEL** keystroke sequence to reboot. The warm boot may fail, or even do nothing at all, with severe lock-ups. Just a few programs will try to intercept the warm boot (one older word processor, for instance, interprets that key-stroke combination to exit the program); for safety's sake, avoid those programs.

Cold Booting



A cold boot is rarely necessary; you can usually reset the computer with just a warm boot. If the computer won't warm boot, pressing **CTRL**+**SHIFT**+**ON** does a hardware reset.

On your desk bound PC, a hardware reset is the equivalent of turning off the computer then turning it back on (also known as "Hitting the big red switch"), with one exception: When it reboots, the HP 95 will ask you if you would like to reformat the C: RAM disk:

```
Initialize RAM Disk?  Enter Y or N:
```

When you reformat C:, all of your custom configuration files in C:_DAT\ will be erased. Even if you select **N** to bypass reformatting C:, there is still some chance that the C: RAM disk may be corrupted, forcing the computer to reformat it anyway.

Note

After a cold boot, the HP 95's clock is set to 1980; you will have to reset it to the current date. Why does it return to 1980 instead of zero? When the IBM engineers designed the original PC, memory was expensive; a bit saved was a bit earned. The year a file was created is stored as an offset from 1980: 1992 is stored as 12, 1993 is 13, and so on. By assuming that it will always be later than 1980 and that the PC will only be popular for 127 years, MS–DOS' designers managed to store the month, day and year in the same space that the year alone might have required.

See also

Page 86 has instructions on how to use CHKDSK to resurrect disk drives.

Notes About Rebooting

- Data *will* be lost if you reboot without saving and closing files. Close any open system manager applications before rebooting, this will assure you that all data has been written to the disk and all files are closed.
- If you can, try to make backup copies of important programs before rebooting.
- After a crash, reboot then run CHKDSK.EXE (see page 86 for instructions) to verify that the C: RAM disk is in good shape.
- If after all these attempts you still can't get the computer to boot, try removing the batteries (the backup battery and AC adapter, too) and letting the computer sit for a while. Then put the batteries back in the computer and try the **ON** key.
- Neither warm- nor cold-boots will erase a RAM card. However, if you reset the computer while files on a RAM card are open, data loss may result.

A Few Cautions

- Don’t turn the computer off while the beeper is on — this leaves the speaker circuitry in high-power mode. If you absolutely must silence a “stuck” speaker, you can press **CTRL**+**ALT**+**DEL** to do a warm reboot.
- The keyboard buffer remembers up to 15 keystrokes. With early versions of MS–DOS 3.20, overflowing the keyboard buffer (repeating keystrokes until it beeps repeatedly) could crash the computer. The HP 95 does not have this problem. Some applications will flush the keyboard buffer when you press **CTRL**+**BREAK**.
- Try to make back-up copies of important files before trying new programs.
- The cold boot (**CTRL**+**SHIFT**+**ON**) is not absolutely reliable. There is a slight chance of corrupting the RAM disk.

Compatibility

The two sides of the compatibility issue are data and hardware. The HP 95's suite of built-in applications minimizes the importance of hardware — and therefore software — compatibility.

Text files and Lotus 1–2–3 spreadsheet files are easy to move between the HP 95 and desk bound computers. Other HP 95 files use *Proprietary file formats.* This means that the data is organized following standards created by the folks at HP and Lotus.

There's one all encompassing solution to data compatibility. The HP 95 Connectivity Pack (Hewlett–Packard part number HP F1001A) converts appointment and phone book files to formats recognized by several popular PC software packages. It also includes versions PC versions of the HP 95's Appointment, Filer, Memo, Phone Book and HP Calc applications, which all but eliminates the file format conversion difficulties.

Software Compatibility

The unique HP 95 computer hardware makes software compatibility a thornier subject. The HP 95 should be able to run most MS–DOS-compatible programs. It is, after all, a PC-compatible computer.

The main concerns are disk storage, available memory and display size — or more accurately, a shortage of these ingredients. If you can make the program fit in the computer and the display characteristics are acceptable, the program is likely to run.

Many newer programs assume 512K available memory and a hard disk with several megabytes of storage available. While PC software companies try to develop programs that run on virtually any PC compatible, they're not likely to be very sympathetic if you cannot make their program fit in your computer.

Commercial software frequently comes with the disks sealed in an envelope covered with disclaimers and warnings. Once you've broken the seal, software is rarely returnable. When in doubt, call the distributor first and check for HP 95 compatibility — or if there is a special 95 version — for software you are considering.

Another problem is getting the software into the computer: you will undoubtedly need another PC with a floppy disk drive.

HP 95 Hardware Features

Some significant hardware differences between the HP 95 and the IBM PC will affect how software runs. Some of these differences are a curiosity, others are limiting. Let's review some features and quirks of our HP 95.

Display

- The LCD display is MDA (Monochrome Display Adapter) compatible. This is similar to how the display of the IBM PC Convertible and some other (older) laptop computers work. Most DOS application programs that aren't aware of the HP 95 will treat the display as 80-columns, monochrome device that cannot display graphics.
- The only color attributes recognized are black on white, inverse video, and flashing. The high intensity attribute is ignored.
- A 40x16 text window scrolls through a standard 80x25 screen.

- **ALT**+Cursor key tracking will interfere with programs that, for instance, check for the **ALT** key to be pressed to open a menu.
- The cursor doesn't flash, but instead alternately turns on or restores color attributes to underlying pixels. This will be apparent with programs that display inverse video (white on black), where the cursor will often be invisible. This may be changed in later HP 95 releases.
- The HP 95 displays graphics in a new, non-standard graphic mode, which was introduced just for this computer. Any graphics-based software written before the introduction of the HP 95 will not run on the HP 95 in graphics mode, and newer applications will have to be modified for the 95. Since most programs will recognize the 95 video adapter as MDA, which doesn't support graphics at all, they will be more likely to quit with an error message than try to use, for instance, Hercules monochrome graphics.
- The character set is slightly different from standard, MS–DOS characters. The HP 95 uses code page 850 (Multilingual, Latin 1), while most PC compatibles use code page 437 (English, also known as PC–8). This primarily affects double-lined box drawing characters and mathematical symbols. ASCII codes below 155, the characters we use most often, are not affected.
- The Print screen function (**SHIFT**+**ESC**) assumes an 80 column, 25 line display — even when we're running the built-in (40-column) applications. There is no parallel printer port. The BIOS sends the "screen dump" to the serial port.

Keyboard

- The HP 95 keyboard is PC–AT compatible; this allows programs access to special keystroke combinations like **CTRL**+↑, not recognized by the original IBM PC. It doesn't support some keystroke combinations or the **F11** and **F12** function keys or, of course, the alternate cursor pad. The AT sends **ALT**+Cursor combinations to applications programs, while the HP 95 intercepts those keystrokes to scroll the display window.
- Some programs that recognize **SHIFT**+Cursor combinations won't feel (or work) quite right on the 95. This should be of little concern because most of those programs are large, integrated packages or elaborate text editors.
- There is no **PAUSE** key. In MS–DOS and many utility programs, you can simulate **PAUSE** by pressing **CTRL**+**S**, though more sophisticated programs will ignore this work-around, or will use **CTRL**+**S** for other purposes.

MS–DOS/BIOS

- From a programmer's point of view, MS–DOS 3.22 is quite compatible with other versions. Most programs require DOS 2.1 or later; only a few require MS–DOS 3.1 or later.
- From a user's perspective, some features are missing. DOS 3.22 does not have some external commands. MORE, COMP and SORT utilities are missing — this can stop batch programs from working.
- Software that uses overlays or some floating point emulation may cause problems with conflicting interrupt vectors.
- Most interrupt handling is compatible with the PC standard. For instance, WINK.SYS, a cursor enhancer device driver, was designed just for the HP 95, yet it runs successfully on desk bound computers.
- The Phoenix BIOS, while customized for the HP 95, is the recognized leader in performance and compatibility for desk bound, laptop, and now palmtop PCs.

Memory

- Memory is partitioned between conventional and RAM disk. If you use a large RAM disk, there may not be enough conventional memory available to run larger DOS application programs.

- EMS (Expanded Memory Specification) support is currently not available on the HP 95. While you might try to use an EMS emulator that swaps data between main memory and a RAM card, the page frame reduces conventional memory by 64K — unless you bank switch into higher memory addresses, which will require a special HP 95 driver.

RAM Disks

- Software is constantly changing and growing. "Version 1.0" of a program might have fit comfortably on a single disk. The software version number seems to reflect less the innovations of the product than the number of disks in the box… until you get past version 6.0, when each new revision adds two more disks. Even with a two megabyte RAM card, many programs just won't fit in the HP 95.
- Most fixed disks formatted for MS–DOS store files in units of 2048 bytes. For disk efficiency, the HP 95 formats the C: drive to store files in units of 512 bytes. This causes no problems, and actually saves disk space.
- Software uses many methods to describe disk drives; these methods may occasionally misinterpret the HP 95 RAM disk. Version 6.01 of Norton Utilities, for example, recognizes the C: RAM drive as a 360K non-removable floppy disk (floppy disks are always removable).
- The C: RAM drive identifies itself to software as media type FA hexadecimal (250 decimal). This value corresponds to a 5¼-inch, 1-side floppy disk with 8 sectors per track, for a total capacity of 320K. The actual size of the RAM disk, when we include the files in ROM, will usually exceed 700K. This discrepancy can confuse programs, especially file recovery utilities. Use these programs with great care.

Lotus 1–2–3 File Compatibility

There are at least four versions of Lotus 1–2–3 available — saying the HP 95 includes 1–2–3 isn't very descriptive. Version 2.3 is an upgrade of the HP 95's version 2.2. Version 3.1 is a powerhouse product — requiring powerhouse hardware. Variations on 1–2–3 are also available for other hardware platforms and operating systems. Finally, *Lotus 1–2–3 For Windows,* (nicknamed *123W)* was released in September of 1991. 123W was met with mixed reviews, though it's an easy to use, powerful, graceful product, which should be easy for the HP 95 user to adopt. Most 123W display and memory management issues were addressed by the end of 1991.

Exchanging 1–2–3 Files

If you are transferring spreadsheets to the HP 95, be sure to save the files in WK1 format. Attempting to load a file from version 3.1 or Lotus 1–2–3 for Windows version 1.0 will cause the "Worksheet file revision is out of date" error.

Many other popular programs, such as Borland Quatro Pro, Microsoft Excel for Windows and Microsoft Multiplan, can convert files to and from the standard Lotus WK1 file format. If a program can import or export only a single spreadsheet file format, the format will surely be Lotus WK1.

Lotus Spreadsheet File Name Extensions

Lotus Version	Extension
Release 1, 1A	WKS
Release 2.0, 2.01, 2.2, 2.3	WK1
Release 3, 3.1, Windows 1.0	WK3
Symphony Release 1, 1.01	WRK
Symphony Release 1.1, 1.2, 2	WR1

Note

When you convert spreadsheet files to the WK1 format, be sure your files do not use any features that are unique to the specific spreadsheet version. Some data or formulas may be lost when you translate to WK1. Be sure to split-up WK3 files that contain multiple spreadsheets.

Say you are using 123W, select the File, Save as menu and change the WK3 file extension to WK1. With Microsoft Excel, version 3, select File, Save as, then click on Options; finally, select the WK1 item from the File Format list. Now that the file is in WK1 format and Windows is running, switch to the Windows Terminal program and zip it over to your 95.

About The Future?

Ensuring that software runs on all of today's disparate computer hardware is a nerve wracking experience for the software developer. Where once we had a single IBM PC with a just single CPU (the Intel 8088, a member of the 8086 family), today we have a dozen processors — and that number is quickly rising as the processor war takes-off. Throw in a half dozen video standards, several keyboards, mice, and disk drive interfaces.

Where once software tried to keep up with hardware, now it's hardware that's straining to "simulate" compatibility. Software developers follow some rules (well, mostly), and hardware follows those guidelines. The key to compatibility is adhering to accepted standards.

It's a tough game to play: hardware and software engineers are a creative and diverse lot, but they're painting with a restricted palette, oft times stuck in (or painting into…) a corner. For the sake of compatibility and safety, changes are evolutionary, and rarely as innovative as anybody wants. To paraphrase George Will, making major changes is "Like making the transition from driving on the left side of the road to the right… a little bit at a time." A lot of people will find themselves sitting at the side of the highway. Natural selection often takes time — but it can happen in a generation.

It's been more than a decade. The halcyon days the 8086 have passed. All recent PC compatible computers, the HP 95 included, can run software written for the original IBM PC, but the process of natural selection has changed the standard. New, innovative software takes advantage of — and suffers without — high-powered hardware. Lots of memory and disk space (a valued resource on the HP 95) are taken for granted.

It's an issue of comfort more than compatibility. If you can, select software designed or customized for the HP 95.

Resource Management

Plugging in cards adds programs and storage space, but the HP 95 is still a sealed box. The goal, then, is to stretch these finite resources.

Memory

You paid for 512K of memory. Half of it is a RAM disk, and Lotus 1–2–3 says there's 100K available. So where's the rest?

The only way to keep the whole 512K available is to not turn on the computer. Everything, even turning on the HP 95 and displaying the current time, uses memory. Just booting the computer, before you start any of the built-in applications, takes at least 100K of memory.

Because of the internal design of the HP 95, it's unlikely that main memory will be expanded much beyond 512K. Memory addresses that remain unused on other PC-compatible computers are already spoken for on the HP 95, for things like bank switching of data and programs. Expect to see EMS (Lotus, Intel and Microsoft designed this Expanded Memory Specification), which adds extra memory for Lotus 1–2–3 and several MS–DOS applications.

Application Program Memory

Main memory is a common pool. When a program or command needs more memory, it requests a block. When it no longer needs the memory, it returns it to the pool. When there's not enough memory available for an application, somebody has to get out of the pool.

The two memory categories in this table are Memory and Data Limit. The Memory category is the amount required to run the program, without concern about data files. The Data Limit column is how much additional memory the program may use to load and edit data files.

Built-in Application Memory Requirements

Application	Memory	Data Limit	Practical Limit
Appointment Book	20K	47K	About 500 entries.
Clipboard	·	8K	220 lines, 1300 words.
Data Comm	20K	·	·
Hearts & Bones	9K	·	·
HP Calc	24K	·	·
Lotus 1–2–3	30K	Memory	Available memory.
Memo	15K	51K	1400 lines, 8700 words.
Phone Book	12K	49K	About 450 entries.
Tiger Fox	11K	·	·

Notice that there are no entries for Filer or Setup. These built-in applications and the System Manager reserve memory for several important functions. To avoid deadlock, there's always enough additional memory set aside so we will always have room to run Setup and Filer.

While Lotus is listed in the table as requiring only 30K, remember that much of the system overhead is for 1–2–3, regardless of if the program is running or not. In addition, Lotus will refuse to run if available memory falls below about 48K, and it will flash the "Mem" warning annunciator if memory falls below 4K while it runs.

Lotus 1–2–3 can create files as large as available memory. Always be sure there's enough disk space available to save spreadsheet files that are currently loaded into main memory.

Checking Available Memory

The Setup program shows how memory is divided between the C: RAM disk and main memory, but it does not tell us how much memory is currently available.

- In Lotus 1–2–3, press **MENU** then select Worksheet Status. The amount of available memory shown will be reduced by the memory required to run 1–2–3.
- In Filer (or at the DOS command line) run CHKDSK.EXE. The difference between total memory and bytes free will be around 98K if the System Manager is loaded, or 18.5K if you booted from plain-old DOS.
- You can see how much disk storage space is available by switching to Filer and running the DOS program CHKDSK.EXE. It also shows currently available main memory.
- Instead of CHKDSK, try running MEM.COM, included with this book. MEM shows only information about memory, not disk drives, and it runs a bit faster.

See also

Page 33, *Configuring Memory,* has instructions for changing RAM disk size.

How Much Memory Is Enough?

It's frustrating, after sailing along with a 20K Phone Book database and a 15K Memo text file open, to have the computer refuse to load Lotus 1–2–3 because of insufficient memory. We can close an application then load 1–2–3, but if we could allocate all of our 512K RAM to main memory, it would be a cinch to run everything at the same time.

The more memory you have available, the more applications you'll be able to leave open concurrently. HP 95 memory management is a trade-off between C: RAM disk and program memory. The HP 95 ships with the C: drive configured to 254K.

The Setup program lets us vary this configuration between giving virtually all of our 512K of memory to the RAM disk, and all to main program memory. This table compares RAM disk and main memory allocation, and how the results affect Lotus 1–2–3.

C: RAM Disk Vs Main Memory

C: RAM Disk	Main RAM	Lotus 1–2–3
30K	482K	328,864
62K	450K	296,096
122K	386K	230,560
246K	266K	107,680
254K	258K	99,488
262K	250K	91,296
270K	242K	83,104
310K	202K	42,144

All of the built-in applications edit data in RAM. It's one of the great ironies of the HP 95 that data is copied to RAM for editing, then copied back to RAM disk for storage — and it's all part of the same memory. For each 10K Memo file, we need another 10K of disk storage space for the permanent copy.

Most Lotus spreadsheet files will probably never reach, let alone exceed, 100K unless you use your HP 95 for Fortune 500 accounting. Phone Book files and Memo note files are rarely larger than 10K. Other applications limit data files to under 64K each. At 200-300K, dictionaries are well within the capabilities of the HP 95, while this book, at over 700K, probably is not.

Evaluating memory requirements is a personal, trial-and-error process. The first rule is there's no such thing as enough memory.

1. If you work with several smaller programs and small data files, increase the RAM disk partition as far as you can until the built-in applications you regularly use won't run concurrently.

2. If you work with a few very large files or many small files, you are a candidate for a RAM card. You will probably want to reduce the C: RAM disk to 62-122K to make more room available for large files in main memory, then add at least one large RAM card to store your data files.

3. The typical user will start with the standard memory partition, deleting or copying unneeded files to another computer as necessary. A RAM card will be indispensable for permanent data storage.

Conserving Memory

- Use tab characters (entered with the **TAB** key) to indent text in Memo text files to save memory and disk storage space. A single tab character can replace as many as eight spaces; do it several times in a Memo file and the savings will quickly mount.
- Turn off Lotus 1–2–3 Undo. While 1–2–3 is running, press **MENU**, then select Worksheet, Global, Default, Other, Undo, then select Enable or Disable. The Default Settings screen shows the current 1–2–3 Undo status.
- Close unneeded applications.

Notes About Memory Management

- Don't create spreadsheets (or any files) larger than available RAM disk space. Use a plug-in RAM card then reduce the size of the on-board RAM disk.
- You can increase available memory by about 1056 bytes by reducing buffers in the CONFIG.SYS file. See page 67.
- If you're short on memory but still want to run Lotus 1–2–3 alongside several other applications, start Lotus first, then the other applications. Lotus will confirm that there is enough memory and load normally, then you can switch to other applications, and Lotus will share the available memory with these applications.
- There are many powerful tools built-into the HP 95. If memory is at a premium (which, of course, it always is), consider using the built-in applications programs instead larger, non-95 programs.

RAM Cards

Plug-in RAM cards might really be a panacea. At least they offer several advantages with virtually no downside. Besides of course increasing disk storage, RAM cards free you to configure most of the main 512K as program memory and less as a RAM disk; this lets you build larger 1–2–3 spreadsheets and run memory-hogging DOS programs.

The HP 95 supports RAM cards as large as two megabytes. If you're using mainly the built-in applications, this is probably much more than you'll need. For DOS applications, the sky's the limit. When you're deciding what size RAM card to buy, remember that your needs will likely grow with your interest and understanding of your computer. Storage space requirements *always* grow to exceed capacity.

We currently have 128K, 512K and 1MB (megabyte) and 1.5MBcards, with 2MB cards on the way. If you add too little, you can always add another card later, perhaps dedicating a separate card for each application. That the cost per kilobyte is usually less with larger cards.

RAM Cards For Security

If you don't have another computer for permanent data storage, or your HP 95 spends most of its time away from the desktop, you'll appreciate, and soon depend on, the convenience and security of a RAM card.

Ram cards are as close to archival storage as the 95 gets. Keeping important data on cards lessens the chance of file corruption or data loss when the power fails or a program goes haywire.

All of the 512K of internal memory, regardless of if it's configured as main memory or RAM disk, is located in a contiguous block; a "runaway pointer" crash that might normally just hang the computer could also garble data on the internal RAM disk.

Notes About RAM Cards

- The HP 95 supports RAM cards as large as 2 megabytes. Cards larger than 2MB would be a needless expense since the 95 would not recognize their extra capacity.
- 8K of memory for the HP–75C Portable Computer listed for $195; ten years later 128K for the HP 95 lists for the same price. Some predict geometric growth in density and reduction in cost; taken too literally, a bathtub-full of storage would come free with your morning newspaper. But free or not, memory is a housekeeping responsibility.
- A one megabyte card may cost more than twice as much as 512K.
- The larger the RAM card, the shorter the card's battery life.
- Good housekeeping habits (discarding outdated, unneeded or duplicated files) can forestall the purchase of additional RAM cards.
- RAM cards are more secure than HP 95 main memory and the C: drive, but they are not infallible. If you can, periodically make back-up copies of RAM cards on permanent, off-line storage (usually floppy disks).
- Save room on the C: RAM disk by erasing files from C:_DAT\ that are duplicated in the hidden C:_SYS\ directory.

Power Management

The 95 has a strong sense of self-preservation. It goes to elaborate extremes to conserve battery power — and our data. We'd have to try really hard to lose data. The 95 checks battery level throughout the day:

1. Whenever you turn on the computer.
2. Once each minute while the computer is running.
3. Whenever you run the Setup program.

Even when you're not, the computer is busy. It's constantly checking the keyboard, serial port and watching for 5 minutes to pass so it can take a serious nap. To extend battery life, it takes a short nap, called *light sleep,* whenever programs check for waiting keystrokes. This means that even programs not specifically designed for the HP 95 can conserve power if they are polite and let the BIOS handle the keyboard buffer (most programs are polite that way). Programs that sit in tight loops and check the keyboard through non-standard methods will have higher power drain. For instance, TF.COM seems to cause higher than average battery consumption.

Judging Battery Life Expectancy

When new, the main batteries supply about 3.0 volts (two 1.5 volt batteries in series). When you press **SHIFT+FILER** to run Setup, the screen shows the current system battery level as a "Gas gauge," with "F" and "E" markers. Full means three volts, while empty is 1.8 volts. The battery gauge feels, at best, vague. Each notch represents a large jump, and it doesn't show back-up battery level at all. If, like me, you're used to putting in five dollars-worth and driving 'til it's pegged again, the 95's gauge will take some vigilance.

See also

We've included a program called BATTERY.COM that shows main and back-up battery levels to $^1/_{10}$ volt accuracy. See page 94 for program instructions and listings.

The magic value is 2.5 volts, which is about halfway down the battery gauge. When system batteries dip below 2 volts, the computer displays "Low Main Battery" each time you power-up, until voltage rises above 2.5 volts. Batteries do recover a bit naturally — though it's unlikely they'll recover from 2.0 to 2.5 volts.

When the battery level falls to 1.8 volts, the computer's fail-safe mechanism kicks-in. Here's where the backup battery gets its workout, because the computer shuts down and relies on the backup battery until you replace the main batteries. Empty on the 95 means shut-down and curl up into your clamshell, and wait for new batteries.

A good rule of thumb is to start looking for batteries when you're at a half-tank.

Notes About Power Management

- Don't use NiCad (Nickel Cadmium) batteries. Most NiCads produce 1.2 volts when fully charged, for a total of 2.4 volts. The HP 95 low-battery warning starts at 2.5 volts.
- Use BATTERY.COM for a more accurate check of main and back-up battery levels.
- Power consumption increases whenever Filer is running or during any serial communications (using the Comm program or DOS' CTTY).
- Use the AC adapter, if you can, whenever you are doing serial communications.
- The computer will not time-out when it's plugged-into the AC adapter.
- Don't leave impolite programs running when the computer is turned-on. These are programs that sit in a tight loop and either don't check for waiting keystrokes or check for them via non-standard methods (such as looking directly at the keyboard buffer). Unfortunately the only way to see if a program is a battery hog is to try it.

Midnight Madness

The HP 95 does housekeeping chores every night at midnight. If the computer is off, it will turn on for less than a second. If the computer is on, the running program will not respond to keystrokes for a second or so. The only time this might be an issue is when the HP 95 is being used for automated data collection. Try to time readings to skip the midnight rush.

Configuring and Customizing

In this chapter we'll talk about several files and computer settings, and note how changing these settings affects computer performance and memory efficiency.

Configuration Files

We're usually so busy making notes and calculations that we forget that the computer has needs, too. It lives for files. A few important files customize how the HP 95 works:

HP 95 Configuration Files

File Name	Purpose
C:_DAT\APNAME.LST	List of System Manager programs and keystrokes.
C:\AUTOEXEC.BAT	Batch file, run when booting MS–DOS.
C:\CONFIG.SYS	Loads device drivers, configures environment.
C:_DAT\TOPCARD.PCX	The Business Card graphics file.

The computer automatically creates TOPCARD.PCX. To get the most of the 95, you'll want to create custom versions of all of these files; we'll offer suggestions on how to set-up these files later in this book.

Files in the next list are created and maintained automatically by the built-in applications. These files hold program settings and data; for instance, the calculator storage registers are stored in CALC.ENV.

Application Configuration Files

File Name	Application
C:\123.CNF	Lotus 1–2–3.
C:_DAT\APPTBK.ENV	Appt.
C:_DAT\CALC.ENV	HP Calc.
C:_DAT\HEARTS.CNF	Hearts & Bones (HB.EXM).
C:_DAT\MEMO.ENV	Memo.
C:_DAT\PHONE.ENV	Phone Book.
C:_DAT\SETUP.ENV	Setup.
C:_DAT\TIGERFOX.CNF	TigerFox (TF.COM & TFOX.EXM).
C:_DAT\WATCH.ENV	Appt.

If you delete these files, the applications will lose their current settings, and will revert to the factory configuration. For instance, after you've been playing with TigerFox for awhile, you can delete TIGERFOX.CNF to reset the high score to zero.

Caution

Don't delete 123.CNF, it's needed by Lotus 1–2–3, and SETUP.ENV (containing global settings like the beep level) is write-protected so you can't accidentally delete it; don't try to edit or delete this file. Don't try to edit the *.ENV files with Memo. These are not text files; altering them can cause the associated applications programs to work erratically, or maybe not at all.

Most of computing is data manipulation. HP 95 built-ins recognize several file types.

Applications Data Files

File Name	Application
*.ABK	Appt program.
*.CTF	Comm program character translation.
*.DCF	Comm settings.
*.EQN	HP Calc Solver file.
*.FCF	Filer configuration.
*.LCF	Comm logon configuration script.
*.PBK	Phone Book.
*.PCF	Printer configuration.
*.PCX	Business card graphic file (Setup).
*.PIC	Lotus 1–2–3 graph file.
*.TXT	Memo text file.
*.WK1	Lotus 1–2–3 spreadsheet.

See also

The table on page 112 lists many common file name extensions.

The sample 1–2–3 files in the root directory with names beginning with an underscore and the WK1 extension cannot be deleted, nor can you erase CHKDSK.EXE, COMMAND.COM or TF.COM. Since these files are in the ROM portion of the C: drive, they consume no usable disk space (though they do take directory entries, which are a perishable commodity, because the root directory can only have 100 files).

Note

The first time you booted your HP 95, it created the C:_DAT\ directory and copied several files there. These files include *.DCF, SAMPLE.CTF, TOPCARD.PCX, and _1.PBK. None of these files are needed to operate the computer (and most of them aren't very interesting). You can safely delete these files to conserve disk space; if you later decide you need them, there are copies available in the hidden C:_SYS\ directory.

Turning Down the Beeper

So your coworkers think you're playing with a Nintendo Game Boy whenever you balance your expense account in Lotus 1–2–3. That's easy to fix: turn the beeper down. You probably don't want it off completely — some warning beeps are important — just make it a bit quieter.

Changing the Speaker Volume

Select Item	What it Does
SHIFT+FILER	Enters the setup program.
System	Enters the system configuration sub-menu.
ENTER	Selects the first item (Volume).
← or →	Press the left and right cursor keys until the volume is to your liking. There are four volume levels, plus silent.
ENTER	This enters the volume setting. You could press **Esc** instead to cancel the volume change.
Q	Quit the System menu, return to the Setup menu.
Q	Quit the Setup program.

Notes About the Beeper

- The first time you boot your computer, the beeper was set to the factory level (a bit too loud for the office, a bit too quiet for a car on the freeway). If you change the volume with the Setup program, your preferred setting will be maintained in the file C:_DAT\SETUP.ENV, so each time you reboot, your custom volume setting will be restored.
- Turning off the speaker completely may conserve batteries slightly, but you may miss important warning messages.

- You can turn off warning beeps just in Lotus 1–2–3 without affecting beeps elsewhere. The command is *six* levels deep in the Lotus menus, follow this procedure to keep from getting lost in the maze of menus: After starting Lotus 1–2–3, press **MENU** then select Worksheet, Global, Default, Other, Beep, No; then press **ENTER**. After you press **ENTER**, you will be returned to the next menu level higher (The "Printer Directory Status Update Other Autoexec Quit" menu); in this menu, select Update to change the default setting (so Lotus will start in quiet mode next time), then select Quit to return to your spreadsheet.

Configuring Memory

It is possible to partition memory and RAM disk space so there is insufficient memory to run Lotus 1–2–3. The minimum allowed by Setup is 310K for disk and 202 for main memory.

Before re-partitioning memory between main memory and the C: RAM disk, all applications besides Setup must be closed. This is to ensure that no data in memory will be lost when Setup reboots the computer after changing the memory partition.

Changing Memory Allocation

Select Item	What it Does
SHIFT+FILER	Enters the setup program.
System	Enters the system configuration sub-menu.
Memory	Enables the memory "volume" control.
← or →	As you press the left and right cursor keys, the Memory volume level and the numbers in the lower left corner will change. Continue until you reach the desired setting.
ENTER	Confirms that you would like to re-partition memory.
ENTER	At the prompt, press **ENTER** to confirm the memory change. If you press **ESC**, the change will be canceled.

See also

Page 26 lists memory requirements of built-in HP 95 applications and suggestions for re-sizing main memory and the C: RAM disk.

The HP 95 Business Card

When all applications are closed-down and the HP 95 doesn't have anything else to do, it displays a graphics screen, called a Business Card, which is stored in a file named C:_DAT\TOPCARD.PCX.

When you booted your 95 the first time, it placed a copy of the standard TOPCARD.PCX file in the C:_DAT\ directory. You can create your own custom graphics file or save some disk space by deleting this file and using one of the two other graphic files built-into the computer. Don't worry about erasing TOPCARD.PCX, there's always a spare copy in the hidden C:_SYS\ directory. You delete the file with the Filer program, or exit to DOS and type this command:

```
DEL C:\_DAT\TOPCARD.PCX
```

If you don't mind not seeing your name and the time on the screen, you can use the copy of TOPCARD.PCX stored in ROM in the hidden C:_SYS\ directory. Press **SHIFT+SETUP**, then select Owner, Picture-File. Now type the new file and path name:

```
C:\_SYS\TOPCARD.PCX
```

The other built-in Business Card file is a thousand dollar bill with Grover Cleveland's picture (though the picture looks more like Grover from Sesame Street). The U.S. Mint has stopped printing this note; this may be the only place you'll find one. Try this hidden Picture-File:

```
C:\_SYS\1000.PCX
```

It really doesn't matter what's in the graphic file you use; if you'd like, you could copy the thousand dollar bill to the TOPCARD.PCX file name, and display the time over it (showing the time on that picture isn't very pretty, but it works). From DOS enter:

```
COPY \_SYS\1000.PCX \_DAT\TOPCARD.PCX
```

Note

The HP 95 will display the current time and date over your graphics screen only if the file is named TOPCARD.PCX, and it must be in the C:_DAT\ directory. If the computer can't find TOPCARD.PCX, it displays the time and date, but no graphic file.

By the way, once you've closed all applications to have a look at your Business Card, take a moment to see a short tribute to the creators of the HP 95. Press **ALT**+**123**, then release those keys, then press and release them again; the names will scroll by. If you want to see the list again, press **ALT**+**123** one more time.

The HP 95 doesn't have a built-in paint program, but you can create Business Card files on other computers. Be sure the files are in the PCX format, monochrome, and no larger than 240 by 128 pixels (in Microsoft Paintbrush, that's the same as 2.50 by 1.33 inches). You can create PCX files from scratch, but it will often be easier to start with an existing file — even if you erase the entire image and start again. Try copying TOPCARD.PCX or 1000.PCX to your PC to use as a starting point.

Notes About the Business Card

- Pixels on the 95 are wider than they are tall. If you create PCX files for the 95 on a square-pixel computer (such as a PC with a VGA video card), be sure to adjust the drawings to avoid turning circles into watermelons.
- To eliminate the picture file, run the HP 95 Setup program and delete the file name from the Owner screen. This won't delete the file from your disk.
- The HP 95 will only show the current time and date if the Business Card file is C:_DAT\TOPCARD.PCX. Any other file name or any other directory will suppress the running clock.

The System Manager

Imagine adding new blue application keys to switch between built-in applications and other programs. Say, switch from Lotus 1–2–3 to TigerFox and back again without quitting either. Well, you can! We'll show you how to run TigerFox (and the new program Hearts&Bones) on page 37, but first an introduction to the HP 95 System Manager.

The System Manager is an applications program standard that offers tools and consistency for the built-in HP 95 applications. In addition, the System Manager is extensible, making it possible to add new applications to the computer that work exactly like the built-in programs. These new applications are written to follow the same standards, called *System Manager Compliance.*

Benefits for the user include reduced memory requirements, more consistent user interface, and non-preemptive multi-tasking. The one disadvantage is that programs will have to be modified, sometimes considerably, to conform to the System Manager. It's possible, as in the case of TigerFox, to create two versions of the same program: MS–DOS and System Manager.

The bulk of program development time — and the bulk of programs themselves, in fact — is taken up by the user interface. Every programmer has a better idea about your computer and your programs should work. The System Manager adds some stability so that we don't end up

with a computer full of uniquely, and sometimes arbitrarily, different program designs. System Manager programs usually work in similar ways because they're written with the same tools.

Writing System Manager compliant programs is philosophically not unlike writing Microsoft Windows applications. Though without the complexity, and without the investment in learning and tools required for Windows.

Conforming to the System Manager standards doesn't constrain programmers, but instead frees them to concentrate on the important, individual, creative issues of program design. Basically, the System Manager takes care of the large, tedious, expensive part of the task, and leaves the fun to the programmer. If the HP 95 is as successful as early reports indicate, expect new, innovative, and efficient programs designed for the HP 95, not just cobbled-up from another machine.

System Manager Program Types

The HP 95 System Manager adds a couple of new program file types: EXM and XIP. You don't run these programs from MS–DOS, but from special *hot keys* that you designate. Just as you start Lotus 1–2–3 from the dedicated **123** hot key, you can add new programs and assign them to the key code combination of your choosing.

EXMs (there are two of 'em hiding in C:_SYS\) are similar to the 95's built-in applications: you switch to them with a special keystroke, and you can switch to another program without exiting them. One difference, though, is they are disk-based programs; when you run them, the computer loads a second copy from disk into main memory, just like conventional MS–DOS applications programs.

XIP programs are *exactly* like the built-in applications. XIP is an acronym for "eXecute In Place," meaning the programs reside on a RAM disk card, and the computer can execute programs from the card — without copying them into main memory! This saves memory and speeds program loading.

The PCMCIA (the Personal Computer Memory Card International Association) is creating standards for XIP software. As this standard develops, expect to see more applications for the HP 95 available on plug-in cards.

Assigning Hot Keys

For the HP 95 to take advantage of new System Manager applications (and to even know they exist), we need to create a text file with the program name and hot key. This file is named APNAME.LST, and it must be in the C:_DAT\ directory. You can edit the file with the HP 95 Memo program.

The APNAME.LST file is quite simple: it's a text file with one line for each System Manager compliant program. Each line includes three items separated by commas: a fully qualified path and file name exactly as you find it in the disk directory, the hot keycode (in hexadecimal), then the title of the program. Each line may be up to twenty-eight characters long. Here's the format of each line in the file:

```
PROGNAME.EXM,hotkey,program title
```

Hot key codes in APNAME.LST are four-digit hexadecimal numbers. The first two digits are the scan code — traditionally, though not always, related to the position of the key on the keyboard; for instance, **Esc** is the first key in the upper-left corner of the keyboard, so it's 01. The third and fourth digits are the character the key returns when pressed; most special keys have no character code, so this value is usually 00.

See also

An introduction to hexadecimal numbers begins on Page 12. Look for the keycode table on page 106.

Say you would like to assign a program to the **E** key. Regardless of if it's shifted or not, the **E** key always returns scan code 12 (in hexadecimal). Depending on which state keys are down,

the character code could be 00 (for Alt+E) or 05 (That's **CTRL**+**E**) or the hexadecimal representation of "e" or "E" or even "æ" (that's **CHAR**+**E**).

E Keystroke Combinations

Keystroke	Character	Keycode
E	e	1265
CTRL+**E**	♣	1205
SHIFT+**E**	E	1245
ALT+**E**	(none)	1200
CHAR+**E**	æ	1291
SHIFT+**CHAR**+**E**	Æ	1292

You use any key with a character code of zero for a hot key. The only key in this list with a zero character is **ALT**+**E**. So to assign a program to **ALT**+**E**, you would specify the hot key as 1200. For a System Manager program named STARWARS.EXM, the line in C:_DAT\APNAME.LST looks like:

```
C:\STARWARS.EXM,1200,Star Wars
```

See also 

We'll show how to create hot keys for the System Manager versions of TigerFox and Hearts& Bones on page 37. The ASCII codes table on page 107 lists the HP 95 character set and keystrokes used to enter these characters.

Some of the more useful keystroke combinations are (putting it tactfully) arcane, so refer to the lists on page 106. **SHIFT**- and **CTRL**-modified regular keys are already taken, so this table lists just **ALT** or **CHAR** modified key combinations.

Once you've created and saved the your key assignment list to C:_DAT\APNAME.LST, close any open applications then press **CTRL**+**ALT**+**DEL** to reboot the computer.

Adding System Manager Sessions

Wouldn't it be great if your favorite program could edit two files at once instead of just one? Or maybe two people could switch-off playing TigerFox — with different game boards running at different speeds. Don't complain to the programmer, fib to the computer: tell it you have two editors or two games, when you really only have one.

System Manager compliant programs that use different data files, such at text editors, and programs that use no files at all and no shared system resources, such as many games, can frequently be assigned to more than one key at the same time. This means that you can run more than one copy of a program (called an *instance)* at the same time.

Each instance is invoked (loaded) separately from its own hot key, and uses its own memory. And you can switch between each copy with the hot keys. This APNAME.LST example tells the computer to load Hearts&Bones via the **ALT**+**H** keystroke, and a second copy when you press **ALT**+**B**:

```
C:\_SYS\HB.EXM,2300,Hearts&Bones
C:\_SYS\HB.EXM,3000,Bones&Hearts
```

Remember, though, that when programs exit they often create or update the same configuration file, so settings from the last one out of memory will prevail. Keep in mind also that each is a separate program, oblivious to the others' existence, so each will consume the full amount of memory required by the complete program — there is no code or data sharing.

If a program can make use of data from the clipboard, you can share information between instances by shuttling it back and forth through the clipboard.

Notes About The System Manager

- The APNAME.LST file must be in the C:_DAT\ directory.
- After you edit APNAME.LST, you must reboot the computer before the changes will take effect. First close all open applications, then reboot by pressing **CTRL**+**ALT**+**DEL**.

- You can't change the name of a program file to the EXM extension and expect it to run!
- Documentation that comes with application programs will state whether they are System Manager compliant or not.
- If you load any TSR programs from Filer, the computer will no longer be able to recognize your custom key assignments. You should load TSRs from AUTOEXEC.BAT or, if possible, as device drivers from CONFIG.SYS.
- Don't copy XIP programs from plug-in cards to the C: drive. Besides violating copyrights, program files can become fragmented and they will undoubtedly crash the computer when you try to run them.

Notes About Hot Keys

- Don't assign programs to important keys. For instance, changing the **MENU**, or **FILER** key could leave you dead in the water, with no way to recover short of doing a cold boot and re-initializing the RAM disk.
- You cannot reassign keys that return standard ASCII character codes below 128.
- Try assigning the same program to two different hot keys. This will give you access to multiple copies of the program, and you can edit different files with each copy. Remember, though, that each copy of the program runs independently, and will require additional memory.
- Avoid assigning programs to Alt-modified character keys. **ALT**+**H** is mnemonic and easy to remember, but it's also likely to conflict with another program (some programs use **ALT**+**H** to open a help screen). Remember that Lotus 1–2–3 uses **ALT**+ letter key combinations to play-back macros.
- You can hide applications by assigning them to unlikely key combinations (who would ever think of typing **CTRL** plus the equals key?).

Calling Hearts&Bones and TigerFox

Everybody wiles-away idle moments with the HP 95's TigerFox game; just switch to Filer and run TF.COM. English-language HP 95s include *two* versions of TigerFox: the one root directory in the file named TF.COM, and a second named TFOX.EXM, hidden in the C:_SYS\ directory.

Also in the C:_SYS\ directory is another, more challenging game, called Hearts&Bones, in the file named HB.EXM. With these new EXM programs, thanks to the Applications Manager, we can squander time more conveniently: while other applications are running!

These programs are *Applications Manager compliant.* You can switch between them and any other built-in application program. You cannot run these programs, or any program with the EXM file name extension, from the DOS command line (though, of course you can run TF.COM).

See also

Creating the APNAME.LST file and custom hot keys are discussed on page 35.

Before you can run these programs from hot keys, you will first need to create a file named APNAME.LST containing the hot key assignments. The APNAME.LST file must be placed in the C:_DAT\ directory. Say you want to assign TigerFox to the **ALT**+**T** hot key, and Hearts&Bones to **ALT**+**H**, here's what the APNAME.LST file looks like:

```
C:\_SYS\TFOX.EXM,1400,TigerFox
C:\_SYS\HB.EXM,2300,Hearts&Bones
```

After you reboot your computer, everything will look the same, but now when you press **ALT**+**T**, TigerFox will pop-up, even if you're in the middle of balancing your checkbook with Lotus 1–2–3. To make it even more cool, you can switch between TigerFox, Hearts&Bones, and any other built-in applications with the new hot keys.

Programming the HP 95

Lotus 1–2–3 is often called “The programming language for non-programmers;” when you consider Lotus macros, maybe it is a real language. HP Calc is malleable, especially in regard to the calculator’s HP–Solve function. Of course, DOS’ batch command language fulfills most of the technical criterion of a real programming language, but many people discount batch because of the mediocre performance and limited data and control structures.

Writing software to run on the HP 95 and other PC-compatible computers requires understanding of the underlying hardware. At this level of complexity, anyone who tells you he’s an expert on PC-compatibles is most probably misinformed. The only way to grow is through conscientious ignorance: learn what not to learn, and study the rest well.

Creating useful programs is still an individual process, but competing requires tools, time, understanding, peace and passion.

The HP 95 Programming Environment

The Fiat 850 Spyder, circa 1970, was perhaps the ideal sports car: beautiful, small, nimble, fun and quirky. You could drive it flat-out without getting tickets. The HP 95 is much the same. It’s fun in that you have to write efficient, elegant code (no room for bloated, inefficient languages here). It’s quirky in that… well just look at it, the AC adapter is larger than the computer.

Programmers have long used the best or largest or fastest machines available when writing software and manuals — in fact some new development tools *require* at least an 80386 processor. The problem arises when the developer uses only high-powered, high-ticket, high-bulk machines for testing; it’s up to the end user to discover that the program is a slug. The ideal place to test HP 95 code is on the HP 95. Add at least one RAM card, to quickly recover from crashes.

The HP 95 uses a V20 processor. You can think of the V20 as a real mode-only 80286. The V20 is plug-compatible with the 8088. For an investment of less than $20, the V20 not only improves the performance of your PC–XT, but it makes it possible to test code fragments on your PC.

HP 95 Development Tools

For serious development work, you will need professional languages and a powerful PC with a hard disk. Today’s compilers often need over 10 megabytes of storage.

For casual work (especially when System Manager compliance is not important), DEBUG and a copy of GWBASIC that will run on the 95 (from MS–DOS 3.2 or 3.3) are good starting points. But *real* programmers don’t use interpreted languages (at least not when anyone is watching), they want *real* compilers.

GWBASIC, which is furnished at no extra charge with most versions of MS–DOS before version 5.0 (but, as you’ve noticed, not HP’s version of DOS 3.22), runs mostly successfully on the HP 95. A few keywords may be incompatible. The BEEP and SOUND keywords, for instance, can make the speaker “stick” on (if you try this, you will have to reboot the computer to turn the speaker off). Test programs carefully, and save often.

We frequently use standard MS–DOS-based programming languages for writing larger HP 95 applications. The main considerations are efficiency, compatibility, and System Manager compliance. Bulky editors and over-sized runtime modules just won’t make it in the 95 world. The HP and Lotus programmers used Microsoft C ($495 list price) and Microsoft Macro Assembler (also known as MASM, $150) to create the HP 95. The latest Microsoft C optimizing compiler often makes code as good as or better than assembly (it’s more daring, it’ll optimize long functions, and it never gets lazy or takes the easy way out). Your compiler’s 80186 or 80286 switch will improve performance and efficiency on the HP 95.

The author used Microsoft MASM 6.0 to write the HP 95 *Utility Pack;* it was a good choice because of development speed (improved macros and structures), and of course nothing makes smaller programs than assembly language. The *Utility Pack* is made up of about 10,000 lines of assembly language source code (commonly used code is replaced by macros, which cuts source size and minimizes errors). Here's the only secret about writing in assembly language: It takes a lot of code to do anything, so the more information on the screen — as many as 50 or 60 lines — the easier it is to follow program logic.

Microsoft C and MASM are excellent tools, but they aren't designed for the HP 95. Languages and tools from other vendors have the same problem: they don't have "hooks" to the System Manager or other HP 95 features. Until HP 95 development tools and libraries are available, most products will be DOS applications adapted for the 40-column display.

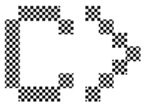
When development tools that take advantage of HP 95 built-in features are available, use them. This will make your applications smaller, friendlier to use, and at the same time faster and easier to write. Instead of reinventing wheels, using System Manager calls starts from a well-known prototype (instead, reinvent hubcaps?)

As of this writing, there are no programming languages or libraries written specially for the HP 95, though of course some are under development. The author is considering a second volume of this book focusing on programming, as well as a collection of HP 95 programming tools. Write for details or if you have suggestions.

You will also need books on MS–DOS and the Phoenix BIOS; both are available in larger book stores. MS–DOS is the most explored and best documented computer operating system ever written.

If you write straightforward programs without using undocumented hooks (BIOS calls are fine, though), keeping in mind the 40x16 display, most Microsoft C, QuickC, QuickBASIC, and Microsoft BASIC 7.1 programs, as well as Borland Turbo Pascal and C code, should run happily on the HP 95.

Exploring MS–DOS



The *DOS* in *MS–DOS* stands for Disk Operating System. Operating disk drives is what MS–DOS is all about. We communicate with DOS (that's the shorthand way of saying MS–DOS) by typing instructions on the command line. These instructions tell DOS to do things like create, copy or delete files, and to run programs.

Applications programs also depend on MS–DOS for disk management. Even when a program is running (including the HP 95 built-in applications), DOS is in the background, handling requests for files, memory and other system-wide services. MS–DOS handles the internal gubbins, leaving the interesting parts to application programs.

The *MS* stands for Microsoft, the author and distributor of the operating system. PC–DOS is just about the same thing, behind an IBM nameplate.

Abstract MS–DOS

The language of MS–DOS is full of words like paths, pipes, templates, streams and switches, and a collection of rather strange single-character commands. When we're done here, these symbols really will mean something:

```
/  +  *  .  :  ?  |  >  <  \  !  "  %
```

The DOS language is brief and easy to remember, but it's not English-like. Some people in the computer business take the word *language* too literally. They reason that if it's supposed to be a different language, it shouldn't look too much like English. Popular programming languages like C (pronounced "See") add an extra level of abstraction by representing lots of tasks by a single character; maybe we could call it low-rogrylphics (not to be confused with hieroglyphics).

For instance, in the C language the exclamation mark ("!") means not, while if you tack-on an equals sign it means not equals ("!="), and the star ("*") means different things depending on where it appears — and depending on how many stars there are (C is a four-star language). The new-age version of C, called C++ (pronounced "See plus plus"), adds even more abstraction, but missed the opportunity of straightening up the original C obscurities, and even adds more (like a new colon-colon :: keyword). Luckily, DOS isn't quite that abstract.

Stepping off the soap box… Now, since the programming community can't make it easy on themselves, there's no reason to expect them to make it easy on us users. We really shouldn't blame programmers: they don't get out very often, and a diet of pizza, cola and candy can have deleterious effects on people.

A Little Less Abstraction

Shorter commands save typing. And they are, once you've committed them to memory, easy to read. Another coined term we'll introduce is brain-clock-cycles. Languages are symbolic representations of things and ideas. The goal of DOS' short-cuts is to make symbols recognizable quickly, and without having to pay too much attention — cutting clock cycles.

You may be tempted to skip the section on redirection and piping, one of the most abstract aspects of MS–DOS (especially if you skipped the section on binary numbers). Please don't, because while you'll probably not need it often, it's the foundation and reasoning behind many of the heuristic-machinations we go through.

Exploring MS–DOS From the 95's Filer

Filer is a wonderful place to explore DOS' file chain. It displays a sorted list, and you can look at the contents of any file by moving the hi lighted selector to the file name and pressing **ENTER**. The most powerful Filer command, though, is Goto (the **F5** function key). Goto changes to a different directory, a disk drive, and can even view a file — with a twist: the directory or file can even be hidden.

Let's look at a corner of your 95 you may never have seen before: the hidden _SYS\ directory on drive C:. Press the **F5** key, then at the "Goto: File or directory name:" prompt, type:

```
\_SYS
```

The screen shows the directory name followed by a new list of files. You can confirm that there are about 27 files in this directory (at least, in the English language HP 95); press **MENU**, then select Directory Status. Filer will display information about the disk and directory:

```
C:\_sys
Subdirectories:           0
Files in directory:       27
Bytes in directory:       196479
Bytes remaining on disk:  122368
Volume label:
```

Now press **ESC** to return to the file list. The first file in the directory is 1000.PCX, which you will recall is the picture of a $1000 bill. Now if you press **ENTER** to view the file, you won't see president Cleveland, but a screen-full of smiling faces, upside-down T's, and random-ish characters. Press ↓ or ↑ to browse through the file; it doesn't get any more readable than this. Most of the files in this directory look just as garbled. These are *binary* files, containing information encoded in binary instead of ASCII text. Lotus 1–2–3 WK1 spreadsheets and program files are also binary, but you'll find text and messages here and there.

Files can also be hidden, just like the hidden C:_SYS\ directory. The HP 95 includes several hidden files in the C:\ root directory. Press **F5** again, and at the prompt type:

```
\ROMDUMMY.015
```

This large file is chock-full of the messages and prompts displayed by the HP 95's built-in applications. When you press **ESC** to return to the directory list, you're back in the root directory.

Notes About Filer

- Goto also changes disk drives. You can specify both a disk drive letter and a directory at the same time.
- Filer sorts files by name for display only. It does not sort the files on the disk.
- While Filer can look at hidden files and directories, it doesn't include any mechanism to hide or un-hide files.
- When you use Filer's Goto command to view a file, and specify a complete path with the file, not only will Filer show the file, but it will place you in that directory.
- Use Filer's RENAME (**F8** key) command instead of MOVE (**F10**) when the destination is on the same disk. MOVE copies then deletes the original, while RENAME just changes the directory entry.
- Selecting COMMAND.COM does the same thing as **MENU** System, and takes the same amount of memory.
- Filer's System command is the gateway to MS–DOS. When you use Filer's System command to go to the DOS prompt, you will be logged-onto the same directory you used in Filer.

- Don't leave Filer running when it is not needed. The program uses battery power to constantly poll the serial port. This can noticeably reduce battery life.
- Avoid loading TSR (Terminate and Stay Resident) programs from Filer. This reserves more memory than the programs should require and can make the computer unstable.

The MS–DOS Command Line

We can follow two similar courses to reach the MS–DOS prompt from the safe harbor of Filer: Press **MENU** then select System, or move the selector bar to COMMAND.COM in the file list then press the **F4** function key. Both methods leave the same amount of memory available for MS–DOS applications.

```
(Type exit to return to Filer)

Microsoft(R) MS-DOS(R)  Version 3.22
             (C)Copyright Microsoft Corp 1981-1989.

C>
```

Pretty austere, isn't it? While you're at the DOS prompt, none of the HP 95 built-in applications are available; pressing the **123** key does nothing at all. You will have to return to the safety of the System Manager and Filer to gain access to the built-ins. To get back to Filer in a hurry, type EXIT then press **ENTER**.

Note

Remember to close all applications except Filer when you shell-out to DOS. If an application is open, the computer will remind us — we really don't have any choice in the matter. This is just as well, since we can be assured that all data files are closed.

It's difficult to lose or corrupt data from closed files. Since all of the HP 95's built-in applications are closed when you exit to MS–DOS, there is little concern about losing data if a DOS program crashes — unless the crashed program corrupts the disk.

The MS–DOS Command Editor

The MS–DOS command line editor is a primitive contrivance. The editor remained virtually unchanged for the first ten years of DOS' existence (the command editor was finally updated it in 1991, when Microsoft introduced MS–DOS 5.0). If you find the editor balky and unfriendly, don't worry about it, millions of other people do too.

A thriving cottage industry offers enhancements or replacements for the DOS command line. The HP 95 Filer is a typical command line replacement: easy to use, but missing a few essentials. No matter how good the command line replacements may be, sometimes you need the flexibility of the MS–DOS command line; it's worth the time you'll spend getting to know it. Even if the built-in applications do everything you want from your HP 95 today, somewhere down the road you will still need to turn to the MS–DOS command line.

Note

The **MENU**, **123** and other special keys are unique to the HP 95. Since other computers don't have these keys, software is not designed to recognize them, so programs either will ignore the keys entirely, or the keys will be misinterpreted as something completely different (and likely not very useful).

DOS Command Template

One of the nicest features of the command line is the previous command you typed is available as a template you can use to type the next command. Many DOS users ignore the template and re-type each command; the small keyboard on the HP 95 encourages us to save keystrokes — that's the real strength of the template. Let's walk-through a few exercises using the template and the DIR command.

The DIR (short for directory) command lists information about files in a disk directory. The directory is similar to the table of contents in a book, listing chapters, and where you'll find them in the book. Every disk has a directory, the section of the disk that contains information

about files and their size and location on the disk. DOS' DIR command lets us look at the table of contents of a disk. Using it is as typing the name on the DOS command line:

```
DIR
```

If the last command line began with DIR and you'd like to see another directory listing, pressing either the **F1** template key or the → (right cursor) key four times will bring-back the word and the space after it. Each time you press **F1** or →, one character from the template is copied to the command line. Now, DOS will place the cursor at the end of the line so you can add the new file specifications.

```
DIR *.COM
```

If you mis-type a character, press ⇐ (Backspace) or ← (the left cursor key) to delete the typo. Remember that the cursor is always at the end of the line, so you'll have to erase everything, backing-up to the mistake, then retype the character.

When you press **ENTER**, MS–DOS will show a listing of all COM program files in the current directory. If you're in the root directory of the C: drive, DIR will always show at least two COM files: COMMAND.COM and TF.COM.

Suppose we were looking for CHKDSK, which we thought was a COM program. Apparently it's not, so we'll recall the command line, up to the dot, by pressing **F1** six times. Watch the display: after each key press, DOS will copy another character from the template to the command line:

```
DIR *.
```

The cursor is at the end of the line, just past the dot. Now we can type the EXE file name extension then press **ENTER** to see the directory entry for CHKDSK.EXE (and any other programs with the EXE file name extension).

```
DIR *.EXE
```

Keep in mind the difference between the command line and the template: the command line is where you type new commands, while the template is where the previous command is stashed. As you edit the command line, you can copy parts of the template into the new command. Here's a summary of the template keystrokes.

DOS Template Editing Keys

Keystroke	Purpose
F1 or →	Copy one character from template to the command line.
F2	Copy the template, up to the character you specify.
F3	Copy the remainder of the template.
F4	Delete from current position to the character specified.
F6 or **CTRL**+**Z**	Places an end of file marker (character 26) in the line.
⇐ or ←	Delete character before the cursor. Template is unchanged.
DEL	Delete a character in template at the cursor position.
INS	Insert characters at cursor position in the template.
ENTER	Accept the command line. Tells DOS to execute command.
ESC	Cancel the command line. Does not change the template.
ALT+**CURSOR**	**ALT** plus any cursor key (or **ALT**+**SHIFT** and a cursor key) scrolls the HP 95 window around the full 80x25 display. **ALT**+ cursor key tracking is available only on the HP 95.

We'll talk about MS–DOS wild cards in a few minutes on page 47, but DIR is a prime candidate for using wild cards, so here's a preview. The **F1** and **F3** keys make it easy to use the previous command as a starting point for new commands. Say instead of EXE files, you want to see a listing of EXM files (those are the special HP 95 Application Manager compliant programs). Press **F3** to recall the entire template:

```
DIR *.EXE
```

Now press ← or ⇐ to erase the E from the end of the line, then replace it with M. Finally, press **ENTER**.

```
Volume in drive C has no label
Directory of  C:\
File not found
```

What happened, we know there should be at least two EXM files? Remember, the HP 95 ROM includes System Manager versions of TigerFox and Hearts&Bones, which are in the hidden C:_SYS\ directory. We're logged-onto the root directory of the C: drive, so we'll have to tell DOS where to look for the files.

Note

Unlike the HP 95 Filer program, DOS' DIR command does not sort file names in directories. Subdirectory names will not be placed at the top of the list. MS–DOS writes files to the directory in the order they are created; as files are added or deleted, the order of files in the directory will change. You might try using a utility program (such as Norton Utilities) to sort the directory, and write the sorted list back to disk, but some utilities may have a fit when they try to re-order files that are permanently burned in the HP 95's ROM.

Let's modify the command line to add the path so DOS will be able to find our missing files. First press **F1** four times to recall the word DIR and the space. Now, press **SHIFT+DEL** (that's the equivalent of the **INS** key on the HP 95) to enter insert mode then add the directory to the command line:

```
DIR \_SYS\
```

When we move the cursor or recall anything from the template, the command line returns to overstrike mode. The rest of the command is still in the template, so we can press **F3** to recall the remainder of the line:

```
DIR \_SYS\*.EXM
```

Now when you press **ENTER**, DOS will find the files in the _SYS\ directory, and display their directory listings:

```
HB       EXM     6702   3-29-91   4:34p
TFOX     EXM     8315   3-18-91   1:54p
```

Instead of pressing **F1** several times to copy the beginning of the template to the command line, this time let's try **F2**. This command requires an argument: the first character on the line that you don't want. DOS doesn't display the character you type, so it's time for some visualization. To copy the template up to, but excluding the file name extension, press **F2** then **E** (the first character in the file name extension):

```
DIR \_SYS\*.
```

Now type the new extension. The on-line help files for the HP 95's built-in applications are in the _SYS\ directory, so type the HLP extension:

```
DIR \_SYS\*.HLP
```

Note

The **DEL** key works on the template, not the command line (that's what the ⇐ and ← keys are for). Since the cursor is always at the end of the command line, there would be nothing to delete there, anyway.

Here's the last example of reusing the template. This is an outlandish way to save a few keystrokes, but follow through anyway, some of these maneuvers are quite useful. We'll look at the setup files (file names ending with ENV) in the C:_DAT\ directory, so to recall up to the underscore in the template, we'll press **F2** then **S**:

```
DIR \_
```

Now we'll delete the next three characters from the template by pressing **DEL** three times to delete three characters from the template. The display hasn't changed, but the next characters in the template are "*.HLP". The default is over type mode, so we'll change to insert mode by pressing **SHIFT+DEL** (the insert mode key, again). Now type **D A T**:

```
DIR \_DAT
```

We still need the next backslash, star and dot from the template, so we'll press **F2** followed by **H** to recall up to, but not including, the extension:

```
DIR \_DAT\*.
```

Now we can add the extension (ENV), then press **ENTER** to see the matching files:

```
DIR \_DAT\*.ENV
```

That's it, probably the most roundabout way to save a few keystrokes. In real life we would probably press **F2** then **D**, then since the template is in overstrike mode anyway, we would just overwrite the SYS with DAT. The backslash character is a shifted keystroke on the HP 95, so we would probably press **F2** again and type the remainder of the command, just as we did before.

If you aren't in the root directory, prefixing a drive letter to the file specification doesn't mean the root directory — it just means the drive, without specifying any path, which means the current path. Say you're still in the C:_DAT\ directory, this next command would seem to show the root:

```
DIR C:*.COM
```

It actually shows the files in the current directory! This is because it's possible to be logged-onto other than the root directory of more than one drive. If you're logged-onto the A: RAM disk, the C:*.COM path will show the matching files in C:_DAT\. To be sure you get the root directory listing, you must specify the full path name; in this case, that means just adding a backslash:

```
DIR C:\*.COM
```

These exercises show the template as a powerful, yet goofy-looking tool. It's especially useful with the small keyboard and frequently shifted keystrokes of the HP 95. We copy the template to the command line, then we use the template editing keys to (usually) make the job easier.

Note

If you make a typing mistake and would like to start over again, yet still re-use the original template, press **ESC** to erase the command line. The command line will be cleared, but the template is intact. The template is versatile, but you're working in the dark. You can also use this feature if you've forgotten what's in the template. Press **F3** to recall the entire line to the display, then press **ESC** to cancel; the cursor will move to the next line, but DOS will leave a copy of the template on the line above.

Note

With most versions of MS–DOS, the template from the previous command line is usually available, even after executing large applications programs.

One feature unique to the HP 95 is the longevity of the template. The previous command line is always available, even after you reboot the computer — it just won't go away! In fact, when you first move from Filer to the DOS command line, you can press **F3** to recall the previous command (which will probably be EXIT, the command you typed to return to Filer).

Almost miraculously, the HP 95 will recall the previous command line after a cold boot — even if you've re-initialized the RAM disk (though not after you've enlarged the RAM disk, and reduced conventional memory). Just about the only way to get rid of the old command (short of entering a new one, but that's too easy) is to unplug the AC adapter, remove the batteries and let the computer sit for a few minutes until it's completely forgotten its purpose in life (and the contents of all of memory, and the C: RAM disk).

Notes About The Template

- You can use the **F2** key several times while editing the same template, recalling a few characters at a time, then inserting or deleting others.

- Another interesting use for the template **F2** function is to recall just the command name without anything that follows. Since a space separates each command and whatever arguments it requires, pressing **F2** then the space bar tells DOS to copy the command name from the template, up to the space character, then place the cursor on the right.
- If you start to reuse the template then change your mind, press the **Esc** key. This clears just the command line; the template is still there, unchanged, so you can reuse it as if nothing had happened.
- Relax, you can forget about the template and all those editing options; they aren't mandatory. Most of the time we just type commands as we need them, and ignore the rituals and eccentricities.

MS–DOS Commands

These commands are the language MS–DOS understands.

MS–DOS recognizes two basic types of commands: internal, built-in commands (such as CLS or EXIT, or as we've already discussed, DIR), and everything else (external commands like CHKDSK.EXE or user programs like TF.COM). *External command* is just another way of saying utility program. External DOS commands in program files are treated exactly like any other program. You can circumvent the confusion of what to call things by calling everything a *command,* or the more generic term: *keyword.*

If the MS–DOS command processor recognizes the keyword as an internal command, it carries out the request immediately. However if it isn't a command DOS recognizes, the command processor assumes it must be a program, so it searches the disk for a program file with that name to run.

The MS–DOS command processor recognizes three program file types for external commands: COM, EXE and BAT (short for command, executable and batch). MS–DOS searches for program file types in just that order; if you have two programs with the same name, but one has the COM file name extension and the other is an EXE, DOS will run the program with the COM extension. Say you create a batch file named TF.BAT, when you type the TF command — even if you include the BAT extension — DOS will run the version of TigerFox in the TF.COM program file anyway.

Note

If you have experience with HP BASIC, you're used to running programs with a CALL or RUN statement. Since everything in DOS that isn't built-in is a program we call, preceding the command with CALL is unnecessary. There is a wonderful simplicity in DOS commands; no extra quotation marks or odd-looking delimiter characters, just a shorthand language. Many DOS commands have even shorter, abbreviated forms (like typing CD instead of CHDIR).

See also

The HP 95's MS–DOS external commands are documented in this book, beginning on page 86. The table on page 109 lists MS–DOS internal commands.

Arguments and Switches

Arguments are options we type on the command line following the keyword. These options modify how commands and programs work. MS–DOS passes the tail-end of the command line (called the *Command tail)* to programs without change. Each program recognizes different options, and some programs use none at all.

Switches are a type of argument, usually beginning with a forward slash ("/"), followed by one or more other characters. Additional arguments, like file names, are simply preceded by a space. MS–DOS interprets only redirection symbols, everything else is sent unchanged to the program or command — the commands interpret these arguments.

As we've seen, the DIR command displays information about files. We can enter just the command alone on the line to see a simple directory listing:

```
DIR
```

MS–DOS will interpret this as meaning “Show me every file in the current directory.” Here’s a listing from the root directory of the HP 95’s C: RAM disk:

```
 Volume in drive C has no label
 Directory of  C:\

COMMAND  COM      184   2-11-91  10:36a
CHKDSK   EXE     9680   2-11-91  10:37a
_CFLOW   WK1     4418   3-20-91   7:38a
_EXPENSE WK1    12406   3-20-91   7:38a
_STAT    WK1     5444   3-20-91   7:38a
TF       COM     8170   3-20-91   7:38a
_HOMEBUY WK1     4362   3-20-91   7:38a
_CARLOAN WK1     5232   3-20-91   7:38a
_DAT         <DIR>      4-01-91
123      CNF      376   9-04-92   2:47p
95       COM     2015   7-02-91  12:02p
AUTOEXEC BAT       71   8-10-92   1:23a
BATTERY  COM      128   3-23-92   9:29p
CONFIG   SYS       46   8-10-92   1:02a
       19 File(s)    156672 bytes free
```

What if we want to see only a few files, or maybe another directory? Or more likely, the listing is so long that much of it scrolls off the screen before we can find what we’re looking for. We can add a command line switch or a file name, or even both.

As we said, a special switch character often differentiates command line switches from file names. By conventions set by MS–DOS, switches begin with the slash (“/”), but a few programs require a minus sign (“-”) instead. For instance, the /W switch tells the DIR command to do a multi-column listing of file names without file sizes or date stamps:

```
DIR /W
```

This form of the DIR command tells MS–DOS to display a wide directory listing. On desk bound PCs, the listing is formatted for 80-columns (or as much as will fit within the display width); DOS 3.22 on our HP 95 formats the list for 40 columns:

```
COMMAND  COM  CHKDSK   EXE  _CFLOW   WK1
_EXPENSE WK1  _STAT    WK1  TF       COM
_HOMEBUY WK1  _CARLOAN WK1  _DAT
123      CNF  AUTOEXEC BAT  BATTERY  COM
CONFIG   SYS
```

Since the DIR command assumes that we want to see every file, we frequently enter the command without the *.* path specification (as we just did). This is the same as:

```
DIR /W *.*
```

If the file list is still too long for the 16-line display, DIR also recognizes the /P switch to tell it to pause at each screen-full of files. You can use several switches together, though most programs require that a space separates each switch:

```
DIR /W /P *.*
```

Note

Don’t confuse the forward slash “/” switch character with the back slash “\” used for directory paths.

Wild Cards and Directory Symbols

Say you want to see a list of just HP 95 Memo program TXT files, or a list of only the Lotus spreadsheet files that begin with “JAN.” That’s the job of MS–DOS wild cards. DOS’ wild card characters are place holders, representing zero or more unknown characters. When you specify files using wild cards, MS–DOS will repeat the command for each file that matches the patterns.

The two wild card characters are the question mark ("?") which represents any single character, and the star ("*") representing any number of unknown characters — or none at all. Here is a summary of the most frequently used DOS symbols and combinations:

DOS Shortcut Symbols

Symbol	Meaning
..	The previous directory up the tree.
.	All files in the current directory.
*.	All files without an extension.
*	Usually means *.*.
?	Any single character.
.	The current directory.
\	The root directory of the current drive.
/	Switch prefix.
-	Switch prefix (less common).

Notice that this list combines wild cards and directory shortcuts. This is because one can often be substituted for the other.

In DOS, we most frequently use wild cards with COPY, DEL and DIR commands; our examples will use DIR. DIR is always non-destructive, you can experiment with various ways to apply wild cards using DIR without the possibility of damaging or erasing files. These examples should encourage you to follow a consistent file naming strategy, which will make it easier to locate and identify files.

Note

You can use DIR to test wild card names before committing files to history with DEL. Try the file specification first with DIR and watch to make sure the directory listing matches only the files. When you're satisfied with the file listing, type DEL then press **F3** to recall the file name from DOS' command template.

The question mark wild card represents any single character at one place in a word. DIR recognizes P?D as PAD, POD, but it could never stand PAT — it requires an exact match for every character position except where the question mark appears.

```
DIR P?D
```

The question mark is again useful when we are looking for several similarly named files, such as a series of daily reports. If you name your spreadsheet files using the date as a number beginning with the year, such as 920804.WK1 for August 4, 1992, you could use wild cards to search for just the August files:

```
DIR 9208??.WK1
```

When several characters may match, such as they might above, we often use the star ("*"), which matches zero or more characters. We could rewrite the last example as:

```
DIR 9208*.WK1
```

This says to DOS: "Show me all spreadsheet files with names beginning with 9208, and followed by any other characters." However, since the star is a place marker for any number of characters, files found could include 92080000.WK1, and even 9208.WK1 — as many characters as are allowed to follow, or none.

The star is all-encompassing, every character following the star is ignored. If we're looking for files from any year, but only August, we might be tempted to specify something like:

```
DIR *08*.WK1
```

What we would get, though, is a list of all WK1 files. Keep in mind that while we can use any number of question marks, everything following a star — up to the dot for the file extension — will be ignored. The star always takes precedence over the question mark.

If the star is too flexible, try using several question marks. When a question mark is the last character in the name, it will match a single character, or none. Continuing with the same naming convention, his example will find the files from any year and any day, for a total of up to six characters, but it will also find ??08.WK1 files:

```
DIR ??08??.WK1
```

Both MS–DOS wild cards also work with file name extensions, following the same rules and inheriting the same quirks. Star will look for files with any extension, and the question mark represents any single character.

```
DIR *.EX?
```

We could say "*.*" with wild cards, a shortcut meaning any file with any extension in the current directory. So to look at all of the files in the current directory we would usually type:

```
DIR *.*
```

The shortest-cut of all is just a dot (“.”). A single dot is a synonym for the current directory name, without specifying any files. DOS will fill-in the blanks, implicitly adding the "*.*" string, to interpret this shorter version just the same:

```
DIR .
```

DOS is a Disk Operating System — there are lots more disk commands like DIR. Here's a look at interesting uses for some of these commands.

Some More File Commands

Command	Notes
`CD _SYS`	Change to a different directory.
`CD ..`	Change to the next directory up in the tree.
`CHDIR _SYS`	Synonym for CD.
`COPY a b`	Copy a file from one place to another. You can specify a path only or a path and new file name for the destination.
`DEL _SYS`	When used with a directory name, deletes all files in the directory. With a file name, it deletes all matching files.
`DEL .`	Works like DEL *.*
`ERASE _SYS`	Synonym for DEL.
`RD _SYS`	Short for "Remove Directory." Deletes empty directories
`REN a b`	Rename a file or group of files. Cannot be used to rename directories.

Note

If you've used programming language like HP BASIC, DOS commands may seem strange looking. There are no strings in MS–DOS, so it isn't necessary to put file names in quotation marks. Second, there is no *TO* keyword in the COPY command; just type two file names separated by a space.

Two useful internal commands are TIME and DATE. The HP 95 Setup program has menu entries for setting the time and date, but many people prefer to set the time in DOS, since the format is simpler and you can leave out the seconds or add everything down to the hundredth for absolute accuracy (or we can pretend that it's absolutely accurate). Simply type the command name at the DOS prompt:

```
TIME
```

DOS will show the current time and ask you to enter the new time:

```
Current time is  14:02:28.00
Enter new time:
```

You can type the time in 24-hour format, separating the fields as in the example. When you press **ENTER**, DOS will set the internal clock. If you change your mind and decide not to set the time, press **CTRL**+**C**. The DATE command works exactly the same.

Caution

DOS commands are so short and simple that it's just as easy to do damage as good. Be careful about COPY and DEL, especially with wild cards. Once a file is deleted or overwritten, it's gone forever.

In Filer, the Goto command makes changing to another directory or disk drive, or even viewing a file a simple fill-in-the-blanks affair. DOS is just as easy, but there are a few things to keep in mind, as we've seen. TYPE replaces the file viewer, CD changes directories, and to change to a different disk drive we just type the drive letter:

```
C:
```

Note

If you try to change to a bad drive letter (say, A:, when there isn't a card), Lotus 1–2–3 won't stop you. From then on, you are stuck with the bad letter and must exit from the program to straighten things up.

DWIM: Do What I Mean

DOS commands like DIR expect all sorts of vague path and file names. DOS will put up with a lot of typing abuse. The process is known as *Do what I mean,* or *DWIM* for short. If you leave out a name or a few characters, MS–DOS will still try to make sense of it. You can enter commands and path names in several strange ways, and commands — actually DOS' command processor — will figure out what you mean.

The file specification in the command above could also be written as ".*.*" because DOS automatically fills-in the missing file name. In fact, we can take further advantage of the intelligence of DIR by not specifying anything:

```
DIR
```

DOS will assume we mean the current directory, so it will fill-in the missing star-dot-star. Two dots means the next directory out of the tree. We can see a list of all of those files with "..*.*" or even the shortcut version, and let DOS fill-in the missing file name:

```
DIR ..
```

Internally, MS–DOS always expands file name. It converts every character to upper case, then it replaces stars with question marks until the file name fits into this template:

```
????????.???
```

After expanding the name, DOS makes the comparison by expanding disk file names into the same template. If you specify "*.TXT" MS–DOS will expand the template to:

```
????????.TXT
```

The disk file, perhaps README.TXT, is coerced into:

```
README??.TXT
```

Following the question mark wild card rule, we have a winner. These file names match. When DOS displays the file name, it replaces the question marks with spaces.

Notes About Wild Cards

- When the question mark wild card appears at the end of a word, it can represent any single character, or no character. COMMAND?.COM will find the file COMMAND.COM. However, when the question mark is at the beginning or within a word, a character is required at that position; COM?MAND.COM would not find COMMAND.COM.
- You can use wild cards only with file names, and to find paths, not to search wild card paths for files. For instance, DIR ?DAT will find the _DAT\ directory, but DIR \?DAT\ will not find the files in the _DAT\ directory (DOS will return the "Invalid directory" error).
- You can often abbreviate "*.*" as just "." (the dot). The dot represents the current directory, and therefore every file in the current directory, while "*.*" represents any file name with any extension — which is really the same thing.

- Programs aren't always consistent about handling of wild card characters. Many programs (such as text editors) don't allow wild cards at all.
- Many programs accept wild cards file names; this often will tell the program to load a list of matching files. Since the wild card mechanism is built-into MS–DOS and is available to programs, most programs simply use the DOS services without change, so it's well worth the bother to familiarize yourself with how wild cards work.
- DIR really is flexible: it will accept a file name without an extension, then assume the ".*" extension. If you want to see only files without an extension, you must add the dot after the file name. DEL, on the other hand, requires the properly specified file extension — which is actually to our advantage, adding a bit of extra security to this potentially destructive command. "DIR X*" will show all files with or without an extension, while the DEL command recognizes "X*" only with files having no extension.
- Most programs ignore any extra non-blank characters following the three-character file extension. If a command asks for two file names (such as REN, which asks for the original and new file name) and you leave out the space between them, DOS will truncate everything beyond the three-character file extension, including the second file. Since DOS will miss the second file name, the error message it returns may not fit the crime.

The Trouble With Arguments

In its infancy in the early '80s, MS–DOS didn't recognize disk directories, or for that matter any kind of storage beyond floppy disks. The forward slash, commonly used (then and now) by other operating systems to delimit directory names, wasn't needed. So DOS programs used it for the command line switch character (even though the minus "-" was already in common use at the time). When MS–DOS matured, lo and behold the forward slash was already taken, so the backslash became the directory name delimiter.

Today, MS–DOS internal commands and utility programs follow the slash-convention. Some programs — throwbacks or programs imported from the UNIX operating system — still require the minus. To further confuse things, DOS allows the minus in file names. In attempts to appease everybody, some programs recognize both the slash and minus.

Note

We usually enter switches on the line before file names and path specifications. Often this is just a convenience, but many programs will ignore commands that follow file names or — worse yet — they will interpret them incorrectly and show error messages and warnings or refuse to work.

Programming utilities usually require the switches before the file specification. DOS external commands aren't as consistent. For instance CHKDSK can accept the file name anywhere, while MIRROR, a command from MS–DOS 5.0 and later, requires the disk drive letter before any switches, otherwise the drive letter will be ignored.

When the command or program expects two file names, remember that the source file comes first, followed by the destination file.

Here are a few hints for avoiding the trouble with arguments:

Notes About Arguments

- Many programs require a space to separate each argument. Avoid crunching-together several switches.
- Avoid beginning file names with the minus sign. Some programs don't require spaces between arguments, so it's best to avoid the minus anywhere in file names.
- Instead of using a minus sign to separate two-word file names, try the underscore character. On the HP 95, press **SHIFT** + = (the equals key) to type the underscore.
- When in doubt, type commands with all other arguments preceding file or path names.
- If all else fails, maybe it's in the manual.

Pausing and Canceling Commands

When directory listings go scrolling off the top of the screen, you can always execute the command again and hope you see the file you're looking for this time. As we've seen, there's another option: the /W switch, which shows a multi-column listing of just file names. There's yet another solution if the list is still too long, or if you want to see file sizes and date stamps (and for programs and commands that don't recognize the /W stitch.

Three special keystrokes give us more control over how MS–DOS commands work:

Keystroke	Purpose
CTRL+S	Suspends program display until you press another key.
CTRL+C	Cancels the program, returns to MS–DOS.
CTRL+⇐	Breaks-out-of the program, returns to MS–DOS.

When you've found the information you're looking for, you can press **CTRL+C** or **CTRL+⇐** to quit the program. If the program is displaying lots of useful information, more than will fit on the display, you can press **CTRL+S** to suspend it at each screen-full; the next key you press will continue the program.

Note

While you can follow **CTRL+S** with any other keystroke to continue the program, try to get in the habit of pressing **CTRL+S** again to continue. This will make sure that extra, unwanted keystrokes won't be sent to your program or to MS–DOS.

See also

You can tell MS–DOS to check for the break key more often with the BREAK statement in CONFIG.SYS. Page 67 explains how this command works.

CTRL+S and **CTRL+C** keystrokes will only work with programs that use MS–DOS for keyboard and display services. This excludes most large, sophisticated programs. Smaller utilities — like DOS' built-in commands and utility programs such as *HP 95 Utility Pack* — usually will support these keystrokes.

One way to spot programs that do not support these services is by observing how the programs work. Programs that recognize unusual keystroke combinations (such as **SHIFT+ENTER** or **CTRL+SPACEBAR**) or do formatted, full-screen display, or run in graphics mode, will usually ignore the pause and break keys.

Caution

These keystroke services are provided by MS–DOS, not by the programs. DOS does not inform programs that they have been paused, or that they are about to be involuntarily terminated.

CTRL+C or **CTRL+⇐** will stop many programs in their tracks. If they are in the middle of performing important tasks (such as copying several files), those tasks will not be completed. Data will not likely be lost because MS–DOS automatically closes all open files when programs exit.

See also

Text that you can pause or stop is often called "Stream Output." We'll talk about the stream again when we discuss redirection and piping on page 57.

Here a few things to keep in mind about stopping-up the I/O stream:

- On desk bound PCs, pressing **CTRL+NUMLOCK** will pause most programs even if there is a key in the keyboard buffer. The HP 95 does not have a pause key.
- If another key is in the keyboard buffer, pressing **CTRL+S** or **CTRL+C** won't do anything. This means if you press a key that isn't important to DOS while it's displaying text, DOS will ignore the key, but it will still be placed in the keyboard buffer. When you do press a key DOS should recognize, it won't see the key because the key will be placed in the buffer following any keys already there, and DOS only looks at the first key.
- With most versions of MS–DOS, you can use the MORE filter with redirection to pause the display at each screen-full. MS–DOS 3.22 on the HP 95 does not include MORE, so you will have to use **CTRL+S** or a third-party MORE-like program (such as TMORE, included in the *HP 95 Utility Pack)*.

- You can also use **CTRL**+**C** to quit batch files (.BAT extension). MS–DOS will ask for confirmation, then if you press **Y** the program will be terminated.
- To stop a process that's gone out of control, press **CTRL**+⇐ (that's the same as **CTRL**+**BREAK** on your desk bound PC).

Missing Commands

The HP 95 has very few external MS–DOS commands. FORMAT, which is an external command in most versions of DOS, is internal on the 95. External command files include CHKDSK.EXE, COMMAND.COM, DEBUG.EXE and SERINT.COM.

ROM-based MS–DOS 3.22 on the HP 95 is missing some commands, and a few commands are different from DOS conventions. FORMAT, for example, can format only RAM cards; therefore it doesn't need any command line switches for disk size.

The most noteworthy absentees are ANSI.SYS, ATTRIB, BACKUP, COMP, DISKCOPY, EDLIN, FC, GRAPHICS, LABEL, GWBASIC, MODE, MORE, RECOVER, RESTORE, SORT, SYS and XCOPY. It's either a glaring oversight or commendable frankness. Surely the latter. The HP 95 gets along quite nicely without most of these programs.

We can be thankful that Hewlett-Packard left-out EDLIN.EXE, the primitive text editor included (and ignored by users of…) each DOS version from 1.0 through 5.0. Along with BACKUP, RECOVER and RESTORE, it's the most justifiably maligned tool in DOS' arsenal.

The SYS command, which copies the start-up files to a boot disk, is not needed by the HP 95, because the start-up files are always available in ROM. Some other utilities, like XCOPY and MODE, aren't needed because of the RAM disks or limited number of display options.

Files and Devices

Most file manipulation is done in programs — Lotus 1–2–3 creates spreadsheets, Memo edits text files, even HP Calc uses data files for Solver. MS–DOS commands create directories, make back-up copies of files, and prune the overgrown directory tree, but we rarely create files directly in DOS.

File Names

There is nothing to stop you from creating text files in the HP 95's Memo program with COM or EXE extensions. And there's no built-in protection to stop the computer from trying to run a non-program file with a program's name.

See also

If you're new to MS–DOS and the file chain, you may want to start with the introduction to files and directories on page 10.

MS–DOS File Name Characters

Symbol	Meaning
A-Z	Letters.
0-9	Numbers.
_	Underscore.
^	Caret.
$	Dollar sign.
~	Tilde.
!	Exclamation Point.

Symbol	Meaning
#	Number sign.
%	Percent sign.*
&	Ampersand.
-	Hyphen.*
{ }	Curly braces.
( )	Parentheses.*
@	At sign.*

Symbols in this table marked with "*" should be avoided at the beginning of file names because of conflicts with MS–DOS commands and programs.

See also

In addition to the letters A-Z, you can also use the high-order characters from the HP 95's character set. These are characters with codes above 127. The table on page 107 lists the HP 95 character set and keystrokes for entering special codes.

Note

If you're familiar with HP computers like the 70- and 80-Series, you probably expect file names to begin with a letter, and included only letters and numbers in the names. Also, HP file name extensions are *hard coded;* the operating system protects us from giving a file the incorrect file type extension. As we've seen, file names are more flexible under MS–DOS. The HP 95 does not have any file type restrictions or protection — we are responsible to make sure that files have the correct extensions.

See also

The table on page 112 lists common HP 95 and MS–DOS file name extensions.

Notes About File Names

- MS–DOS is not case sensitive. You can enter file names and paths in upper or lower case, or a combination. DOS will convert the names to upper case.
- File names cannot contain spaces or other blank-like characters.
- File names can be up to 8-characters plus an optional 3-character file name extension.
- File name extensions are not required, but files cannot have just an extension without a file name.
- Avoid creating file names containing the hyphen ("–") character. This can conflict with command switches.
- File name extensions are used to group similar file types. Programs, for instance, are always BAT, COM or EXE, and each of those types represents a different program file format. Lotus 1–2–3 uses WK1, and the HP 95 Memo program prefers the TXT extension. It's a good idea to adhere to the standards. Following these conventions will make it easier to identify files later on down the road.
- Use descriptive file names, which describe the contents of the file. This will save time later when you're searching for a long-forgotten file.
- Use peculiar extensions and names to help protect names. People will be less likely to snoop at files named BASEBALL.CAP, CORN.COB or LETTER2.MOM.
- MS–DOS reserves several names for devices. Do not use the following as file names: AUX, CLOCK$, COM1, COM2, COM3, COM4, CON, LPT1, LPT2, LPT3, NUL, PRN.
- Directories follow the same naming conventions as files. Directory names are allowed to have extensions, just like files.
- You are responsible to make sure file names and especially extensions follow MS–DOS conventions.

Identifying Program Files

MS–DOS takes it on good faith that files named COM or EXE are valid programs. Beyond checking for the EXE file header, there is nothing to identify a program from any other type of file. Nothing good can come of fooling MS–DOS into thinking a file is a program when it isn't. When you're not sure if it's a real program file, there are some ways to check.

Batch files, ending with BAT, are text files containing commands, much as we would enter them from the keyboard, in English-like form. You can list or even edit batch files with the HP 95 Memo program. All DOS commands are allowed in batch files, including a few new commands (which turn batch into a useful language) that we'll talk about later.

Both COM and EXE files are compiled programs, written in a machine-readable language (called, of course, machine language). These files are created from instructions written in programming languages, which are translated to machine language by other programs called compilers.

The COM file type is the oldest MS–DOS program file format. It's an exact image of how the program will look when DOS loads it into memory. COM files are always less than 64K of memory, which is the size of a single code segment. There is nothing in the file to identify it

as a valid program to MS–DOS. It takes it on good faith that any program with the COM extension is a valid program (keep this in mind when naming text files).

Note

COM programs frequently begin with a JMP instruction, which is encoded as either E9 hex (a long JMP) or EB hex (a short JMP). You can look at these files with DEBUG, and check for either of these values. The codes translate to 233 and 235 decimal, which are displayed as "ù" and "ú" if you look at the files with the HP 95 Filer program. This is common practice, but it is not a rule; many COM programs do not begin with these characters.

COM program examples in this book, for instance, rarely begin with a JMP instruction. If the program doesn't begin with a JMP instruction, you may still be able to identify it. Try the DEBUG U (Unassemble) command; if it displays a series of valid instructions (not a series of DB or ??? lines), it may still be a program file.

The EXE file is more complex, it can contain multiple segments and various other information. EXE files always begin with a header of at least 512 bytes, so small utilities — especially for the HP 95 — are usually written as COM programs. You can identify EXE files by looking at them with Filer: if the first two bytes are "MZ" and just about everything else is unreadable, it's probably an EXE file. The "MZ" isn't program code at all, it's the initials of Mark Zbikowski, one of the developers of MS–DOS. It's not there out of vanity (well, maybe a little bit), but to help identify the file as a valid program.

Caution

If the program extension is EXE but it doesn't begin with the "MZ" signature, MS–DOS won't automatically reject it as an invalid program file. Instead, it will assume that it is a COM program and DOS will try to execute the file anyway.

Devices and Device Drivers

Considering that it's manual-shift, MS–DOS has an automated way of looking at computers. The goal is device independence, so programs can run on any MS–DOS-compatible computer, regardless of how dissimilar the underlying hardware may prove to be. Device drivers are programs — usually part of the BIOS, called resident devices — that handle hardware-dependent tasks like video, keyboard and disk drives. Drivers are conduits for programs to communicate with resident devices.

DOS is an expanding universe. The original, idealized MS–DOS world had just these few devices, but it on the chance that. Millions of computers and billions in development investments later, other installable device drivers control mice (the pointing kind), touch screen panels and infrared ports. These drivers could be supplied by third-parties.

MS–DOS automatically installs several devices when you boot the computer:

MS–DOS Devices

Device	Purpose
COM1	The serial and infrared ports.
CON	The console (display and keyboard).
LPT1	Line printer, usually a parallel port. Same as COM1 on the HP 95.
NUL	The null device (everything is discarded).
PRN	The printer (usually a parallel port, though on the HP 95, the serial port).

In everyday use, we treat devices like files. If you want to print a file, you could send it to the *PRN* device. This will not create a file named PRN, but it instead passes the information to the device driver with that name:

```
COPY CONFIG.SYS PRN
```

The *CON* device represents the console, both the display and keyboard — it can both receive and send data. We call the keyboard *Standard Input,* or *StdIn* for short. The display is *Standard Output,* also known as *StdOut.* Together they are the CON device. These distinctions, and the

console device driver, will seem more important and even useful in a few minutes, when we talk about redirection. Using the CON device, we can use COPY like TYPE:

```
COPY CONFIG.SYS CON
```

This sends the file to the console, and displays "1 File(s) copied" when it's done. The other side of the console is StdIn; we can create short text files from the keyboard without running any text editor:

```
COPY CON HELLO.TXT
```

Now MS–DOS will record your keystrokes in a file named HELLO.TXT. Every time you press **ENTER**, DOS adds another line to the file. To stop recording text, press **F6** or **CTRL**+**Z** to enter the end of file character. DOS will again display "1 File(s) copied."

The *NUL* device is a *bit bucket;* whatever you throw at it is happily accepted, then just as cheerfully thrown away. Here, the file is copied to the NUL device:

```
COPY COMMAND.COM NUL
```

The file is copied nowhere, but DOS will also display "1 File(s) copied" when it's done. You can eliminate that message, too, so the computer will do quite a bit of busy work, but the result will be nothing at all being done. Here's a hint about redirection: instead of the console, the message will be redirected to the null device.

```
COPY COMMAND.COM  NUL > NUL
```

Here's a trick to keep snoops from bothering your computer while you're away:

```
COPY CON NUL
```

DOS looks just fine, accepting keystrokes and displaying text with abandon. MS–DOS will display no error messages, but nothing worthwhile will happen, either. There are three ways get out of this predicament: **CTRL**+**C**, **CTRL**+⇐, or **CTRL**+**Z** then **ENTER**.

See also

Installing device drivers with CONFIG.SYS is discussed on page 67. Page 83 compares device drivers and TSRs (Terminate and Stay Resident programs).

Caution

MS–DOS includes a few other standard devices, and some software may add new devices. Unless you're sure what the results of using these devices will be, avoid experimenting. Two MS–DOS devices to avoid are CLOCK$ and AUX.

Taming the RAM Disk

Most of our working files end-up in the C:_DAT\ directory. Frankly, lots of files we really don't care about land in C:_DAT\. You can clean-up some of that clutter by deleting unneeded DCF, CTF and PBK files, and even TOPCARD.PCX, that the computer created the first time you booted. You'll find spare copies in the hidden C:_SYS\ directory. See page 93 for how to use SYSINIT.BAT to restore these files.

See also

The MS–DOS Command Table on page 109 lists all MS–DOS commands. This table compares using Filer and the MS–DOS command line to perform similar tasks:

MS–DOS File Commands

If You Want To	Filer	DOS
Copy a file to a card	**F2**	COPY
Rename a file	**F8**	REN
Delete a file	**F3**	DEL
Copy contents of a directory	**F9** to select files, then **F2**	COPY *.*
Create a directory	**MENU** Directory Create	MD
Delete a directory	**F3**	RD
Rename a directory	**F8**	.

The Filer Rename (**F8**) function can also rename directories. There is no MS–DOS command to rename directories.

The HP 95 has several permanent files in the C:\ root directory. You can look at them, but you can't change or delete these files. Now say you're an old-hand at Lotus 1–2–3 and you'd like to recover the space taken by the sample WK1 files in the root directory… Well, you can't, but don't be concerned about the wasted space because while these files are part of the C: RAM disk, they are in the ROM portion of the drive. The same is true for COMMAND.COM, TF.COM, CHKDSK.EXE and all the files in the C:_SYS\ directory.

The Filer can only recognize the first 100 files in any directory. Since DOS allocates subdirectory entries on an as-needed basis, reusing deleted directory entries, some new files may be excluded from the Filer list when it exceeds 100 files, even though alphabetically they come before the last entry that made it into the listing.

MS–DOS stores files as units of 512 bytes; 512, 1024, 1536, and so forth. Each file we create, regardless of how small, occupies at least 512 bytes of disk space. Whenever a file grows past a 512 byte boundary, 512 more bytes are allocated to the file.

Other computers and other disk types will use different file storage standards. One of DOS' strengths is its ability to hide these differences from applications programs. From the program's perspective, it doesn't matter if files take 512 bytes or 2K (a common minimum size for hard disk drives).

Notes About Disk Taming

- Resist the temptation to stash every program utility in the root directory and every data file in C:_DAT\. A scheme of subdirectories will make organization nearly automatic.
- Keep the root directory clean. The number of files we can create in root directory of all disk drives is limited, and the cleaner we keep the root directory, the easier it will be to navigate through the disk.
- Create working directories for data and programs. Many DOS fans expect a program directory named \BIN\ or \DOS\ or \UTIL\.
- A PATH environment variable will make it easier to hide utilities and still be able to execute the programs.
- While subdirectories can grow as we add more files, keep them under 100 files, because Filer can only display the first 100 files in any directory.
- Before deleting a directory, you must first delete the files in the directory.
- Filer sorts the directory for display. DOS reuses directory entries as files are created, deleted and renamed, so files on the disk are rarely sorted, and their order in the directory listing can change.
- MS–DOS stores files on RAM disk in multiples of 512 bytes. As files grow, the disk space they require is allocated as additional multiples of 512 bytes.
- The HP 95 C: RAM disk allows a maximum of 100 files in the root directory.

Redirection and Piping

The device-independence of MS–DOS makes it easy to treat printers and files and displays as much the same thing. The handful of redirection and piping commands adds flexibility to device-independence.

Redirection is usually used with command line utilities and built-in DOS commands like TYPE and COPY.

When a utility asks DOS for input it reads from the console, which is usually the StdIn device. Output is then sent back to the output-half of the console device. The trick of redirection is separating the tasks of input and output from the console, in effect deceiving programs about

where information is coming from (or going to). With really complex, convoluted redirection tricks, programs won't whether the data is coming or going. And, MS–DOS is a party to this scheme. O' what tangled webs we weave…

Redirection and Piping Symbols

Symbol	Purpose
>	Redirects the output to a file or device. If the output is going to a file and the file already exists, the original file is replaced by the new file.
>>	Appends the redirected output to a file or device. If the output is a file and the file already exists, the information is appended to the end of the file.
<	Use this symbol to read data from a file instead of entering it from the keyboard.
|	This is the piping symbol. Use it to redirect output from one program to StdIn of another program. The second program is usually a filter or search program.

These symbols are easier to remember if you visualize them as arrows. The > symbol points data at a device (usually a file). The < symbol points data to a program (usually from a file). The pipe symbol (|) can be interpreted literally: it acts as a conduit, moving information from one device to another.

DOS Console Devices

Name	Device
StdIn	The standard input device (usually the keyboard).
StdOut	The standard output device (the display).
StdErr	The standard error device (also the display).

The > Output Redirection Symbol

We often use COPY CON to create small batch files. For single-line files, we can do it even easier. The following example will create the file APNAME.LST, with a single command to assign TigerFox to a key:

```
ECHO C:\_SYS\TFOX.EXM,1400,TigerFox > C:\_DAT\APNAME.LST
```

This says to DOS: "Instead of displaying the text, create a file named APNAME.LST, and send the text there."

We often use output redirection to copy a directory listing to a file or the printer using DOS' DIR command. This example sends a directory listing to the printer:

```
DIR >PRN
```

If a program or command requires command line arguments, include them on the line before the redirection symbol:

```
DIR *.WK* > 123FILES.TXT
```

This says to DOS: "Show me all the Lotus 1–2–3 spreadsheet files, and send the list to the file named 123FILES.TXT."

The >> Output Redirection Symbol

So far, everything we've redirected has the same potential weakness: it could overwrite the contents of an existing file. The DOS designers thought this through thoroughly, adding another symbol, >>, to append the output to an existing file. If the file doesn't exist, >> works just like > to create the file, otherwise the new information is tacked-onto the end of the file.

Using ECHO with output direction is fine for one-line files, but if we later change our mind and want to also include Hearts&Bones, we could start over, or use the append symbol (">>") to add the second line to the file:

```
ECHO C:\_SYS\TFOX.EXM,1400,TigerFox > C:\_DAT\APNAME.LST
ECHO C:\_SYS\HB.EXM,2300,Hearts&Bones >> C:\_DAT\APNAME.LST
```

The first line creates the file with the > symbol. The second line, using >> instead of the > symbol, adds another line to the end of the file. If you're distributing commercial software for the HP 95, appending an entry to the end of APNAME.LST with output redirection is a polite, easy way to make keyboard assignments.

Another use for appending output to a file is to create a log of system activity. You could add these three lines to AUTOEXEC.BAT to have the computer automatically maintain a log of the status of the computer each time you reboot:

```
ECHO Rebooted computer >> C:\_DAT\WORK.LOG
BATTERY >> C:\_DAT\BOOT.LOG
NOW >> C:\_DAT\BOOT.LOG
```

Each time you run the batch file, these commands will append three lines to the file named WORK.LOG in the C:_DAT\ directory. The first line is text, the second and third lines are the redirected output from two programs. BATTERY.COM displays the current battery level. NOW.COM displays the current time and date. Instead of displaying the results from these two utility programs, the information is sent to the file.

See also 

BATTERY.COM and NOW.COM are included in this book. Page 94 shows how to enter these programs.

The < Redirection Symbol

This symbol is often thought of as the opposite of >. It takes the contents of a file and sends it through a program. The file becomes StdIn. The input redirection symbol is easiest to visualize as similar to TYPE, but instead of displaying the information, you are sending it to another program.

This example says to DOS "Send the contents of the file BATTERY.SCR to DEBUG.EXE:"

```
C:\_SYS\DEBUG < BATTERY.SCR
```

Input redirection works only with programs that receive input from StdIn.

See also 

Input redirection is often used to enter DEBUG script files. Instead of entering the commands directly in DEBUG, we place them in a text file, then redirect the text file to DEBUG. See page 88 for an introduction to DEBUG.EXE.

The | Piping Redirection Symbol

Piping means to sent the output from one program (or process) to the input of another. This feature is useful primarily with filters like DOS' MORE command, but unfortunately the HP 95 doesn't include any filters.

MS–DOS will run the first program, redirecting the output to a temporary file. After the first program ends, DOS runs the second program, and redirects the contents of the temporary file to StdIn. The second program sees the data as the contents of any stream. Finally, after the second program ends, DOS erases the temporary file.

If you could halt the process midway-through, say around the pipe symbol, then look at the directory, you would find a strangely named file. But when the programs end, that temporary file will be gone. One side effect of piping is that you must have enough disk space available for the temporary file, yet you'll never see the file.

The piping-form of redirection is most useful with filters. The HP 95 doesn't have any filters, so here's the equivalent to the input redirection example above, but re-written using a pipe:

```
TYPE BATTERY.SCR | C:\_SYS\DEBUG
```

This says to DOS: “Type the file BATTERY.SCR, but instead of sending it to the display, send it to the StdIn side of DEBUG.” There will be a short delay while BATTERY.SCR is copied to the temporary file, then sent on to DEBUG.

Filters read the input stream then modify the information according to how it feels about the data, then it displays whatever is left. For instance, MORE reads and displays a screen-full of information then pauses for a keystroke, then it displays another screen-full, and so forth through the end of the file. Another utility might translate the HP 95’s character set to plain-ASCII.

Say you have a program that needs a single character input, but it doesn’t read the command line. You could run the program then press the key at the prompt; a bother if you use the program often, and always with the same keystrokes. Fortunately, there aren’t many programs this dim-witted anymore. One that remains is a public domain hard disk head parker called SHIPDISK.EXE. This example echoes the number 1 (for the first hard disk) and pipes it through SHIPDISK:

```
ECHO 1 | SHIPDISK
```

Caution

Always keep in mind the difference between the output redirection and piping functions. If you accidentally redirect information *to* a program instead of piping it *through* the program, you will overwrite the program file with the output data. For instance, say you want to list a long text file and you are using a utility named MORE to split-up the file into screen-size bytes. The correct command line will look like:

```
TYPE MYFILE.TXT | MORE.COM
```

If you accidentally used the > symbol, as in:

```
TYPE MYFILE.TXT > MORE.COM
```

you would overwrite MORE.COM with the contents of MYFILE.TXT. To lessen the chance of causing any damage with this mistake, enter the program name without the file extension. Then, the worst that can happen is DOS will a create file with the same base-name as the program.

Caution

Be careful with piping with program that expect and accept only a limited number of options. When input is redirected and there is nothing left to read, the keyboard is inactive and the program cannot ask for information. For example, the following will send the incorrect information to the TIME command, and since TIME expects redirected input, it will ignore the keyboard:

```
ECHO ? | TIME
```

The solution is to press **CTRL**+⇐ (the break key) when the computer gets stuck. Even **CTRL**+**C** won’t work here, so that’s the only solution. Many programs disable the break key; avoid using redirection with those programs if you cannot be assured that they’ll get the right information sent to them.

Notes About Redirection

- MS–DOS adds an extra level of device independence. By using the devices, programs and users can share information transparently.
- If you have redirected the console device using the CTTY command, many utilities will receive input from the external keyboard and will display information on the external device. A useful application for this feature is running DEBUG on the 95, but use the keyboard and display from a larger computer.
- Even if you redirect output to the NUL device, some programs will still show error messages on the DOS command line. They do this by writing directly to the display, or by using the StdErr (Standard Error) device. StdErr is rarely redirected.
- You can use >> to append information to the end of a file at any time. If the file does not exist, DOS will create it for you.

- Not all programs use DOS for input or output. Redirection will be ignored by programs that use the BIOS for keyboard and display, and “impolite” programs that communicate directly with the hardware.
- Most programs in the HP 95 Utility Pack accept input from the command line, input redirection, or they will ask you for information. Utility programs usually aren’t that forgiving, so you will have to determine how programs require data.
- Don’t confuse the purpose of the output redirection symbol > with the piping symbol |. If you do, you may accidentally overwrite the program with the redirected output.

Viruses And Worms

If you feel the computer is acting flaky for no apparent reason and all rational solutions have failed, there is a chance — a very slim chance — that your computer has been the brunt of a vandal.

The HP 95 is less susceptible to sabotage than most desktop computers. The operating system and most applications run from permanently programmed ROM chips (there are no hidden boot files), and the hardware is self-contained and sealed.

A virus is transferred from computer to computer by running a program. The program copies itself to the next computer or to another program. There are several kinds of viruses: worms, Trojan horses and bombs, but they all have similar goals. The purpose is to eventually inflict damage on as many computers and people as possible. A virus program can show immediate effects, or it may spread for months or years undetected before a special date or event triggers it.

Perpetrators are out to show us how clever they are — or they’re simply vindictive. It really doesn’t matter. The goal is to avoid them and their wares.

Small companies and major institutions have been brought to their knees by these people, but you can’t let them control your life. So what do about protecting yourself? As your Mother told you: “Don’t touch it if you don’t know where it’s been.” Here are a few more suggestions:

- Use write protect tabs on floppy disks. Flip the write protect switches on RAM cards and 3½" disks.
- Always know the source of a file or disk. Don’t borrow or loan disks.
- Watch program file sizes and date stamps (COM, EXE or SYS extensions). If these files change, a virus may have hooked itself to the program file. Don’t run the programs! Start with fresh copies, and check them again often.
- Avoid pirated and questionable public domain software. There is no such thing as a free lunch. Before downloading that “free” software from a bulletin board, ask yourself how badly you really want it.
- If you decide to try shareware or freeware, make sure that the archived files and disks are the originals, created by the software distributor.
- Viruses often spread by hooking themselves onto programs everybody uses. Like MS–DOS itself, or Lotus 1–2–3. The HP 95 runs these applications from ROM, so the problem is minimized. On your desktop computer, keep an eye out for changes to COMMAND.COM.
- If you are still worried, try fumigating your computer with a virocide program. Be aware, however, that some shareware/freeware virus detecting programs are themselves viruses.
- People who write or pass viruses are sociopaths and vandals. Avoid their haunts.

DOS Pronunciation Guide

We usually try to pretend that computer jargon, acronyms and abbreviations are real words. This leads to confusion and accidental humor, especially if the words have counter-intuitive pronunciations.

After saying “Con-fig-dot-sis” a dozen times on the phone, we often shorten it to “Con-fig-sis.” After another dozen times, it becomes “Con-fig.”

A roomful of compu-techno-nerds pronouncing punctuation quickly begins to sound like Victor Borge’s stage act. Here are a few social gaffes to avoid:

“DOS” is pronounced with a short O, as in Moss, not Dose.

“EXE” is usually X-E.

“.” is pronounced Dot, not Period. Except software revision numbers are often pronounced Period, or even not at all (version 3.2 might be called Three-point-two-two or even Three-two-two).

“*” is pronounced Star, not Asterisk.

“$” is Dollar. In the BASIC programming language, it’s often called String.

“..” is Dot-dot.

“*.*” is “Star-dot-star.”

These are only suggestions; there are regional differences. After all, software people usually say “Data” as “Date-ah” and begin “Gig-a-byte” with a short “G” sound, while hardware people “Dat-ah” and say “Jigga-byte.” Finally, the problem remains: how do you pronounce the file name DOT.DOT?

Customizing MS–DOS

We've already talked about personalizing the HP 95 by adding new System Manager programs and a custom Business Card. MS–DOS adds even more ways to adapt the computer to our needs. We'll talk about these features and writing batch files in this chapter.

Customizable MS–DOS Personality Traits

Feature	What it Does
Path	Where MS–DOS searches for program files.
Prompt	DOS' greeting on the command line prompt.
CONFIG.SYS	Loads device drivers, configures memory.
AUTOEXEC.BAT	Batch file, run when the computer boots.

MS–DOS doesn't have to be an austere, impersonal place. You can settle for Spartan simplicity, or batch files and settings that make your computer like no other. DOS is malleable, the choice is yours.

Environment Variables

The MS–DOS environment is a storage space set aside for text needed to run DOS and share information among programs. It's a succession of named cubbyholes, not unlike string variables in the BASIC programming language. The SET command creates or deletes environment variables. It also shows the current settings; at the DOS command line, type:

```
SET
```

On a freshly booted HP 95, MS–DOS will show something like:

```
PATH=
COMSPEC=COMMAND
```

Only two variables are automatically created. We'll talk about these two important settings and a third, named PATH, later in this chapter. Adding or changing variables uses the same SET:

```
SET h=Hello, World!
```

Now when we type SET, DOS will display:

```
PATH=
COMSPEC=
H=Hello, World!
```

Notice that "H" was converted to upper case, but the text stayed how we typed it. The SET command also deletes variables. To get rid of that less then helpful H variable, type the variable name with the equals sign but without any text:

```
SET H=
```

Caution

When creating environment variables, don't place spaces before the equals sign, because MS–DOS will preserve them in the variable names. This confusion can cause multiple, seemingly identical environment variables — not to mention confused programs. The following two statements are not equivalent:

```
SET ZYZZX=Nevada
SET ZYZZX =Nevada
```

See also

The SET command creates or destroys environment variables. DOS does not have a method to display individual variables. We'll show how to use separate variables in batch files in *The Other ECHO,* starting on page 71.

Notes About Environment Variables

- Watch out for extra spaces in variables and names. Spaces don't show, but they can change the meaning of the variable, or even the name. This really confuses programs.
- Many DOS programs use variables. Built-in HP 95 applications do not.
- Environment variables follow the same naming conventions as files.
- New environment variables are lost when you return to Filer.
- The HP 95 environment is only 160 bytes. Conserve!

Customizing the PATH

The word "path" has two similar meanings. The path can be the current path — meaning the current directory and disk drive — which on the HP 95 is often C:_DAT\. The second meaning is the search PATH Environment Variable. The PATH variable tells MS–DOS where to look for applications programs, and sometimes tells programs where to look for themselves.

Each directory in the PATH string is separated by a semicolon:

```
PATH C:\;C:\_SYS;A:\
```

This says to MS–DOS: "If you can't find the program in the current directory, look next in the root directory of C:, then in C:_SYS\, and finally on the A: drive." Now, regardless of where you are, you can type the name of your favorite program and DOS will look for it.

When your programs are in the path, regardless of where you are, the computer will be able to find your programs, it isn't necessary to change to the directory containing the programs. In the struggle for organization, this one tool makes all the difference: we can keep the programs in one place — such as on a read-only RAM card — and our data files in another. This is especially powerful when it's time to back-up files, since we can simply copy everything from the data directories.

If you're using only built-in HP 95 applications, you have executable programs in the root directory and in the _SYS\ directories of the C: drive. This path setting tells DOS to look in those two places for programs, regardless of where you are right now:

```
PATH C:\;C:\_SYS
```

You could also add an equals sign:

```
PATH=C:\;C:\_SYS
```

Or even:

```
SET PATH=C:\;C:\_SYS
```

All three forms work because the word PATH, like PATH, is both an environment variable, and a DOS internal command. The command version has one difference, however: it converts the string to upper case, the SET command preserves the case.

Caution

Avoid setting the path to include the A: RAM card unless you always keep a card in the port. If the card is in your pocket instead of the port and DOS can't find the program, it'll complain:

```
Not ready error reading drive A
Abort, Retry, Ignore?
```

Not a lot of choices. You can turn the computer off, insert the card, then press **R**. If you know the program can't be found — maybe you mistyped the file name — press **A** to return to the DOS prompt. Ignore isn't a choice — it's difficult to ignore the fact that there isn't a disk in the drive. Some incarnations of MS–DOS add a forth choice: Fail, which isn't any more helpful than ignoring the problem.

One scheme favored by many DOS fans is placing data files in a subdirectory off of the program file. This doesn't seem practical because it just makes for more wandering throught the directory tree, but you can make it more usable by adding the next directory up (the ".." directory) to the path variable. This says, "Look in the usual places, but if you still can't find it, try in the next directory up from where I am now:"

```
PATH C:\;C:\_SYS;..\
```

Most built-in HP 95 applications do not use the path. As a rule, they expect to find help files in the hidden C:_SYS\ directory, and data files in C:_DAT\. Changing the path won't alter how these programs work. This doesn't mean that everything is going to pile-up in the C:_DAT\ directory. Lotus 1–2–3 expects spreadsheet files in the C:\ root directory, but we can customize the setting to any directory we choose. Memo starts in the C:_DAT\ directory, but we can store files anywhere.

Notes About the PATH

- Unlike most HP 95 built-in applications, Lotus 1–2–3 lets us permanently select a default path for spreadsheet files.
- MS–DOS looks for files first in the current directory, then in the PATH environment variable.
- You can list several directories and disk drives in the path; separate each with a semicolon (";"). It is not necessary to place a semicolon at the end of the line.
- Any non-existent or invalid directories or disks specified in the PATH statement are ignored. If MS–DOS can't seem to find your program files, be sure that you entered the path correctly.

Customizing the PROMPT

There you are, staring at C> trying to figure out where you are. You could type CD without an argument whenever you get lost, but there's an easier way: the computer can tell you without you asking. That's the purpose of the customizable PROMPT.

Each time DOS stops and waits for the next command it displays the drive letter and a greater than symbol; that's called the *command prompt*. You can customize the prompt with the (as obvious as it sounds…) PROMPT statement, which is stored in the environment in (even more obviously…) the PROMPT environment variable..

See also

See the Prompt Symbols table on page 111. Some of these codes may seem frivolous, but each has a purpose. Some of the PROMPT symbols are included to avoid conflicts with redirection characters — otherwise, there would be no way to enter these characters without inadvertent piping or redirection. You can embed other characters, even spaces, in the prompt.

The default prompt is "NG," showing the drive letter followed by the greater than symbol. Just about the first thing a Power User does when customizing a computer is type:

```
PROMPT $P$G
```

Now, the prompt will include the complete path with the greater than symbol:

```
C:\_DAT>
```

Codes can be quite elaborate without having a noticeable effect on how fast the prompt returns after each DOS command. This long convoluted example is more useful on a deskbound computer with a color monitor and the ANSI.SYS device drive (loaded via CONFIG.SYS). It displays the current date and time in red letters on a white background in the upper right corner of the screen, displays the path at the cursor position in cyan on black, then displays the cursor in white on black:

```
PROMPT $E[s$E[1;54H$E[0;31;47m $t$E[3D  $d $E[36;40m$E[u$P$G$E[0;37m
```

You probably wouldn't want to use this convoluted prompt on the HP 95 — partly because the 95 can't display red letters, though mostly because the HP 95 doesn't use the ANSI display driver — so we won't explain all of the ANSI escape sequences.

You could show the complete time stamp, with hundredths of seconds, but it takes up quite a bit of the display, and it's more distracting than useful. A better idea is to let the computer display the complete time, then use the "$H" code to erase unneeded bits of seconds, then replace it with the date or a more useful part of the prompt. Be sure to enter spaces as shown.

Useful HP 95 PROMPTs

PROMPT	Display
`$N$G`	`C:>`
`$P$G`	`C:\_DAT>`
`$T$H$H$H $P$G`	`12:34:56 C:\_DAT>`
`$T$H$H$H$H$H$H $D$_$P$G`	`12:34 Fri 09-04-1992` `C:\_DAT>`

Note

The default prompt, "NG," is probably enough if you spend most of your time running built-in applications. Once you're accustomed to seeing exactly where you are just by looking at the prompt, it's hard to go back to simpler prompts.

Notes About PROMPT

- Entering the PROMPT command without an argument restores the default (C>) prompt. This action is quite different from PATH, which shows the current path setting.
- The customized prompt is lost when you return to Filer or reboot the computer. It is *not* lost when you run a program from the DOS command line.
- The equals sign is optional when you enter a PROMPT. Either PROMPT=PG or PROMPT PG is acceptable. Under MS–DOS 2.x, the equals sign was required.

CONFIG.SYS

When the HP 95 boots, it searches first the plug-in card port, then the root directory of drive C:, for a file named CONFIG.SYS. This is a special text file containing configuration commands. By adding a few lines to CONFIG.SYS, you can customize how the HP 95 works and even save some memory. You can edit the file with the HP 95's Memo program or any other plain-ASCII text editor.

CONFIG.SYS Commands

Command	Default	Purpose
`BREAK=`	`OFF`	Turn on or off extended **CTRL**+**C**/**CTRL**+**BREAK** checking.
`BUFFERS=`	`3`	Number of temporary file buffers.
`COUNTRY=`	`001`	Internationalizes time, date and number formats.
`DEVICE=`	·	Installs device drivers or TSRs.
`FILES=`	`20`	Maximum number of files that can be open at one time.
`LASTDRIVE=`	`F`	Number of logical disk drives in the computer.
`SHELL=`	`$SYSMGR`	Specify the command processor shell.
`STACKS=`	`9,128`	Internal stacks used for hardware interrupts.

Caution

DOS isn't very forgiving about experimenting with CONFIG.SYS settings. Be careful creating or modifying CONFIG.SYS. If it's incorrectly written, you may not be able to boot the computer, and may have to re-initialize the RAM disk to recover. This is also true if you try to load a device driver (usually a file with the SYS extension) that isn't compatible with the HP 95.

CONFIG.SYS Remark Text

MS–DOS 4.0 and later add a comment feature; the keyword is REM. This means you can add lines to the file as reminders, which will be ignored when the computer boots. REM is great

when you're trying various combinations of commands and constantly rebooting — you can find your way back to the previous version painlessly. DOS 3.22 doesn't have a REM command, but you can use it anyway:

```
REM FILES=20
```

When DOS comes upon the line, it displays the following message, then continues anyway:

```
Unrecognized command in CONFIG.SYS
```

Note

The command will be ignored, then DOS will continue to the next line in the file. As we will see on page 68, when we talk about the FCBS command, this error message doesn't always mean that DOS has ignored a command.

BREAK= Command

The default is extended break checking off (BREAK=OFF), and for good reason: it adds little, and just slows down DOS utilities. The **CTRL+C** or **CTRL+⇐** (**CTRL+BREAK**) keystroke will terminate utility programs and DOS commands. You can also turn break checking on or off from the keyboard with the BREAK command:

```
BREAK OFF
```

Type the keyword on the DOS command line without any argument to see the current extended break setting:

```
BREAK
```

The computer will display:

```
BREAK is off
```

Normally DOS checks for these keystrokes only when writing text to the screen or waiting for a keystroke. During long-running processes, such as copying large files, **CTRL+C** may be missed until the deed is done. Extended break checking causes DOS to check the keyboard more often — during every internal system call. This can make it easier to cancel out of control processes, but it slows general operation of the computer slightly.

HP 95 built-in applications ignore the state of break checking, as do most sophisticated programs; there's usually little need to change the break setting.

BUFFERS= Command

MS–DOS uses buffers for temporary storage when reading or writing files. Buffers require about 528 bytes each (the size of a disk sector plus 16 bytes). The default setting is BUFFERS=3.

The HP 95 uses a RAM disk, which shows minimal (if any) benefits from buffering. Limiting DOS to a single buffer increases memory available to programs. Adding the following line to CONFIG.SYS increases memory by about 1056 bytes:

```
BUFFERS=1
```

The built-in applications store data in RAM, so there's little interaction with the RAM disk except when you save data to disk. DOS file operations will run slower with only a single buffer, but not noticeably on the HP 95, since RAM disks are by nature much faster than floppy or fixed disks.

DEVICE= Command

Device drivers add an extra level of independence from physical devices. Communicating directly with the printer hardware depends on there actually being a printer connected. If you instead have a driver, whose job it is to communicate with the printer, you could This also makes it easier for programs to run on less-than-compatible computers. Just as important, say a program tries to print when there isn't a printer, the device driver could catch the error and

send the information to the serial port, the display, a file, or even to nowhere at all — the program could print without concern about whether there is a printer.

See also

We'll talk more about the DEVICE= command on page 83 when we compare device drivers and TSRs, and the purpose of device drivers on page 55.

When you have the choice between a TSR and a device driver, chose the device driver; it will take less memory and will be less likely to have compatibility problems.

Device drivers don't have to be in the root directory. Since the these files are accessed only once, when the computer boots, there's no reason to keep them around cluttering the root directory. Say you're using WINK.SYS, and it's in the directory C:\BIN\, you can tell DOS to look for the file in the sub-directory:

```
DEVICE=C:\BIN\WINK.SYS
```

Caution

If a device is incompatible in any way with the HP 95 and stops the computer from booting or operating correctly, you may not be able to restart the computer without first reformatting the RAM disk — this will cause the loss of everything on the disk.

If you have a plug-in card, you can work-around this problem: place identical CONFIG.SYS files on the card and the C: drive, but remove the DEVICE= line from one of the files. Now if the computer crashes when you boot from the card, remove the card and boot again; if you booted from C:, insert the card and boot from A:. Once you've recovered, be sure to eliminate the offending file and DEVICE= statement. A quick-fix when you don't want to (or can't) edit CONFIG.SYS is to rename or delete the device driver. DOS will complain about a missing device, but it will boot the computer anyway.

FCBS= Command

Older MS–DOS applications often used File Control Blocks (FCBs) instead of file handles to open files. FCBs are cumbersome and outdated, and few programmers even like to use them any more. Yet DOS maintains support for File Control Blocks — otherwise, programs from the early '80s might not run!

FCBs are (very) occasionally useful in programs, so you probably don't want to eliminate them entirely. The default setting is four file control blocks. Each FCB costs about 56 bytes, reducing the number to two could save 112 bytes — if we had a way to do it.

There is a way, though the computer complains about doing it. The FCBS= command is not documented for MS–DOS 3.22, yet the HP 95 recognizes it:

```
FCBS=2
```

When MS–DOS reads the FCBS= statement, it issues the "Unrecognized command in CONFIG.SYS" error message. But, the command still works!

Caution

While the FCBS feature appears to work, there is no guarantee that it will be reliable on all HP 95s. Decide for yourself if you would like to use this feature. Old-style applications that depend on FCBs to open files will be limited to two files; this is usually enough.

FILES= Command

The FILES command tells DOS the maximum number of files that may be open at one time. MS–DOS reserves about 56 bytes of memory for each open file. The default for MS–DOS 3.22 is a maximum of 20 files (with other DOS versions, the default is usually 8 files). The maximum setting allowed by MS–DOS is 255 files.

You can fudge a bit, and reduce the number of files to save memory. With FILES reduced to just 10, you can still run all built-in applications simultaneously with complete access to the online help files. This example will save about 560 bytes when you reboot your computer:

```
FILES=10
```

A more conservative setting of 12 will still save 448 bytes.

LASTDRIVE= Command

MS–DOS allows 26 drives, designated A: through Z:. A small amount of memory is reserved for each possible disk drive letter, up to a maximum determined by DOS at boot time (usually E) unless you specify another amount; the HP 95 defaults to LASTDRIVE=F.

Some MS–DOS versions will run in slightly less memory if you specify only as many drives as needed; the HP 95 appears to reserve the same amount of memory (.5K), even if you set LASTDRIVE=C.

Note

It is not necessary to add a colon after the drive letter.

SHELL= Command

The first time your HP 95 booted-up, the System Manager was in charge. The System Manager (also known as $SYSMGR) is the default MS–DOS shell for the HP 95. The shell program, then, is the front-end of MS–DOS; it responds to our keystrokes, displays information and runs programs.

Other MS–DOS computers use a shell named COMMAND.COM instead of the System Manager. You can make the HP 95 work like your deskbound computer by specifying the same command interpreter:

```
SHELL=COMMAND /P
```

Now, when you reboot the computer the HP 95 will act just like a typical (boring) MS–DOS computer. You can return the built-in applications by typing the $SYSMGR command on the command line:

```
$SYSMGR
```

Loading the command processor is the final phase of booting the computer. Once MS–DOS passes control to the command processor, it is the command processor that executes AUTOEXEC.BAT The HP 95 $SYSMGR, does not look at the AUTOEXEC.BAT file.

Caution

The /P switch after COMMAND is vital to ensure that if the user types EXIT, the computer isn't halted. Much like **MENU Q** exits most HP 95 built-in applications, EXIT is the keyword used by COMMAND.COM to quit. And it will. With no command processor running the computer and no program to return to, nobody is there to respond to our keystrokes — nothing happens. Always remember the /P switch when you use the SHELL statement!

Notes About CONFIG.SYS

- DOS will only interpret the first file of this name that it finds. This file is interpreted only when the computer boots. If you make changes to CONFIG.SYS, you will have to reboot (press **CTRL**+**ALT**+**DEL** at the same time) for the changes to take effect.
- Adding the line LASTDRIVE=C does not save memory.
- You cannot set STACKS=0,0 to save memory with DOS 3.22, while this command would be useful with DOS 4.0 or later. The HP 95 Owner's Manual states that the STACKS command is set at 9,128. Tests seem to confirm that the computer is actually using STACKS=0,0.
- For memory conservation, consider reducing the values set with the BUFFERS, FILES, and FCBS commands.

Batch: DOS' Programming Language

It's a joy breezing through files and directories with the HP 95 Filer. But that pleasure is short lived when we pick-off a program to run by pressing **F4**, and the program complains about missing arguments. Batch files are a semi-elegant solution to that problem, turning mystical commands and switches into simple, intuitive, customizable keywords.

Batch files are just text files — you can edit them with Memo or any other text editor — containing MS–DOS commands just like we enter at the DOS prompt. When we run batch files, DOS replays the commands. Batch files always have the BAT file name extension. The batch language adds only a few new keywords; the next few pages talk about these commands.

Batch File Commands

Command	Purpose
ECHO	Displays text, enables or disables command echoing.
FOR	Repeats commands for a series of files.
GOTO	Go to a label. Labels begin with a colon (:).
IF	Conditional test.
PAUSE	Stop and wait for a key press.
REM	Remark.
SHIFT	Moves command line arguments left.

Note

Very few people learn to use FOR, IF and SHIFT. They're powerful and flexible commands, but they are usually more than we need. Please take the time to read those sections, but you probably won't use them frequently enough to worry about remembering the syntax. Besides, you can always refer back here later.

Every program and utility we add to our HP 95 adds to the power of MS–DOS — and extends the batch language. For everything we can type from the keyboard can be automated in batch files. Every convoluted command string we struggle to commit to memory (or commit to a cheat sheet taped to the back of the HP 95) can be preserved in batch files.

See also

Page 109 lists useful MS–DOS commands.

Say you spend most of your time with the built-in HP 95 applications and only occasionally run DOS programs. You could set environment variables and suchlike with AUTOEXEC.BAT, but that would waste memory that could be put to better use creating larger 1–2–3 files. One solution is writing a batch file to set the PROMPT and PATH when you want to switch to DOS, then give the memory back when you return to Filer. This little file does just that:

```
ECHO OFF
PROMPT $P$G
PATH C:\;C:\_SYS
ECHO (Type EXIT to return to Filer)
COMMAND
```

Notice that we added C:_SYS\ to the path. That's so DOS will be able to find DEBUG.EXE. But batch files can do much more. Say you use a program called HOG.EXE that requires environment variables and an odd start-up directory, then leaves the display a mess when it's done (this is not all that unusual). This batch file recovers from all these annoyances, demonstrating WHEREAMI.COM in the process:

Batch Command	What it Does
WHEREAMI >C:\GOHOME.BAT	Save the current directory.
SET IMAHOG=zyzzx	Allocate the environment variable.
C:	Log onto the C: drive.
CD \HOGDIR	Change to the odd directory.
HOG	Run the program.
COMMAND /c C:\GOHOME	Return to the original directory.
DEL C:\GOHOME.BAT	Delete the temporary batch file.
SET IMAHOG=	Release the environment variable.
CLS	Clean-up the display.

See also

WHEREAMI.COM formats the current directory as a two-line batch file. In the example, WHEREAMI's output is saved in the temporary batch file named GOHOME.BAT. WHEREAMI is explained on page 99.

Note

Usually we edit batch files with the HP 95 Memo program. For very short batch files, try DOS' COPY CON command. There are virtually no editing features (it's the same command line editor we always use), and if you make a mistake you'll have to start over, but for simple tasks or tests it's more than enough. This example creates a single line file with a DIR command:

```
COPY CON DIRW.BAT
DIR *.* /W/P
```

Press **ENTER** after each line. When you're done editing, press **CTRL+Z** then press **ENTER** to mark the end of the file. DOS will display "1 file(s) copied" to let you know the file was successfully created. Now, whenever you want to see a wide directory listing just type DIRW (or select the program file from Filer), and DOS will run the batch file.

ECHO Command

The ECHO command has two functions in life: to enable or disable displaying commands as the batch file runs, and to display messages.

As DOS interprets each batch file command, it *types* each line to the display, as if somebody is inside the computer, typing from the other side of the keyboard. You can use ECHO OFF to turn off the display of each command and the redundant redundancies:

```
ECHO OFF
```

This does not stop the other ECHO command from working, it just suppresses the display of extraneous stuff. Unfortunately, each time you use ECHO OFF, MS–DOS displays that line, you can't redirect the results to the nul device (well you can, but it doesn't work).

On the long shot that you might want to turn echoing back on within a batch file, DOS offers another option:

```
ECHO ON
```

When the batch file ends, MS–DOS turns command echoing back on anyway, so the command is rarely useful. However, don't use ECHO OFF from the keyboard because the command prompt and other important information won't be displayed. You can remedy this situation by entering ECHO ON, again.

Note

With version 3.3 and later, MS–DOS recognizes the "@" (at sign) to mean "Don't echo this line." Many batch files printed in magazines and books begin with "@ECHO OFF", a nice little command that turns off echoing without telling us each time (alas, it's not available in MS–DOS 3.22). If a batch file from a magazine doesn't work on the HP 95, see if any lines begin with the at sign. Deleting the "@" characters should make the files run fine.

The Other ECHO

ECHO, like DISP in HP–BASIC or PRINT in Microsoft BASIC, displays messages on the screen. It can display a simple line of text:

```
ECHO The sky is falling!
```

Or, as we'll see throughout this section, it can display environment variables and command line arguments:

```
ECHO The command processor is: %COMSPEC%
```

ECHO with just a space or nothing at all will cause DOS to display:

```
   ECHO is on
```

Fine, but how do you display a blank line? ECHO will ignore a period following the ECHO command; if that's all there is, DOS will display a blank line:

```
ECHO.
```

You could also use the plus, colon or quotation mark, but the accepted convention is to use a period.

Note

ECHO, like all DOS commands, requires a space between the command and any arguments. ECHO will ignore the first space (which is why "ECHO." works), but any additional leading spaces will be displayed. A tab character looks like several spaces on the display, but if you start with a tab instead of a space, DOS will delete the tab, then show the text left-flush.

MS–DOS recognizes ASCII character 7 as the bell — when you echo character 7 to the display, instead of displaying a character, the computer beeps. You can enter the bell character by pressing **CTRL+G**, which MS–DOS displays as "^G":

```
ECHO ^G
```

Every ECHO is displayed as if you were typing it, then again as it is executed. If you want your batch program to look attractive, remember to use the other ECHO — ECHO OFF — at the beginning of file.

Note

When we redirect ECHO's work, the message is not displayed. This is fine until we want to tell our users to change disks or give them a chance to quit before something awful might happen. We've included a program, called ECHO2.COM, that works like ECHO, but it ignores output redirection. This command will just about always send the message to the display:

```
ECHO2 Busy... Don't press Ctrl+C right now! >NUL
```

See also

ECHO2 works by displaying text through the StdErr device instead of StdOut. You'll find program listings for ECHO2.COM on page 95.

If you call ECHO2 without anything to display, instead of displaying something annoying like "ECHO2 ON" it'll display a blank line. When it's not important if output is redirected, use ECHO instead of ECHO2, it will often be faster because internal commands have less overhead than external commands — DOS will have to load the ECHO2.COM program file for each call. Since the HP 95 uses a RAM disk, the time to load programs is barely noticeable.

Batch File Arguments

Like COM and EXE programs, batch programs can accept command line arguments. We access these arguments in our batch programs with *replaceable parameters* preceded by the percent sign. Replaceable parameters are placeholders for individual command line arguments. This example passes the first and second arguments from our batch file to DOS' REN command:

```
REN %1 %2
```

Think of the command line as a series of individual words separated by spaces. The replaceable parameter for the first word is "%1," the second is "%2," on up to "%9." DOS also provides us with "%0," which is the command used to start the program (complete with path and the file name, if you furnish them).

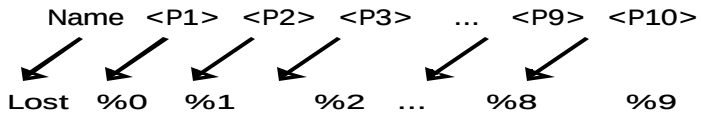
This example, called ECHOALL.BAT, displays the batch program name, then the first 9 arguments:

```
ECHO OFF
ECHO The program name is %0
ECHO %1 %2 %3 %4 %5 %6 %7 %8 %9
```

When there are fewer command line arguments than replaceable parameters, the extra parameters are removed, replaced on the command line by nothing.

Batch File SHIFT Command

The SHIFT command copies the value of replaceable parameter "%1" to "%0," "%2" to "%1," and so forth. If there are more than nine command line arguments, the next argument will shift into "%9." As you shift the arguments over, unused replaceable parameters are removed, until even "%1" returns nothing. After each SHIFT, "%0" will contain the previous contents of "%1." This illustration shows each replaceable parameter (P1... P10) shifted one position to the left:



Note

The illustration on page B-21 of The *HP 95LX User's Guide,* Edition 1, demonstrates the results of DOS' SHIFT command inaccurately. The similar illustration above corrects the oversight.

When there are more than nine or a variable number of arguments, or the batch file is easier to write when you always refer to the same argument in a loop, try SHIFTing. This example, called ECHO1.BAT, displays one word from the command line each time through the loop:

Batch Command	**What it Does**
`ECHO OFF`	Turn off command echoing.
`ECHO This program is: %0`	Display the program name.
`:Start`	Top of loop label.
`IF "%1" == "" GOTO End`	Quit if no more arguments
`ECHO "%1"`	Display the text.
`SHIFT`	Shift the next argument into place.
`GOTO Start`	Return to the top of the loop.
`:End`	Label at the end of the file.
`ECHO Done! %0`	Say that we're done.

Execute this batch file with a command line something like:

```
ECHO1 hp 95 lx
```

As it runs, ECHO1.BAT will display:

```
This program is ECHO1
"hp"
"95"
"lx"
Done! 95
```

Instead of the program name, the last line displayed includes "95," the second to last word on the line, which was moved into "%0" with the last SHIFT command.

Caution

Once you've SHIFTed the arguments over, they are gone forever.

Always check that replaceable parameters exist before passing them to a program or command that needs the information. See IF, later in this section, for a method to ensure that the arguments are valid.

Batch Files And The Environment

Batch files, like *real* programs, need a place for their stuff: temporary path settings, program switches and flags. That place is the environment — the same environment that's used to store the path. This means that variables are handy to all programs, but at only 160 bytes, we can't store very much.

Just to make batch files harder to read, MS–DOS also uses the percent sign to represent replaceable environment variables. Where a single percent sign precedes replaceable parameters "%0" through "%9," we tack a percent on both ends of environment variables. This example displays the current path setting:

```
ECHO The path setting is "%PATH%"
```

Caution

You cannot use the batch file double-percent tricks from the keyboard, only batch files. MS–DOS will only expand the "%PATH%" to the current path while a batch file is running, so if you do it from the keyboard the "%PATH%" will be returned unchanged.

In a batch file, you can assign the contents of one environment variable to another:

```
SET TEMPPATH=%PATH%
```

A great use for this feature is to append something to the current path. Say your program loads from the C:\BIN\ directory and the current path is C:\;C:_SYS\, you can set the path to your new path and the current path:

```
SET PATH=C:\BIN\;%PATH%
```

Now when you type PATH, you'll see that the original path now follows the C:\BIN\ directory. You can also add your directory to the other end of the path, in fact this is the preferred place, because many people carefully design their path — what's first will be searched first — so adding your directory in the front is just impolite.

Caution

One caveat: while DOS'll let us create environment variables that look like replaceable parameters, batch files won't be able to read them.

```
SET 1=Hello!                    Assign the environment variable.
ECHO Arg: "%1%"                 Display the environment variable.
ECHO Var: "%1"                  Display the first command line argument.
SET 1=                          Erase the environment variable.
```

When we run this batch file, DOS will display the command line argument in both ECHO statements — it never sees the "1" environment variable. Environment variables and replaceable parameters are two separate places in memory; you can't change replaceable parameters with SET, but you can copy parameters to the environment:

```
SET MyVar=%1
```

If you would like to every environment variable, type the SET command without any argument:

```
SET
```

Note

If your batch file creates environment variables, remember that the environment is only 160 bytes, so be sure to delete the variables when the program ends. If your program creates "X," delete is with this command:

```
SET X=
```

Batch File GOTO Command And Labels

If nothing gets in the way, batch programs run straight through from top to bottom, a line at a time. Sometimes, though, it's helpful to skip or repeat sections of the file. The GOTO command jumps around the file, from the position of the GOTO command to the program label it points to. For every GOTO, also called an *unconditional branch,* there's a matching label. And the label begins with a colon followed by a name of up to eight characters.

```
GOTO Quack
   .
   .
   .
:Quack
```

After a GOTO, program execution continues on the line following the label. This example will sit in a loop until you press **CTRL+C** to quit the batch file:

Batch Command	What it Does
`:LoopTop`	Label.
`ECHO Press ^C to quit`	Display the message.
`PAUSE`	Wait for the next keystroke.
`GOTO LoopTop`	Return to the label.

MS–DOS searches for labels from the beginning of the file each time. If a batch program contains more than one identical label, only the first label found will be executed. The following example begins at the top of the file, but immediately jumps to the label named Test. The next line jumps to Label, which appears twice in the file:

```
GOTO Test                       Branch around the first label.
```

```
:Label                              Will jump to this line.
ECHO Landed at the first label      This line will be executed.
GOTO End                            Get out of the file.
:Test                               The beginning of the test.
GOTO Label                          Jump to the label named "Label."
:Label                              This line will be ignored.
ECHO Landed the at second label     This line will also be ignored.
:End                                The end of the file.
```

When you run this batch file, DOS will display "Landed at the first label" then jump to the End label at the end of the file. This is because it will start at the top of the file looking for the label. It's also partly why batch files run so slow: DOS has to read each line the batch file separately, then when it finds a label it returns to the beginning of the file and starts again.

Note

Most programming languages place the colon at the end of the label, while MS–DOS requires the colon at the beginning of the line. This isn't just to be contrary, but to save time. Batch programs have enough performance problems without having to look down each line for the colon. By starting the line with the colon, DOS has to look at only a single character when it's looking for labels, so label searches are relatively fast.

Batch File IF Command

The IF command adds life to batch files. Programs can test for current conditions and execute different commands while they run, all without our intervention. DOS' IF command understands three types of tests:

Variations of IF

Command	Purpose
`IF string == string command`	Compare two strings.
`IF ERRORLEVEL value command`	Test the exit code of the previous program.
`IF EXIST filename command`	Test if a file exists.

Any DOS command can follow the comparison. Each of these forms of IF can do negative tests using the NOT keyword. For instance, IF NOT EXIST means the test is true if a file does not exist.

```
IF NOT "%1" == "%PATH%" ECHO ^G
```

We usually compare two variables within quotation marks; that's a carry-over from high-level programming languages. All that's really necessary is a place holder — any single character appended to both strings — in case one of the arguments is empty. Say the user enters no command line at all, DOS will interpret the above as:

```
IF NOT "" == "%PATH%" ECHO ^G
```

That is an understandable (parse-able) statement. But if we leave out the quotation marks, it looks like:

```
IF NOT == %PATH% ECHO ^G
```

DOS will complain with a "Syntax error" message. Any character or characters will do, but we must use something. Here it is with a single quotation mark:

```
IF NOT ! == %PATH%! ECHO ^G
```

Tack a GOTO on the line after the test for *conditional branching*. This gives programs the batch program the ability to think: to decide what to do next based on the outcome of a test. Should the user run the program without an argument, we can even offer helpful suggestions:

```
IF "%1" == "" GOTO Help
```

Note

The equality test is case sensitive. That means the two strings must match exactly for the test to prove true. Keep in mind that DOS stores environment variables without converting them

to upper case, except the PATH, which is (usually) upper cased. Also watch for spaces; MS–DOS removes spaces from replaceable parameters, but not from environment variables.

All is not lost. We can use a quirk of DOS to convert the strings to upper case. Here's a kludge to convert command line replaceable parameters to upper case using the PATH keyword (it won't work with "SET PATH=" which just copies the string as-is).

Batch Command	What it Does
`SET T=%PATH%`	Save the current path setting.
`PATH %1`	Convert "%1" to upper case.
`SET T1=%PATH%`	Get the upper case string to "T1"
`PATH %T%`	Restore the path.
`SET T=`	Delete the scratch "T" variable.
`IF "%T1%" == "QUACK" GOTO A`	Compare the variables.

For simple comparisons that can be evaluated to labels, we can forget about the case problem — labels aren't case sensitive. For each valid option, create a matching label, and DOS will do the work for us if the argument on the command line matches a label. This example foregoes all of the overhead and kludginess of comparisons and environment variables:

```
GOTO %1
```

IF EXIST

The second form of IF actually looks at the disk drive to see if a file exists:

```
IF EXIST autoexec.bat GOTO End
```

Another variation is to check for the existence of a file name in a replaceable parameter:

```
IF EXIST %1 COPY %1 A:\
```

IF ERRORLEVEL

The ERRORLEVEL test doesn't test for errors at all! Perhaps it should be called "EXITLEVEL," because it tests the code returned to MS–DOS when programs exit. Zero (0) usually means success, and ever increasing numbers — up to 255 maximum — means bad news. The problem, though, is very few programs actually return codes that mean anything to ERRORLEVEL.

Say a program returns zero if all is well, and larger numbers as increasingly nasty things happen; if the number is at least four, something disastrous happened. A batch program can display the bad news:

```
IF ERRORLEVEL 4 ECHO Something is amiss
```

In HP BASIC, the equivalent code might look like:

```
   IF E >= 4 THEN DISP "Something is amiss"
```

The ERRORLEVEL test proves true if the program returns a number that is *greater than or equal to* the number we're comparing. This means we need to test larger possible results first, then smaller numbers And we can't just check for "ERRORLEVEL 0" because every program will return at least zero. So we have to start at the largest likely exit code and work our way down (or just generalize).

This file fragment uses MEM.COM (listed on page 96) to check available memory, sending the information displayed by MEM to the bit bucket. It then decides which code to execute based on the ERRORLEVEL returned by MEM. MEM returns 3 for at least 300K memory, 2 for 200K, and so forth; if it returns zero, the computer has less than 100K of memory available.

Batch Command	What it Does
`MEM > NUL`	Don't display the results from MEM.
`IF ERRORLEVEL 3 GOTO BigMem`	If at least 300K available memory.
`IF ERRORLEVEL 2 GOTO MedMem`	If at least 200K available memory.
`IF ERRORLEVEL 1 GOTO SmallMem`	If at least 100K available memory.

```
IF ERRORLEVEL 0 GOTO LowMem
```
If not very much memory available.

Note

Many programs — especially small COM files — don't return a useful ERRORLEVEL. They won't necessarily return zero, even when they exit without error. As we seem to constantly say, check the program documentation for more information. Since ERRORLEVEL is so seldom supported (even by DOS commands), its usefulness is regrettably limited.

Batch File FOR Command

The FOR command repeats a command for each matching argument. A single line in the batch file can execute a command any number of times.

You probably thought using both "%1" for replaceable parameters and "%PATH%" for environment variables was confusing. It's about to get worse. Replaceable variables for the FOR command begin with two percent signs! However, if you use FOR from the keyboard instead of a batch file, you must use only a single percent sign.

MS–DOS is very good at interpreting commands typed in all sorts of ways, but when it comes to FOR, it gets picky about syntax. The FOR command has three compulsory components: FOR, IN and DO. IN and DO aren't really parameters, but markers separating arguments:

```
FOR %%X IN (argument set) DO command %%X
```

The temporary variable named after the two percent signs is like an environment variable which exists only while the FOR command executes. The variable contains the results of the argument for each iteration.

The *argument set,* which is always in parentheses, is any file name or string. It can be one or more wild card file names or any text, but it's always separated by spaces.

The *command* is any MS–DOS command. Notice also that you can pass the variable to the command. This example displays a list of all COM and WK1 files:

```
FOR %%A IN (c:\*.com c:\*.wk1) DO ECHO Files: %%A
```

If the file named in *argument set* isn't a valid file, the FOR command is still executed at least once. For each iteration of the FOR loop, the next argument will be placed in "%%A," then the command will be executed. This program will execute NOW.COM six times (once for each period within the parentheses), displaying the current time and date a half dozen times:

```
FOR %%A IN (. . . . . .) DO NOW
```

The argument was ignored — we used it just as a loop counter. But even if the word isn't a file name, we can still pass it to the command:

```
FOR %%A IN (Twas brilig and the slithy toves) DO ECHO %%A
```

Many DOS commands and programs don't support wild cards. TYPE, for instance, works with only a single file. We can use FOR to repeatedly execute commands, just like wild cards. This single-line batch file, named WTYPE.BAT, takes a single wild card and TYPEs each matching file:

```
FOR %%A IN (%1) DO TYPE %%A
```

To use the batch file, type the file name followed by the wild card argument. This example shows the contents of all TXT files:

```
WTYPE *.TXT
```

Note

From the command line, you can use either one or two percent signs with FOR, but in batch files you must use two percents. This is because a single percent precedes a command line argument. In fact, you can save some typing by using just two percent signs, without specifying a variable name at all (FOR %% IN...) from the command line, but you must specify a variable in batch files.

Batch File PAUSE Command

We can pause a batch file to tell the user to change disks (or RAM cards), and to give them a chance to bail out before a batch file does something potentially dangerous. PAUSE is also useful to stop the program to give us a chance to read the display before reams of information go spilling off the top of the screen. PAUSE will stop and display:

```
Strike a key when ready . . .
```

If this prompt is too violent, try redirecting the prompt to the NUL device and replacing it with your own message. Be sure to place the ECHO command before PAUSE, otherwise your message will be displayed after the user presses the key:

```
ECHO Tap a key lightly to continue . . .
PAUSE >NUL
```

PAUSE also gives us an additional chance to stop a batch file before it does something stupid. Press **CTRL**+**C** or **CTRL**+⇐ while the program is paused and DOS will offer a way out of the program:

```
Terminate batch job (Y/N)?
```

The trouble with PAUSE is it doesn't tell the batch program what happened. We've included a work-around for this limit: WAITKEY.COM, documented on page 99. WAITKEY returns an ERRORLEVEL representing the ASCII value of the keystroke. Letters are turned to upper case, function keys are turned into the numbers "0" through "9" and keys you'll probably not need are ignored (like **ENTER** and the cursor keys).

```
ECHO Tap a reasonable key... like A or M
WAITKEY
IF ERRORLEVEL 27 GOTO EscKey
```

See also

The keys returned by WAITKEY are ASCII codes. For instance "A" or "a" is returned as 65, while "5" is 53. The ASCII table on page 107 lists characters and their numeric equivalents.

Batch File REM Command

Like all self-respecting programming languages, DOS' batch has a remark command to insert notes or comments in your files. REM is also useful to comment-out parts of the file that are in limbo or perhaps don't work correctly.

Be sure to use ECHO OFF before any remarks, otherwise DOS will (most annoyingly) display everything.

Note

Since DOS ignores labels until it finds a GOTO, you can use a colon at the beginning of the line just like REM. Usually we enter a space after the colon. Both of the following will be ignored:

```
REM The cow is of the bovine ilk,
: One end moo, and the other end milk.
```

Perhaps the most important reason to begin comments with the colon instead of REM is speed. MS–DOS interprets REM and ignores lines beginning with the colon, so colon-ized comments run at least ten times faster than REM.

Colon comments have no noticeable effect on the speed of the GOTO command.

Nesting Batch Files

Batch files normally don't call other batch files. If they do, MS–DOS won't return control to the original file. This can be to our advantage, doing completely different things based on command line arguments, the existence of a file, or any number of variables. But when we do need to continue after calling another batch file, there is a way: load a second copy of the DOS command processor with the /C (short for "Call") switch, and have that program call the second batch file.

Say you have a batch file named FILE1.BAT and you would like to execute FILE2.BAT then continue, add the following line to FILE1.BAT:

```
COMMAND /C FILE2.BAT
```

This says to DOS: "Load the command processor then execute the command following /C, then exit and return to the previous command processor." We're not really nesting batch files, but actually nesting copies of COMMAND.COM.

Notes About Batch Files

- Batch files are plain-text. You can edit them with any text editor, such as the HP 95 Memo program.
- Each line must be no longer than 128 characters.
- You can stop batch programs by pressing **CTRL+C** or **CTRL+⇐**. You may have to press the key more than once.
- There is no "end" command for batch files. While program flow might move around the file, the only way to exit a batch file is by directing flow to the end of the file. We usually create an ":End" label at the bottom of the file, then use "GOTO End" to exit the program.
- Release environment variables when your program ends. If you've used a variable named "X," use "SET X=" to release the variable. The environment is limited, every byte counts.
- Watch for rampant percent signs. We use "%1" to read replaceable parameters, "%VAR%" to read environment variables, and "%%1" to mark replaceable arguments with FOR.
- MS–DOS recognizes only the first eight characters in labels (not counting the colon). If two long labels are identical up to the first eight character, DOS will find only the first label in the file.
- When you nest batch files (with the COMMAND /C switch) and assign environment variables, those variables will be automatically erased when the second batch file ends.
- It's especially important to use ECHO OFF in batch files that contain REM lines, to keep DOS from displaying all your comments.
- If you would like your batch file to beep, place the bell character (7) in an ECHO statement. You can enter the character by pressing **CTRL+G**.
- You can eliminate the carriage return/line feed characters from the last line in the batch file. This will eliminate the extra line feed you often see displayed after the batch file executes.
- If a batch file calls another batch file, control never returns to the first batch file.
- MS–DOS versions 3.3 and later have a CALL command, which works just like the COMMAND /C switch, to call batch files. This command is not available in the HP 95's built-in MS–DOS 3.22.
- DOS expects to see an end-of-file marker character (26, often displayed as ^Z, pronounced "Control zee") at the end of batch files. This character is optional. You can enter the character with the HP 95 Memo program by pressing **CTRL+Z**; Memo will display the character as "→". Since Memo can edit files beyond the end-of-file marker, you can add extra characters, such as notes, on the end of the file.
- The batch language is extensible — every DOS program and utility we use enhances batch. Making use of the tools available to us doesn't just make batch better, it might just make us better.

AUTOEXEC.BAT

Most MS–DOS computers interpret the file named AUTOEXEC.BAT immediately after booting; this process is handled by the command processor (usually COMMAND.COM). The HP 95 boots MS–DOS 3.22 from ROM, using the System Manager command processor.

Note

Since it doesn't normally load COMMAND.COM, the HP 95 will normally ignore AUTOEXEC.BAT. The HP 95 will only read AUTOEXEC.BAT if the CONFIG.SYS file includes the SHELL=COMMAND /P statement.

AUTOEXEC.BAT is a text file containing commands just like you would enter from the DOS command line, and just like any other batch file, but DOS executes the commands automatically each time it boots. The only thing unique about AUTOEXEC.BAT is its name.

When you go to the DOS command line from Filer, the prompt and path settings return to the setting used when you booted the computer. You can work around this by creating an AUTOEXEC.BAT file that creates some standard environment variables then passes control to the System Manager (the built-in applications). Here's a typical AUTOEXEC.BAT file:

```
PATH=C:\;C:\_SYS
PROMPT $P$G
$SYSMGR
```

When you start from COMMAND.COM, the $SYSMGR command in AUTOEXEC.BAT passes control to the System Manager.

Notes About AUTOEXEC.BAT

- Place commands to install TSR programs at the beginning of the file, before setting the PATH or other environment variables. This will save memory and minimize memory fragmentation.
- MS–DOS will look for the AUTOEXEC.BAT file on the same disk that contains CONFIG.SYS. If you booted from the RAM card, AUTOEXEC.BAT is expected to be there; otherwise it is expected to be in the root directory of drive C:.
- MS–DOS reserves only 160 bytes for environment variables.
- If CONFIG.SYS includes the SHELL statement but there is no AUTOEXEC.BAT file, you will be prompted to set the time and date each time you reboot the computer. The computer will automatically set the clock if there is an AUTOEXEC.BAT file — regardless of if the file does anything useful (a simple ECHO statement will do).

Configuration Suggestions

Each time you shell out to DOS, any environment variables you create will be temporary, and will be discarded whenever you return to Filer. In order to create a permanent PATH, PROMPT or any other environment variable, you will have to add the SHELL=COMMAND /P statement to CONFIG.SYS, then create the environment variables, then execute $SYSMGR as the last line of AUTOEXEC.BAT. Otherwise, you will have to set these values each time you exit to DOS. Though if you do, approximately 2K will be used for the permanent copy of COMMAND.COM.

Note

Before creating a CONFIG.SYS file just to save a few hundred bytes of memory, keep in mind that the file will require 512 bytes of RAM disk storage space — that memory comes from the same pool as main memory. If you are booting from a RAM card, the 512 bytes spent there won't be as expensive.

For many applications, it might conserve memory to set environment variables whenever you shell to DOS. You can do this by creating a batch file with the appropriate SET commands, and COMMAND (without the /P) as the last statement.

Caution

These suggestions apply only to the HP 95. Many of these options will decrease available memory or operational speed for other computers.

If You Need Environment Variables

The CONFIG.SYS file conserves memory by reducing file buffers to the minimum. It loads a single device driver, then passes control to the command processor. AUTOEXEC.BAT sets a

path so DOS can easily find programs, changes the prompt to show the complete path, then returns control to the built-in applications.

CONFIG.SYS

```
BUFFERS=1
FILES=12
DEVICE=WINK.SYS
SHELL=COMMAND /P
```

AUTOEXEC.BAT

```
PATH=C:\;C:\_SYS
PROMPT $P$G
$SYSMGR
```

You could also load TSR (Terminate and Stay Resident) programs from AUTOEXEC.BAT. To conserve memory, be sure to load the TSR before setting environment variables. This alternate example loads Borland's SideKick, a TSR program named SK.COM:

```
SK
PATH=C:\;C:\_SYS
PROMPT $P$G
$SYSMGR
```

Note

We will use Borland's SideKick as an example of terminate and stay resident (TSR) software several times throughout this book. This is not an endorsement for that product. SideKick was written long before the HP 95 was developed, and as such was not tested for compatibility with the HP 95LX.

If You Don't Need Environment Variables

This is the minimalist setup. It's most practicable if you work mostly with the built-in applications, and occasionally venture out into DOS. This CONFIG.SYS file reduces many system resources to minimal, almost skimpy, settings.

CONFIG.SYS

```
BUFFERS=1
FILES=8
FCBS=2
```

For even more savings, try setting FCBS=1.

Since there is no SHELL statement, the computer will ignore any AUTOEXEC.BAT when it boots. Even if this file exists, it will not be read.

More Room For DOS

There is life outside of Filer. But living there can be a bit cramped. Your HP 95 sprang to life with 512K of memory: 254K for the RAM disk and 258K for system memory. 258K is 264,192 bytes, yet CHKDSK reports only about 163,760 bytes available. Let's get back about 83,472 bytes of those missing 100,432 bytes. That's right: MS–DOS on the HP 95 needs less than 17K of memory — most of DOS runs from ROM, regardless of how you boot it! What we need to do is get the memory back from the System Manager.

You can free the memory with just CONFIG.SYS, but for more flexibility, it takes several files. We'll explain each of these files in a few minutes.

More-Room-For-DOS Files

File Name	Purpose
CONFIG.SYS	Installs drivers and the command shell.
AUTOEXEC.BAT	Commands to interpret when computer boots.
DOS.BAT	Reboots the computer at the DOS prompt.
ROM.BAT	Reboots the computer, running built-in applications.
REBOOT.COM	This program reboots the computer.

Normally when the computer boots, it runs applications in ROM using the $SYSMGR command processor instead of COMMAND.COM. $SYSMGR uses the extra memory for shared data, the copy buffer and other information needed by the built-in applications. When you shell to DOS from Filer, you're actually running COMMAND.COM, the command processor, which in turn runs your DOS application programs.

The trick is to start the computer from COMMAND.COM instead of $SYSMGR, optimize wherever possible, and make the extra memory available to DOS.

Running plain DOS requires nothing more than adding a line to CONFIG.SYS specifying COMMAND.COM as the shell program — the command interpreter (the default command processor if you do nothing, is $SYSMGR, which runs the built-in applications). But to save a bit more memory, use the name COMMAND, without the COM extension, and a special internal command processor will take over (this won't work with other DOS machines — just the HP 95). If you're using a CONFIG.SYS file, add the following line; if you don't have a CONFIG.SYS, create one:

```
SHELL=COMMAND /P
```

Now when you reboot the computer, it will go through the usual startup gyrations, then display something like:

```
Copyright (c) 1991 Lotus Development Corp.
    All Rights Reserved
Copyright (c) Hewlett-Packard 1990
Copyright 1984,1985
    Phoenix Software Associates Ltd
Version A

C>echo off
C:\>_
```

If you start with a plain machine, you can type $SYSMGR to return to ROM-based applications.

To do things right, you will probably want several files: AUTOEXEC.BAT to make sure the date is set, DOS.BAT to boot the computer with as much memory available as possible, and ROM.BAT to return to the ROM-based applications. While you could boot by pressing **CTRL**+**ALT**+**DEL**, we've included a program called REBOOT.COM to do the work for you — and to make the process automatic.

See also

You'll find program listings for REBOOT.COM on page 97.

The following are simple versions of DOS.BAT, ROM.BAT, and the support files.

CONFIG.SYS

```
BUFFERS=1
FILES=10
FCBS=2
SHELL=COMMAND /P
```

AUTOEXEC.BAT

```
ECHO OFF
PATH C:\;C:\_SYS
PROMPT $P$G
```

DOS.BAT

```
ECHO OFF
ECHO DOS.BAT resets the 95 to DOS.
ECHO Press ^C to cancel
PAUSE >NUL
REN C:\CONFIG.TMP CONFIG.SYS
REBOOT
```

ROM.BAT

```
ECHO OFF
ECHO ROM.BAT resets the 95 to ROM apps.
ECHO Press ^C to cancel
PAUSE < NUL
REN C:\CONFIG.SYS CONFIG.TMP
REBOOT
```

When you run ROM.BAT, it renames CONFIG.SYS to CONFIG.TMP then reboots the computer. As you'll recall, if there is no CONFIG.SYS file, the computer boots from the System Manager. DOS.BAT, on the other hand, renames CONFIG.TMP back to CONFIG.SYS then it, too, boots the computer. When CONFIG.SYS includes the SHELL statement, the shell program executes AUTOEXEC.BAT.

Note

These batch programs assume that REBOOT.COM is in the root directory. Since it's not a program you'll want to run by accident, you'll probably want to move it to a subdirectory; perhaps C:\BIN\ or C:\UTIL\. It's always a good idea to keep the root directory clean, and especially free of potentially dangerous programs.

Notes About Elbow Room

- The SHELL statement in CONFIG.SYS allows you to break-out of the built-in applications, turning the HP 95 into a plain-DOS computer.
- You can't type EXIT to return to return to the built-in applications when you boot from plain DOS. You must use the $SYSMGR command.
- If you get "stuck" in DOS, relax! You can return to the built-in applications by typing $SYSMGR. Or you can change the name of CONFIG.SYS, then press **CTRL**+**ALT**+**DEL** to reboot.

TSR Programs on The HP 95

TSR simply means Terminate and Stay Resident. After TSR programs are loaded into memory, they disable themselves and return to DOS without releasing the memory they occupy. They sit, waiting for an event such as a key press or a character waiting at the serial port or maybe a clock tick, then they come alive, do their business, then fall dormant again until the next event. This can have several connotations from a simple keyboard patch to full-blown, pop-up applications.

Most TSR programs are of limited use to the HP 95: they frequently duplicate features built-into the computer. This is understandable, since most of the 95's built-in programs were based on a TSR program: Lotus Metro. The most beneficial TSRs are *Device drivers.* These programs add new features or extend the functionality of the computer. Practical TSRs include mouse drivers, Dvorak keyboard translators, and speech synthesis.

Three Ways to Load TSRs

1. Create a CONFIG.SYS file with a SHELL statement. Load the TSRs from AUTOEXEC.BAT and add a $SYSMGR command to the file, then reboot the computer.
2. After booting, switch to the Filer and execute the programs.
3. Another type of TSR is the device driver; these program files are different from the conventional TSR. Device drivers (usually with the SYS file name extension) are loaded from CONFIG.SYS, using a DEVICE= line.

Comparing Device Drivers and TSRs

Device drivers are the simplest, most automatic and most memory efficient TSRs (especially when we use them with the HP 95). Since we load device drivers before the environment, built-in applications, and in fact most of DOS itself, they are, arguably, the most polite and reliable type of TSR. The only real limitation is that you cannot unload device drivers without rebooting the computer. Unfortunately, very few TSRs are available as device drivers.

We'll use WINK, a cursor enhancer program included in the *HP 95 Utility Pack,* in the following examples because it was written for the HP 95, and it's available as both a typical TSR (named WINK.COM) and a device driver (WINK.SYS). WINK sits quietly in the background, ensuring that the cursor is always a large, visible, blinking block (never a skinny little underline) when you want to see it, yet allowing programs to turn the cursor off if they want to.

Memory required by TSRs depends on how and when the programs are loaded as much as how large the programs really are. The device driver version of WINK requires under 200 bytes of memory, while the TSR can require from 400 bytes to over 2K, depending on how the HP 95 allocates memory for it. This table compares TSR memory requirements with HP 95 main memory partitioned into a RAM disk of 254K and System RAM at 258K:

TSR Memory Requirement Comparison

	Bytes	**Memory Cost**
Total Memory	264,192	·
Shell	163,760	100,432
With WINK.SYS	163,616	100,432+144
With WINK.COM	161,440	100,432+2,320

What's wrong with this picture? While WINK.SYS and WINK.COM have similar memory requirements after loading, loading any TSR will add extra memory for DOS data that cannot be reclaimed. WINK.COM returns what memory it can to DOS, but it cannot recover everything, and a small hole in memory (the size of allocated environment variables) will be created.

Installing TSRs

There are a couple ways to load the TSR version of WINK. The simplest way is to execute it from Filer, though that's a bit of a nuisance if you reboot often.

You can also load TSRs automatically from AUTOEXEC.BAT. This is the preferred method because less memory will be wasted and there is less likelihood of incompatibilities. You'll need a SHELL statement in CONFIG.SYS before the computer will even recognize AUTOEXEC.BAT:

```
SHELL=COMMAND /P
```

Now, load the program from AUTOEXEC.BAT, then return to ROM-based applications by executing the $SYSMGR statement. Here's what AUTOEXEC.BAT might look like:

```
WINK
$SYSMGR
```

One way to minimize memory holes and fragmentation is to load TSRs before assigning any environment variables. A copy of the environment is created when every program runs; most TSRs have little need for the environment after they load, so they return the memory to DOS before they install themselves. This environment block exists in memory below the TSR itself, and it's usually quite small, so it will likely be left unused.

Borland's SideKick is one of the most popular TSR programs of all time. Version 1.56B (circa 1986) *can* run on the HP 95 with only a few hitches (and occasional crashes — this is the only program the author has found that hangs the computer). You will have to use **CTRL+ALT** as the hot key, not **LEFTSHIFT+RIGHTSHIFT**, because the HP 95 appears to have only a left shift key (internally, both shift keys are treated the same).

Usually you can unload SideKick by pressing **CTRL+HOME+END**; it might take a few tries to unload SideKick on the HP 95, and you'll have to also press the **SHIFT** key (for a total combination of **CTRL+SHIFT+HOME+END**).

Note

Loading TSRs by loading COMMAND.COM via CONFIG.SYS (the SHELL=COMMAND /P command) then adding the $SYSMGR statement to AUTOEXEC.BAT will cost at least 2K for the portion of COMMAND.COM that remains resident after loading $SYSMGR, regardless of how small the TSR program really is. The minimum COMMAND.COM will reserve is approximately 1.8K for itself.

Loading Device Drivers

Very few programs can load as a conventional TSR or device driver (they usually have the EXE file extension). Most, though, load just one way — using a separate program file if there are two versions. Device drivers load automatically when you reboot the computer, with a single line in CONFIG.SYS. The device driver version of WINK, for example, loads with this line:

```
DEVICE=WINK.SYS
```

Note

Most files that load device drivers end with the SYS extension. A few, however, are named BIN, and just a handful use the EXE extension (combining a DOS program and a device driver in the same file). Drivers are loaded from CONFIG.SYS.

While conventional TSRs leave wasteful memory holes, device drivers don't waste any memory at all. When the device loads, there is no environment — in fact, much of DOS hasn't been loaded, and none of the HP 95's built-in applications have run. Device drivers are, generally speaking, more sophisticated than garden-variety TSRs.

See also

There's more information about device drivers on page 67.

Notes About TSRs

- Whenever you reboot your computer, you will have to reload your TSRs.
- You can load TSR programs in AUTOEXEC.BAT, then add the $SYSMGR command at the end of the file.
- TSRs may cut available memory so that built-in programs, even Lotus 1–2–3, may not be able to load.
- Assign environment variables after loading TSRs. This will save memory, and will minimize memory fragmentation.

Owner's Manual Addendum

The HP 95 has a host of features and built-in programs — just look at the Owner's Manual. But as you've already seen, there's more to this computer than you'll find in the manual. There's a hidden agenda: several new, powerful programs, and this chapter explains how to use them.

CHKDSK.EXE

The name CHKDSK is a contraction of "Check disk." This is a conservative name, it can also display available memory, list fragmented files and fix broken disks. It inspects and can correct the logical structure of your disk. CHKDSK can help keep your disk drive in order and your files safe. Select CHKDSK.EXE from Filer, or run it from the DOS command line:

```
CHKDSK
```

You can optionally include a drive letter on the command line; if you don't, or you run the program from Filer, CHKDSK will look at the current disk drive (usually C: on the HP 95). CHKDSK's report looks something like:

```
777216 bytes total disk space
267776 bytes in 7 hidden files
  2048 bytes in 2 directories
343552 bytes in 62 user files
163840 bytes available on disk

264192 bytes total memory
164672 bytes free
```

This short report tells use more than just the condition of the disk drive. Notice the 7 hidden files and one more directory than there should be. Notice also that in a computer that ostensibly has 512K of memory, CHKDSK reports 777,216 bytes of disk space plus 264,192 bytes of memory. That's about a megabyte, and it still doesn't account for some data in ROM!

Most MS–DOS bootable disks have two hidden files: MSDOS.SYS and IO.SYS (some DOS versions call these files IBMDOS.COM and IBMIO.COM); if there's a volume label, it also counts as a hidden file. But on the HP 95, built-in hidden files are quite different. DOS executes directly from ROM, so there's no need for the boot files. Still, there are seven hidden files, with the name ROMDUMMY, and extensions from 009 through 015 (ROMDUMMY.015, for example).

Correcting Disk Errors

When a program goes haywire or you reboot the computer while files are open, some data may not be written to files. CHKDSK can inspect and correct logical errors, not physical defects. The HP 95 does not have a physical disk drive. That's logical.

Caution

Run CHKDSK first without the /F parameter, then decide what action to take next. If CHKDSK reports no errors, there is no reason to run CHKDSK again with the /F switch — this isn't saying the disk always has a clean bill of health, just that if there are problems, they are beyond the power of CHKDSK.

```
CHKDSK /F
```

After CHKDSK corrects all disk problems within its power, you may find files in the root directory with the name FILExxxx.CHK, where xxxx represents numbers, beginning with

FILE0000.CHK. These files are the lost clusters from partially saved files, most often they contain garbage. However, before deleting these files, have a look at them with Filer.

Notes About CHKDSK

- If CHKDSK reports fragmented files on the C: disk, you can compress the disk. Since the HP 95 uses a RAM disk, file fragmentation doesn't slow disk access appreciably.
- Since the HP 95 stores files in units of 512 bytes, files below 512 bytes cannot become fragmented.
- Always run CHKDSK without the /F switch first. If disk errors are reported, make backup copies of important files before running it again with the /F switch.

Checking File Fragmentation

As we work, we're constantly writing and rewriting the same files, while other files come and go in an instant. MS–DOS handles this potentially wasteful process by allocating just what disk space is necessary; if a file won't fit in one contiguous block, it will extend to the next available block. As time goes on, bits of files can become scattered all over the disk. This is what is known as fragmentation

Disk fragmentation is a concern with floppy or hard disk-based computers because it directly affects disk performance. The more disk drive heads have to move (called *seek)* to find pieces of files, the slower the disk drive seems to run.

The /V switch, which stands for Verbose, tells CHKDSK to display file names as it verifies the disk. This report will also list fragmented files. You can optionally specify a file name, with wild cards if you like. Here's the command line:

```
CHKDSK *.COM
```

CHKDSK lets us include wild cards, such as *.COM, or just /V, or both together, and the list will include hidden files, but the file specification cannot include hidden files (we can't use CHKDSK ROMDUMMY.* to see the hidden files). The results displayed by CHKDSK will be something like:

```
C:\95.COM
   Contains 2 non-contiguous blocks.
C:\SK.COM
   Contains 4 non-contiguous blocks.
```

Packing the C: Drive

Once CHKDSK has verified that you have several fragmented files (and you've decided that this is a problem — it probably isn't with your HP 95 RAM disk), you can pack all files to the beginning of the disk. This will leave free disk space in one contiguous block, and minimize new file fragmentation.

On your desktop PC, you might de-fragment the disk with a program like Norton Utilities or PC Tools. The HP 95 has a disk packer built-in! First delete any unnecessary files, then run CHKDSK with the /F switch to ensure the disk file chain is in good shape. Next, type EXIT to return to the ROM-based applications then press **SHIFT**+**SETUP**. Now select System then Memory. Make a note listing the current RAM disk and System RAM settings, then change the RAM disk size to the minimum setting then press **ENTER**. The computer will display a dialog box with the following message:

```
Reboot of the system needed
Press ENTER to reboot, ESC to cancel
```

Press Enter to reboot the computer. Now return to Setup System Memory and restore the original RAM disk size setting.

Defragmenting the C: Drive

If you have a RAM card or another PC, you can de-fragment the C: drive, making all files contiguous. First copy all important files to the other computer. Now you can delete the files

from C:, or reformat the C: drive; the latter will be faster, but it will cause the computer to run SYSINIT.BAT — copying some files you may not want from C:_SYS\ to C:_DAT\.

You can then copy all important files back to the C: drive. To minimize re-fragmenting later, first copy files that are less likely to change — such as programs — and files that are under 512 bytes to the C: drive.

DEBUG.EXE

DEBUG is a memory browser, a simple assembler, and even a hex calculator. It has survived virtually unchanged through every version of DOS. There is a copy of DEBUG.EXE in the hidden C:_SYS\ directory. It's surely hidden because a few commands can be dangerous, but more likely because it's only occasionally useful.

We won't explain everything about DEBUG here because if you are serious about assembly language, you will want to use a *real* assembler. While it's powerful, fast and efficient, make no mistake about it: DEBUG is a lousy assembler. Calculating instruction lengths for JMPs (and re-booting when you get the numbers wrong) is counter-productive.

DEBUG is useful chiefly for inspecting or patching memory locations or programs, and for converting program listings (like those in this book) into programs. Here's how to start DEBUG from the hidden directory:

```
C:\_SYS\DEBUG
```

You could also use the Filer Goto command to switch to the C:_SYS\ directory then select DEBUG.EXE from the list of files. When DEBUG starts, the prompt changes to a minus sign followed by the flashing cursor:

```
-_
```

Caution

Now *immediately* press **Q** (which stands for Quit) then press **ENTER** to return to DOS. All of DEBUG's commands are a single character, like Q. But until you know the commands (and sometimes after), just trying things can cause serious damage!

See also

We've included a table of DEBUG commands on page 109. Please browse to this table before experimenting.

DEBUG's *Dump* command, abbreviated as D, is great fun. It lets us look at any location in memory, and displays the information as boring hex listings, but also as text — anything that looks even remotely like text is displayed. We can look at copyright notices, error codes, and notes in programs (not remarks, because compilers delete them when they create programs). Anything is fair game, so let's look at low memory at segment zero and offset 413 (hex):

```
D 0:413           ; dump segment 0, offset 413
```

The beginning of this address looks something like:

```
0000:0410           0A 01 00 00 00
```

The example shows a screen-full of information, but the first two bytes are total memory size (byte reversed) in hex. 10A hex is 266 in decimal, so the computer has 266K of memory.

Leading zeros in memory addresses are optional, and if you use only a single number, DEBUG assumes that you're looking at another offset within the same segment. You can continue displaying from the last address by entering D without any address.

You can use DEBUG to see the current date, in decimal format. We're adding a second command on the line: L is the number of bytes to dump

```
D 0:4B8 L 7       ; dump segment 0, offset 4B8, and 7 bytes
```

The L command tells DEBUG to display only seven bytes. If today is December 25, 1992, DEBUG will show something like:

```
0000:04B0                          19 92 12 25 21 05 00
```

The year is the first two bytes, followed by the month then the day. The last three values are the current time in twenty-four hour format.

You can see the BIOS release date. It's an 8-character text string at address F000:FFF5. To keep the amount of typing down, we've calculated the same address as FFFF:0005:

```
D FFFF:5 L 8
```

This time you will have to press **SHIFT+ALT+RIGHT** to see the text:

```
FFFF:0000                    30 34 2F-30 32 2F 39 31    04/02/91
```

Another way to limit the amount of data DEBUG will display is by specifying an ending address after the start address. This example shows the release date, just as above, but instead of specifying 8 bytes with the L command, we are specifying the ending address of the range. The starting offset plus the ending offset in hex (5+8 here) is D:

```
D FFFF:5 D
```

DEBUG's Hexadecimal Calculator

The H command performs 16-bit hexadecimal addition. This is a nice addition to the HP 95 because, as powerful as the HP Calc calculator is, it doesn't do hexadecimal math.

Type the letter H followed by a space then two hexadecimal numbers and DEBUG will display the sum of the two values and the difference between the two numbers.

```
H 1C 1234                  ; add two hexadecimal values
```

And the answer returned by DEBUG:

```
1250  EDE8
```

If the second number is larger than the first (as it is above) the difference is represented in two's complement notation. Here's a simple calculator batch (called H.BAT) program that passes two hex values to DEBUG. It creates a DEBUG script, passes control to the program, then deletes the script when it's through:

```
ECHO OFF
ECHO H %1 %2 > C:\TEMP.SCR
ECHO Q >> C:\TEMP.SCR
C:\_SYS\DEBUG < C:\TEMP.SCR
DEL C:\TEMP.SCR
```

To use the program, exit to DOS then type the program name followed by two hex numbers:

```
H 1234 5678
```

The computer will display:

```
-H 1234 5678
68AC BBBC
-Q
```

Calculating Memory Addresses

While it's a 16-bit calculator, you can use it to figure 20-bit memory addresses. The key is tor keep in mind that the calculation is on the upper 16 bits, the lowest digit will not be changed.

See also

For a refresher on memory addressing, refer to page 15.

Consider the address 0040:00F8, which is the location in low-memory where the key code from the last key pressed is stored. We'll start by adding the segment to the three high digits from the offset:

0	0	4	0	Start with a 16-bit segment.
	0	0	F	Add the high 3-digits from the offset.
0	0	4	F	The high-word of the physical address.

In DEBUG, we would enter the formula like:

```
H 40 F
```

DEBUG will display the result as:

```
004F  0031
```

The result is 004F. Now we'll shift the result over 4-bits and add the remaining digit from the offset. Ok, we'll really just tacked the remaining digit from the offset on the right. The DEBUG calculator stops at 16-bits, so we'll have to do this one by hand:

0	0	4	F		The high-word from the calculated address.
				8	Add the remaining digit.
0	0	4	F	8	Which results in a 20-bit physical address.

Let's verify that the calculated segment and offset (004F:0008) work the same as the original address (0040:00F8). That address contains the address of the last key pressed, so when we press **ENTER**, the result should be 1C, the scan code for **ENTER**.

```
D 4F:8 8
```

If you press Enter, DEBUG will return:

```
32
```

Press F3 to recall the command line then instead of **ENTER**, press **CTRL**+**M**; DEBUG will respond with:

```
1C
```

There's a third way to verify the keycode: Type a character code 13 on the number pad. Hold down **ALT** then press the **1** then the **3** key. When you release the **3** key, DEBUG will show:

```
B8
```

Instead of **M** or **ENTER**, that's the scan code for the **3** key, the last key you pressed.

To clear this up a bit (and possibly make things murkier in the process), consider interrupt vectors. The interrupt vector table look-up table is stored at the low-end of memory, beginning at address 0000:0000. There are 256 possible interrupts, numbered 0-255. Each vector consists of a two-byte segment and two byte offset.

Interrupt zero, at the beginning of the table, is invoked whenever a division by zero error occurs; most programs will change this interrupt to their own interrupt handler when they run, then restore the original when they exit.

This DEBUG command will display the first four bytes of memory (the address of interrupt vector zero):

```
D 0:0 3
```

Assuming the The HP 95 will display the four bytes at the beginning of memory.

```
0000:0000  4E 4B 2D A3
```

In *little endian* fashion (see *Memory Addresses,* on page 15), the low byte of the offset is first, followed by the high byte of the offset, then the low and high bytes of the segment. On the HP 95, this default interrupt vector is usually A32D:4B4E; unscrambling the code in the example, that's what we have. We can use the address of the interrupt vector to see what the interrupt code looks like; this example uses the DEBUG's *Unassemble* command:

```
U A32D:4B4E
```

As you will see from the un-assembled code, unlike most other interrupt vectors, this one doesn't preserve registers. That's because if a program running in DOS causes a division by zero error, and it hasn't setup its own interrupt handler, DOS will take control and terminate the program with an error message.

Note

Calculating addresses from segments and offsets is usually done in assembly language programs to allow memory operations to extend beyond 64K. After the calculation, the high 16-bits are placed in the segment register and the remainder (16 or less) will be the offset. This is often called *Normalizing segments.* The goal is to minimize the offset and move as much of the pointer as possible to the segment values so that pointers can be incremented farther without constantly fiddling with the segment.

Search Command

We could use the D command repeatedly to locate a special message in memory, but the Search command does that for us automatically. The search command is case sensitive; Upper and lower case characters are recognized as different characters. It may take a couple of tries before you find the text. Usually the first letter of a word is upper case and the rest is lower case, but not always. You can start the search at the second letter to make the search easier. This example will search the ROM BIOS for the word "Copyright" with just a bit of work:

```
S F000:0 FFFF "opyr"
```

DEBUG will display the address of each match. On the HP 95, it will look like:

```
F000:CD8A
F000:CDCD
F000:E021
```

The numbers returned are starting addresses, the address of the first character of matching strings. You can easily check the addresses, and see the text. We'll back-up the address by one byte to account for the missing "C" from "Copyright."

```
D F000:CD89       ; display the copyright notice
```

Creating DEBUG Scripts

Working with DEBUG is error prone at the best of times; typing in long program listings is the worst of times. DEBUG's primitive line editor (the same as DOS' editor) is fine for exploring, but for larger projects, it's easier to use the 95's Memo program to create DEBUG Scripts.

A script is simply the same keystrokes you would type on the command line, but entered into a file. The obvious advantage is that you can go back and check your work, and even if DEBUG kicks it out with an error message, you can fix a line or two in the file — instead of typing every line again. There are two general forms of DEBUG scripts: assembly language listings (like CAPSON, listed below), and hex code listings (HELLO.TXT, in the next section). For shorter programs, assembly language mode is easier. With larger programs, hex codes are usually easier to enter, and they take less room on the printed page. If you want to distribute printed programs without source code, of course you'll use hex codes.

Note

We've included some small, deceptively simple utility programs written in assembly language. Our programs have comments on the right of many lines, following a semicolon (in Intel 808*x* assembly language, comments begin with a semicolon). Don't enter the comments in DEBUG, they are just for your information. When a comment says "; a blank line", press **ENTER** without typing any data; this will DEBUG's exit assembly mode.

When you enter these programs, keep in mind that all values are in hexadecimal. If you incorporate any of these routines in your own assembly language programs, be sure to add the assembly language hex radix specifier at the end of each number, in the format "1234H".

Here's a look at using DEBUG to enter hexadecimal data. Start DEBUG and type the following six lines:

```
N HELLO.TXT
E 100 48,65,6C,6C,6F,2C,20,77,6F,72,6C,64,21,0D,0A
RCX
E
W
Q
```

You're back in DOS with a new file named HELLO.TXT. Take a look at the file:

```
TYPE HELLO.TXT
```

It's a plain-old text file, not some string of mystical codes! The N command told DEBUG the name of the output file. The E command entered the data on the second line. RCX recalled the value from CX. Next, A entered assembly mode. The DB assembler directive means *Define Byte* or *Data Byte,* (depending on who you ask). Following on that line are a series of hexadecimal numbers. We cheated a bit.

We could have used assembly mode, and entered a quoted string and a carriage return then a linefeed character (0D,0A), marking the end of the text line.

```
N HELLO.TXT                 ; the output file name
A                           ; enter assembly mode
DB "Hello, world!",D,A
```

Now press **ENTER** once again to exit assembly mode (in other listings from now on, we'll show a blank line with a remark on the right). Then enter the next four lines:

```
RCX                         ; recall CX register
E                           ; number of bytes (in hex) to write
W                           ; write bytes to the file
Q                           ; quit DEBUG, return to DOS
```

Pressing **ENTER** on a blank line tells DEBUG to exit assembly mode. RCX recalls the contents of CX, the E is the length of the text (including the end of line characters). The W command says to write the information to the file. And finally, Q returns us to DOS.

Here's a quick little program that turns on the **CAPS** lock key and demonstrates DEBUG's assembly mode. You could call the finished program from AUTOEXEC.BAT to set your computer in upper case mode each time you boot:

```
N CAPSON.COM                ; the program file name
A                           ; enter DEBUG assembly mode
XOR     AX,AX               ; point data segment at low memory
MOV     DS,AX
OR      BYTE PTR [417],40   ; set the caps lock bit
INT     20                  ; quit the program
                            ; a blank line exits assembly mode
RCX                         ; recall CX register
B                           ; the program is B bytes long
W                           ; write data to the file
Q                           ; quit DEBUG, return to DOS
```

If you'd like to use CAPSON.COM, you could enter the code directly in DEBUG, or write a script file then redirect the file to DEBUG. By convention, script files have the same name as the finished program, but the SCR file name extension. The script file for CAPSON.COM is CAPSON.SCR. Once you've entered the text in the file, exit to DOS and redirect the script through DEBUG:

```
C:\_SYS\DEBUG < CAPSON.SCR
```

This says to DOS: "Send the file CAPSON.SCR through DEBUG." DEBUG will display each line as it works. When it's done, and DEBUG hasn't scolded you with errors or warnings, you'll have a file named CAPSON.COM. Once you've tested the program, you can discard CAPSON.SCR.

Notes About DEBUG

- When you edit DEBUG scripts with the HP 95 Memo program, set word wrap to at least characters. This will keep the editor from splitting lines.
- Press **ENTER** after each line. With Memo, make sure each script file line ends with the diamond-shaped end of line character.
- When you're sure finished programs run properly you can discard script files.
- To DEBUG, or DOS, or us, for that matter, a byte is a byte. Depending on where it is, a byte may be text or data or program code, yet it's really all the same inside.
- All numbers used by DEBUG are hexadecimal.

SERINT.COM

This little program enables (/E switch) or disables (/D) the serial port interrupt. Disabling the serial port can conserve battery power.

```
C:\_SYS\SERINT
```

Running SERINT without any command line argument, will cause the program to display this message:

```
Usage: SERINT { /E | /D }
```

SYSINIT.BAT

This batch file copies TOPCARD.PCX, WTIME.DAT, and all files with ENV, ?BK, DCF, LCF, CTF, CCF, FCF extensions from the hidden C:_SYS\ directory to C:_DAT\.

```
C:\_SYS\SYSINIT
```

SYSINIT is run automatically when the computer does a cold boot, after initializing the RAM disk.

Note

Since there are always spare copies of seldom-needed files like MCI.DCF, even TOPCARD.PCX, in C:_SYS\, you can delete these files from C:_DAT\, then copy them back again with SYSINIT when you need fresh copies.

Sample Programs

MEM.COM and REBOOT.COM are also available in the software package named *HP 95 Utility Pack*. BATTERY.COM is available only here, but a similar feature is included in program 95.COM, also in the Utility Pack.

Caution

Most of these programs are designed to run only the HP 95. Running them on other computers will cause unpredictable, even disastrous, results.

See also

The DEBUG section of this book (see page 91) explains how to enter these DEBUG scripts. Another sample program, named CAPSON.COM, is listed on page 92.

The HP 95 Memo program is a fine place to type DEBUG script files. Each line is less than 40 characters; always press **ENTER** at the end of each line, don't depend on word-wrap. The "0" in these script files represents the number zero, not "O" (oh). The "1" character is the number one. When we need the letter oh, we'll show it in lower case.

Note

While copyrights on BATTERY.COM, ECHO2.COM and REBOOT.COM are retained by the author, you may re-distribute those three programs for any non-commercial purpose as long as no fee is charged. It would be nice, though, if you mentioned where you got them.

BATTERY.COM

This program displays voltages for the HP 95's main and backup batteries. Run BATTERY.COM by selecting the file name from Filer or with this DOS command:

```
BATTERY
```

New batteries report about 3.0 volts. When system batteries dip below 2 volts, the computer displays "Low Main Battery" each time you power-up. When the batteries drop to 1.8 volts, the computer shuts-down until you replace the batteries. You can't just wait for tired batteries to recover a bit then try again, because a fail-safe mechanism requires the batteries to recover to 2.5 volts (very unlikely) before the computer will return to normal. This sample report shows the main battery just marginal at 2.1 volts and the backup battery at full-steam (3 volts):

```
System battery: 2.1V, backup: 3.0V
```

The computer displays "Low Backup Battery" when the backup battery drops below 2.5 volts.

BATTERY.SCR

```
N battery.com
E 100 B4 30 CD 21 3D 03 16 75 21 BF
E 10A 56 01 B8 01 50 E8 1D 00 88 65
E 114 1E 88 45 20 B8 00 50 E8 11 00
E 11E 88 65 10 88 45 12 8B D7 B4 09
E 128 CD 21 B8 00 4C CD 21 CD 15 83
E 132 F8 FF 74 1C 25 FF 00 74 17 B2
E 13C 05 F6 F2 48 3C 19 73 08 FE C4
E 146 3C 14 73 02 FE C4 D4 0A 05 30
E 150 30 C3 B8 3F 3F C3 53 79 73 74
```

```
E 15A 65 6D 20 62 61 74 74 65 72 79
E 164 3A 20 3F 2E 3F 56 2C 20 62 61
E 16E 63 6B 75 70 3A 20 3F 2E 3F 56
E 178 0D 0A 24
RCX
7B
W
Q
```

CURSIZE.COM

This program changes the cursor size for DOS-based applications. The cursor will return to the standard size when you exit from DOS. The cursor is always at the bottom of a 6x8 pixel character cell. The top of the cell is scan line 0 (zero), the bottom is scan line 7 (seven). By changing the starting scan line, you can change the cursor shape. Starting at line 0 makes a solid blinking block, while starting at 7 would make a single skinny line (the cursor would start and end on line 7). The listing sets a solid block cursor:

CURSIZE.SCR

```
N cursize.com              ; the program file name
A                          ; enter assembly mode
mov     CH,0               ; starting scan line for the cursor
mov     CL,7
mov     AH,1               ; function 1 (set cursor size)
int     10                 ; call the video BIOS
int     20                 ; quit the program
                           ; a blank line
RCX                        ; recall CX value
A                          ; length of the program file (in hex)
W                          ; write the file
Q                          ; quit DEBUG, return to DOS
```

You can create several custom versions of the program, saving each under a different name, for custom cursor sizes. To customize the program, change the first line to change the cursor size. Replacing the line with the following will start the cursor on line 3:

```
mov     CH,3               ; starting scan line for the cursor
```

Experiment with several variations. Keep in mind that DEBUG numbers are in hexadecimal, and numbers from 20 (32 decimal) on up to FF (255 decimal) will turn the cursor off.

ECHO2.COM

If results from your batch file are redirected to a file, ECHO, too is redirected. There's no way to warn the user about disk changes or impending calamities. ECHO2 works like ECHO — except it ignores output redirection, and will always send the message to the screen.

```
ECHO2 Hello, world! >NUL
```

This example would display nothing at all with ECHO. However replacing ECHO with ECHO2 ensures that the text goes to the display. Page 72 demonstrates ECHO2.COM.

ECHO2.SCR

```
N echo2.com
E 100 BE 80 00 AC 46 8B D6 98 03 F0
E 10A 91 41 C6 04 0A BB 02 00 B4 40
E 114 CD 21 CD 20
RCX
18
W
Q
```

MEM.COM

MEM displays available system memory, excluding memory used as a RAM disk. Use MEM instead of CHKDSK.EXE when you just want to see memory information without information about the disk drive. Here's a typical MEM report:

```
  1,024 K EMS memory
    768 K EMS available
458,752 bytes main memory
 86,016 bytes used
372,736 bytes available
```

MEM displays main (conventional) memory and EMS memory (bank-switched, Enhanced Memory Specification). If no EMS is available, the report will include only conventional memory. The HP 95LX currently supports only conventional memory — but we're prepared.

MEM returns an error level representing memory. Less than 100K available is returned as 0, less than 200K is 1, and so forth.

MEM.SCR

```
N mem.com
E 100 B8 67 35 CD 21 0E 1F BE CB 01
E 10A BF 0A 00 B9 04 00 F3 A7 75 3D
E 114 B4 40 CD 67 22 E4 75 35 B4 42
E 11E CD 67 22 E4 75 2D 53 8B C2 33
E 128 D2 B9 04 00 D1 E0 D1 D2 E2 FA
E 132 BF DE 01 E8 66 00 58 33 D2 B9
E 13C 04 00 D1 E0 D1 D2 E2 FA BF F9
E 146 01 E8 54 00 BA D3 01 B4 09 CD
E 150 21 CD 12 50 99 B9 00 04 F7 E1
E 15A BF 18 02 E8 3E 00 BB FF FF B4
E 164 4A CD 21 8B C3 B9 70 17 F7 F1
E 16E A2 C9 01 99 58 B9 40 00 F7 E1
E 178 53 2B C3 B9 10 00 F7 E1 BF 38
E 182 02 E8 18 00 58 B9 10 00 F7 E1
E 18C BF 51 02 E8 0C 00 BA 0D 02 B4
E 196 09 CD 21 A1 C9 01 CD 21 92 1E
E 1A0 07 FD BB 0A 00 33 ED 45 83 FD
E 1AA 03 76 07 C6 05 2C 4F BD 01 00
E 1B4 8B CA 33 D2 F7 F3 91 F7 F3 92
E 1BE 04 30 AA 8B C1 0B CA 75 E0 FC
E 1C8 C3 00 4C 45 4D 4D 58 58 58 58
E 1D2 30 20 20 20 20 20 20 20 20 20
E 1DC 20 20 20 20 4B 20 45 4D 53 20
E 1E6 6D 65 6D 6F 72 79 0D 0A 20 20
E 1F0 20 20 20 20 20 20 20 20 20 20
E 1FA 20 4B 20 45 4D 53 20 61 76 61
E 204 69 6C 61 62 6C 65 0D 0A 24 20
E 20E 20 20 20 20 20 20 20 20 20 20
E 218 20 20 62 79 74 65 73 20 6D 61
E 222 69 6E 20 6D 65 6D 6F 72 79 0D
E 22C 0A 20 20 20 20 20 20 20 20 20
E 236 20 20 20 20 62 79 74 65 73 20
E 240 75 73 65 64 0D 0A 20 20 20 20
E 24A 20 20 20 20 20 20 20 20 20 62
E 254 79 74 65 73 20 61 76 61 69 6C
E 25E 61 62 6C 65 0D 0A 24 28 63 29
```

Program listings continue on the next page…

```
E 268 31 39 39 31 20 52 69 63 68 61
E 272 72 64 20 45 2E 20 48 61 72 76
E 27C 65 79 0D 0A
RCX
180
W
Q
```

NOW.COM

This program displays the current time and date. It is usually used with output redirection for logging system or program activity. Enter the command or select the program name from Filer:

```
NOW
```

The program will display the time and date in the format:

```
20:45:00, Sep 04-92
```

You can redirect the result to a file to time-stamp your work, or just show the time every now and again. Be sure to use >> to append the result to the end of the log file; if you accidentally use a single > character, it will overwrite the file.

```
NOW >> WORK.LOG
```

NOW.SCR

```
N now.com
E 100 BF 83 01 1E 07 B4 2C CD 21 8A
E 10A C5 E8 44 00 8A C1 E8 3F 00 8A
E 114 C6 E8 3A 00 47 B4 2A CD 21 BE
E 11E 5C 01 8A C6 98 03 F0 D1 E0 03
E 128 F0 A5 A4 47 8A C2 E8 21 00 91
E 132 2D 6C 07 83 F8 63 76 03 83 E8
E 13C 64 E8 12 00 BA 83 01 B9 15 00
E 146 BB 01 00 B4 40 CD 21 B8 00 4C
E 150 CD 21 D4 0A 05 30 30 86 C4 AB
E 15A 47 C3 3F 3F 3F 4A 61 6E 46 65
E 164 62 4D 61 72 41 70 72 4D 61 79
E 16E 4A 75 6E 4A 75 6C 41 75 67 53
E 178 65 70 4F 63 74 4E 6F 76 44 65
E 182 63 32 30 3A 34 35 3A 30 30 2C
E 18C 20 53 65 70 20 20 34 2D 39 31
E 196 0D 0A
RCX
98
W
Q
```

REBOOT.COM

This program resets the computer, much like pressing **CTRL**+**ALT**+**DEL**. It does not affect the contents of RAM disks. When you execute REBOOT, the computer immediately resets — no confirmation, no protection.

```
REBOOT
```

REBOOT.SCR

```
N reboot.com
E 100 B8 FF FF 50 40 50 8E D8 C7 06
E 10A 72 04 34 12 CB
RCX
F
W
Q
```

SLEEP.COM

This little program turns the computer off. When you press the **ON** key, the computer continues from exactly where it left off. If you are running a batch file, the next line in the file will be read.

SLEEP.SCR

```
N sleep.com                ; the program file name
A                          ; enter assembly mode
mov     AH,42              ; function42h (deep sleep)
int     15                 ; call the HP 95 BIOS
int     20                 ; quit the program with interrupt two-zero
                           ; a blank line
RCX                        ; recall CX value
6                          ; length of the program file
W                          ; write the file
Q                          ; quit DEBUG, return to DOS
```

TIMEOUT.COM

The HP 95 is fast and powerful, but given the opportunity — like nothing to do for awhile — it'll time-out (turn off the display and go to deep sleep). The default timeout is about five minutes. With this program, you can change the timeout value from five minutes to any length you would like, up to about an hour.

As furnished, TIMEOUT.SCR sets the timeout to two minutes, a good compromise between shutting down too fast and wasting batteries.

TIMEOUT.SCR

```
N timeout.com              ; the program file name
A                          ; enter assembly mode
mov     BX,888             ; timeout value (2 minutes)
mov     AH,46              ; function 46h
int     15                 ; call the HP 95 BIOS
int     20                 ; quit the program
                           ; a blank line
RCX                        ; recall CX value
9                          ; length of the program file
W                          ; write the file
Q                          ; quit DEBUG, return to DOS
```

The timeout is calculated in clock ticks of approximately 18ths of a second (more accurately, 18.2 ticks per second). This was calculated as 120*18.2=2184, and converted to hexadecimal. Some common timeout values are shown in the table below. On the left are number of minutes; the right columns list number of clock ticks. Replace the value loaded to BX with a number from the right column. For instance, look down the left column for three minutes; move to the next column in the table to find CCC, then change the code to MOV BX,CCC.

Timeout Hex Conversions

Minutes	Hex	Minutes	Hex	Minutes	Hex
1	444	5	1554	12	3330
2	888	8	2220	15	3FFC
3	CCC	10	2AA8	20	5550

Caution

Don't set the timeout value to very short times or your computer will nod-off in the middle of everything, making it difficult to get any work done. A minimum of a minute (444 ticks) is probably the lowest reasonable setting. The maximum allowable setting is FFFF clock ticks (about an hour). A timeout setting of zero, instead of shutting down immediately, disables timeout. Use zero with caution to avoid accidentally running down the battery.

The timeout setting will return to five minutes when you reboot your computer. For an easier way to customize the timeout, try 95.COM, include with *HP 95 Utility Pack*.

WAITKEY.COM

This PAUSE replacement ignores “garbage” keystrokes, interprets turns function keys into numbers 0-9 and upper cases letters. The codes are returned as an ERRORLEVEL.

```
ECHO Press A,B or C
WAITKEY
```

WAITKEY.SCR

```
N waitkey.com
E 100 CD 28 B4 00 CD 16 3C 1B 74 22
E 10A 3C 20 74 1E 3C 30 73 10 8A C4
E 114 2C 0A 3C 31 72 E6 3C 3A 77 E2
E 11E 75 0C B0 30 3C 61 72 06 3C 7A
E 128 77 02 2C 20 B4 4C CD 21
RCX
30
W
Q
```

WHEREAMI.COM

Some MS–DOS programs are inconsiderate, they’ll leave the cursor an odd size, strange display colors, or worse yet, they may leave us logged-onto a directory where we never go. WHEREAMI can help with one of these problems by displaying the current drive and path:

```
WHEREAMI
```

On the HP 95, WHEREAMI will often show these two lines.

```
C:
CD \_DAT
```

This looks just like the commands you would enter to return to where you started. You can use WHEREAMI in a batch file to create another batch file using output redirection with those two commands, then run your wandering program, then execute the batch file restoring the path.

```
WHEREAMI > GOHOME.BAT
ANYPROG
GOHOME
```

Now, when the program ANYPROG ends, the batch file will pass control to GOHOME.BAT, which will return you to where you started.

WHEREAMI.SCR

```
N whereami.com
E 100 B4 19 CD 21 BF 2D 01 00 05 BE
E 10A 35 01 B4 47 99 CD 21 72 18 8B
E 114 D7 33 C0 B9 FF FF F2 AE F7 D1
E 11E 41 4F B8 0D 0A AB BB 01 00 B4
E 128 40 CD 21 CD 20 41 3A 0D 0A 43
E 132 44 20 5C
RCX
35
W
Q
```

More Than You Probably Want To Know

In one corner of the HP factory in Corvallis, Oregon, no larger than a three-bedroom home, a string of industrial robots in sealed plastic ticket booths methodically assembles components on 95 motherboards. The classical music-like tempo and grace of the robots sets the pace for the rest of the assembly line. Technicians perform exacting tasks, including finishing and testing the computers. It almost looks easy.

But this isn't the birthplace of the HP 95. Most of the work on the 95, the gestation phase, happened long before it reached the factory. Engineering: that's why it looks easy.

HP 95LX Chronicles

According to the folks at HP, "LX" likely stands for Lotus Expandable, though this is of some debate. A few years ago it would have been named "HP–95 LX" instead of "HP 95LX." Because of changes in how HP's internal network handles product names, we won't see dashes in product names again — at least until HP changes software again.

The HP 95 is, to a greater degree than earlier HP handhelds, the product of a group effort. It's a bi-costal, even multi-national, computer. HP designed the "box" and much of the firmware in Corvallis, the display (though massaged for performance and quality by HP engineers) is from Sharp. Inside, the V20 microprocessor is by NEC (Nippon Electric Company), and HP and Intel created the Hopper chip — the "glue" holding the computer together.

HP customized the Phoenix BIOS. Microsoft provided the operating system, MS–DOS version 3.22. Lotus Development Corporation, of Cambridge, Massachusetts, besides the ubiquitous 1–2–3 spreadsheet, delivered most of the built-in applications. HP is also responsible for the elaborate calculator and the built-in games.

The built-in applications are a combination of new and old. Lotus 1–2–3 is, of course, a variation on the wildly successful spreadsheet program. The Filer, Phone Book, Memo and to some extent the other built-in applications are based on Lotus Metro, a well-integrated Terminate and Stay Resident (TSR) software package developed for PC-compatibles. Metro was itself based on another program: Spotlight, another TSR program that was purchased by Lotus.

As advanced as our 95 is, this isn't the only possible incarnation of the technology. It's an open system — open to reinterpretation as infra-red, touch-screen, speech-quality sound, digital to analog converters are needed.

The V20 Advantage

The Norton Utilities, version 4.5, System Information program (SI.EXE) rates the speed of the HP 95 at 2.5 times the speed of the original IBM PC (HP also unofficially rates the computer at 2.5). This number is often quoted, but it's a bit optimistic. In version 6.01, Norton errs in the other direction, rating the HP 95 at 1.2. The truth lies between 2.5 and 1.2; exactly where depends on your perspective.

The HP 95 easily deceives most performance benchmarks. It's a bit more efficient than its deskbound peers, while at the same time it's constantly taking short catnaps to extend battery life (which can distort some benchmarks). There's a square foot of circuitry on a circuit board much less than half the footprint of the machine (if you're measuring, light travels about a foot in a nanosecond; that's a lot of nanoseconds saved).

The NEC V20 processor, operates at 5.37 MHz instead of the traditional 4.77 MHz. The V20 is an improved, more efficient, 8088-compatible processor. Most comparisons show the V20 to be 12 to 18% faster than the 8088.

Most PC compatibles' memory is *DRAM,* short for Dynamic RAM. Dynamic means power must be applied to RAM periodically (called *refresh)* to maintain memory contents. The HP 95 uses the more expensive *PSRAM* (Static or Pseudo-Static RAM); this eliminates the need for memory refresh to extend battery life. Eliminating memory refresh increases system performance about 8%.

A simple test by the author in compiled BASIC, comparing floating point and long integer calculations, clocks the HP 95 about 40% faster than a standard 4.77 MHz PC–XT. The only satisfactory benchmark is whether the computer can do the job in a reasonable amount of time, or are there interminable delays; the 95 passes this with flying colors.

Benchmarks aside, the increased clock speed accounts for about 13%, the efficiency of the V20 processor is good for another 12%, and SRAM adds another 8%. Other factors like tightly integrated design add a bit more — not quite 250% more, but more than enough to win the eleven-ounce, bantam weight division.

Memory (and Memories)

More famous than the MS–DOS command line is the 640K limit. That's the artificial wall the IBM designers imposed on the original IBM PC, which has survived long beyond the PC to haunt a generation of programmers.

Blame for the 640K limit is constantly (and mostly unfairly) placed on Microsoft; it's more justly placed on the times, and conservative engineers.

It made sense when the PC was being designed: RAM was expensive, and the popular CP/M (Control Program for Microcomputers) computers of the era ran just fine with only 64K. So offering ten times as much memory as current computers didn't seem restrictive at all. After all, the original 4.77 MHz IBM PC shipped with 16K of memory — all for only $1600 (just add an audio cassette player for data storage and a home television set for the display, and you're ready for serious work). It didn't seem like anybody could ever use-up 640K of memory; and certainly nobody could afford to try.

The reason main memory is limited to 640K in PC compatibles is the first ten 64K segments of the memory map are allocated to program usage (occasionally called Transient Program Area, or TPA). The memory addresses immediately following are reserved for the video memory, memory mapped plug-in devices and, at the high-end of the address space, the ROM BIOS — the shared, general purpose control programs that made the limited RAM capacity practical. Even if you can plug-in more RAM chips, the video buffers get in the way; more time and money has been invested circumventing the 640K wall than was spent developing the original PC.

The HP 95 is limited to 512K instead of 640K because memory chips only come in even sizes. The next size up after 512K is 1 megabyte, which wouldn't fit in the address scheme, and without fancy footwork would be largely be wasted.

The HP 95 works around the 512K, 640K and 1MB limits by *bank switching,* a method of letting multiple memory blocks share the same physical memory addresses. By enabling and disabling chips in the upper memory area, 64K blocks of program code and 16K blocks of data can be switched in and out. This scheme is used by the built-in applications, but it's extensible, allowing information on plug-in cards to also occupy physical memory addresses. We can plug in a program card and run the program in place, without loading a copy to RAM, and without wasting any of that precious Southernmost 512K.

See also

For a look at the HP 95 memory map, refer to the table on page 108.

The memory map shows interrupt vector at the lowest memory addresses (which is required by Intel processors), followed by data and DOS. After the Video buffer (at segment B000), where conventional PCs have nothing to do, the HP 95 gets interesting. That's where code and data from XIP programs reside.

Glossary

Many computer terms are snagged from real-life situations. Others are acronyms, which we usually type in all-capital letters.

We won't list every computer term here; some phrases are evolving street lingo. Synonyms (read: euphemisms) for crash include break, *hang* and *lunch;* as in "Lunch the machine." An up-and-coming crash-replacement is *assert.* Synonyms for losing data include *garbage, trash* (as verbs), *corrupt* and *lunch,* again. Synonyms for bug include *anomaly* and *feature.* Serious bugs are often called "Incompletely implemented features."

Address

A location in memory. Memory addresses are usually expressed as segment and offsets, in the form SSSS:OOOO. See Segment and Offset, later in this section.

ASCII (American Standard Code for Information Interchange)

ASCII codes use an 8-bit value to represent characters The Standard ASCII character set uses the first 128 values (0-127). The characters represented by high-order numbers (128-255) are dependent on the computer type; for IBM–PC compatibles, these are called PC–8 or code page 437 (English). The HP 95 uses code page 850 (Latin 1), which is similar for values below 155, and uses international characters.

Binary

Binary, base-2, numbers are the lowest form of life in the computer world. Binary digits are the ones and zeros, on and off states, of computer chips and peripherals. See Binary 101, on page 12.

BIOS (Basic Input Output System)

The BIOS is the pre-programmed ROM memory containing programs and utilities for booting the computer and communicating with hardware. Software calls BIOS routines for most of the hardware-dependent operations, which makes programs less dependent on quirks of individual computers. Page 17 discusses the BIOS' job when booting the computer.

Bit

Not very much. A contraction of *binary digit,* the bit is the smallest unit of storage. Usually combined into 8-bit units called *bytes.*

BOOT

Booting is the process of restarting the computer by "Pulling itself up by the bootstraps." The boot process tests the hardware, sets system settings, then loads the operating system (MS–DOS). Booting is described on page 17.

Byte

A byte is the smallest usable piece of information; often used to represent a single character like "A" or "%." A byte contains 8-bits. Numbers aren't stored as characters but as binary values usually of from one- to eight-bytes. See page 12 for more information.

Command Line

The program path, name, and any arguments typed at the MS–DOS command prompt.

Command Tail

The "tail-end" of the command line. Everything following the command name.

Crash

Interaction of questionable logic (in program code or otherwise), misdirected attention, and acts of nature, which occurs just before making back-up copies of important data. See page 18 for some solutions.

CPU (Central Processing Unit)

This is microprocessor, the “brain” of the computer. The CPU reads instructions stored in RAM and ROM in machine code (yet another language, this one is the lowest-level — although you might be on the look-out for P–Code compilers, which are even lower-level) and creates or modifies data in RAM. The IBM–PC family uses Intel 80x86, including 8088, 80286, 80386, and so forth. The HP 95 uses the NEC (Nippon Electric Company) V20 CPU, which is compatible with the 8088, with enhancements and greater performance.

DOS (Disk Operating System)

The disk operating system controls access to disk drives and files as well as high-level access to the keyboard, display, and other input/output devices. The disk operating system runs application programs, which in turn call the operating system for disk access.

Filespec

Short for *File Specification,* filespec refers to any valid file name. Program documentation and DOS books often use this term when any file name would be acceptable, such as “COPY filespec A:” to represent any file to be copied to the A: drive.

Hexadecimal

The numbering system used to represent groups of four binary digits. This is base-16, represented by digits 0-9, followed by the letters A-F. Hexadecimal is often abbreviated as Hex. Binary and hexadecimal number are described on page 12.

Keyword

All MS–DOS commands, program names and other symbolic names.

Kludge

A slightly embarrassing way to work around limitations and bugs. In the computer programming business, nothing seems to fit very well, so we patch and adapt, and hope nobody looks too closely. There is a dreadful kludge demonstrated on page 76, where we found a way to work around a DOS limitation in batch file string comparisons.

[2006 update: The word is spelled *kluge.* There is no ‘d’ in it.]

MDA (Monochrome Display Adapter)

The 80 character by 25 line, text-only, monochrome display used by the original IBM–PC, and emulated by the HP 95. Compatibility of the HP 95 is discussed on page 22.

Mnemonic

A word trick used to make a complex or forgettable phrase easier to remember. Mnemonic, pronounced nee-mon-ick, words are coined from abbreviations, contractions or acronyms. For instance, the term RISC (Reduced Instruction Set Computer) is descriptive, but it also denotes apprehensions felt by the original planners.

MS–DOS (Microsoft Disk Operating System)

See also DOS. Microsoft’s MS–DOS operating system is the most popular piece of software ever written. Over sixty million computers run MS–DOS. Page 40 introduces MS–DOS. The MS–DOS community is larger than France.

OFFSET

Intel-compatible CPUs represent addresses as combinations of segment and offset. The beginning of each segment is at offset zero, so the offset portion of the address is the number of bytes from the beginning of the segment. See also Segment, and page 15.

Parser

Computer languages have words, grammar and rules — just like spoken languages. Parsers are computer programs that interpret commands written in these languages and translate them to commands the computer can understand. The MS–DOS command parser is COMMAND.COM; it interprets text we enter and translates it into commands DOS' language. Parsers are sometimes known as Lexical analyzers.

RAM (Random Access Memory)

This memory, the bits and bytes of main computer storage, can be read and changed. As long as the batteries are up to the job, the contents of RAM are maintained after you turn off the computer — on deskbound computers, RAM contents are lost whenever you turn off the computer; this is known as "volatile" memory. The HP 95 contains 512K bytes of RAM, which is partitioned between main memory (for programs), and the C: RAM disk.

RAM Disk

A RAM disk emulates a floppy- or hard-disk drive. From the point of view of applications, RAM disks look like very fast disk drives. The memory is battery-backed and data is maintained intact as long as the battery is healthy.

ROM (Read Only Memory)

ROM memory is permanently pre-programmed containing programs and data that can be read, but nor reprogrammed. On the HP 95, the BIOS, operating system, all of the built-in applications, and part of the C: disk drive are on ROM chips, and cannot be accidentally (or even intentionally) altered or erased.

Segment

In order to increase the number of bytes addressable by Intel processors beyond 64K bytes, the designers added segments and offsets within that segment. Each segment of memory begins on a 16-byte boundary, and can be as large as 64K. Segments can overlap, but only on those 16-byte boundaries. Addressing memory is explained on page 15.

Symbol

Computerdom is no different than the real world: there are symbolic ways to express almost everything. Money and cars are expressions of wealth; family and peace are symbols of happiness. The problem is the symbols we choose, compounded by the limited number of ways computers have to express things.

System Manager

The System Manager is a system of built-in utilities and protocols for writing add-in HP 95 programs as well as controlling the built-in applications. The System Manager was designed and written by Lotus Development Corp., the creators of the 1–2–3 spreadsheet. The System Manager is described on page 34.

Reference

The tables in this reference section describe the hardware, simplify creating key-assignments, summarize program instructions, and add a new look at an ASCII table, customized for the HP 95.

APNAME.LST Key Codes

These keystroke combinations are useful for assigning programs to keys in the C:_DAT\APNAME.LST file. **SHIFT**- and **CTRL**-modified regular keys are already taken, so this table lists just **ALT**- modified key combinations.

Key	ALT
Q	1000
W	1100
E	1200
R	1300
T	1400
Y	1500
U	1600
I	1700
O	1800
P	1900

Key	ALT
A	1E00
S	1F00
D	2000
F	2100
G	2200
H	2300
J	2400
K	2500
L	2600

Key	ALT
Z	2C00
X	2D00
C	2E00
V	2F00
B	3000
N	3100
M	3200
,	3300
.	3400

It's just the nature of things: not all keycodes are available, and some that are, already perform useful tasks. For instance, you wouldn't want to assign programs to cursor keys, or to **ALT**-combinations of most other keys. Here are some preferred keycodes, useful for assigning programs:

Key	SHIFT	CTRL	ALT	SHIFT+CTRL
FILER	·	AE00	AB00	AA00
COMM	·	AC00	B200	·
APPT	·	B600	B300	·
PHONE	·	BA00	B700	·
MEMO	·	BE00	BB00	·
123	·	C200	BF00	·
+-*/	·	C600	C300	·
(	·	·	8000	·
)	·	·	8100	·
Enter	·	·	1C00	·
Tab	0F00	9400	A500	·
@	·	0300	7900	·
MENU	C900	CA00	CB00	·
*	·	9600	3700	·
/ (slash)	·	·	·	·
- (minus)	·	8E00	·	·
+ (plus)	·	9000	·	·
= (equals)	·	·	8300	·

ASCII / Hex Codes

(This page intentionally left almost blank. This table was always pasted-in, instead of part of the book file, and the original was lost several years ago. The character set used by our HP 95 is called Codepage 850, which is slightly different from the more-common Codepage 437. The big list from this page eventually evolved into ASCIIcat, which includes all sorts of useful tables, and it is always available free of charge. Stop by the home page or search the Internet for your updated copy of ASCIIcat.)

HP 95 Memory Map

Memory	Addresses	Size
ROM BIOS	F000:0000 - F000:FFFF	64K
Bank switchable	E000:0000 - E000:FFFF	16K x 4 Data
Bank switchable	D000:0000 - D000:FFFF	64K Code
Bank switchable	C000:0000 - C000:FFFF	64K Code
-	B000:1000 - B000:FFFF	60K
Video buffer	B000:0000 - B000:0FFF	4K
Operating system	A000:0000 - A000:FFFF	64K
RAM disk C:	?	Varies
Transient program area	?	Varies
$SYSMGR	?	76K
DOS data	?	4K
DOS kernel	0000:2000 - ????:????	10K
BIOS data	0000:0400 - 0000:1FFF	7K
Interrupt vector table	0000:0000 - 0000:03FF	1K

When you exit to COMMAND.COM from the System Manager, approximately 1.6-3.2K of additional memory will be required for the transient portion of COMMAND.COM. When you boot the computer from DOS, the memory required by $SYSMGR will be available as transient program area, increasing available program memory by approximately 76K.

Debug Commands

Code	Purpose	Syntax
A	Assembly language mode.	A [memory address]
D	Display (dump) memory.	D [mem_start][mem_end]
E	Enter data in memory.	E address [data]
H	Hexadecimal Addition.	H n1 n2
L	Load file or sectors	(see text)
N	Name file.	N file_name
Q	Quit.	Q
S	Search for text.	S [mem_start][mem_end] “text”
U	Un-assemble.	L [mem_start][mem_end]
W	Write data to file.	W

See also  Page 88 explains how to use DEBUG.EXE.

MS–DOS Commands

This table lists just about all of the options available on the MS–DOS command line. Optional parameters are shown in square brackets. Here are the syntax rules followed in the table

c: Drive letter.
filespec Any file name.
[] Optional parameter.
| Choose one of two options.

Name	Purpose
$SYSMGR	Loads the System Manager. Works only if you boot from plain DOS. See page 69.
BREAK [0N | OFF]	Enables or disables break checking. See page 67.
CD *[dir_name]*	Change to a different directory. If used without an argument, displays the current directory name.
CHDIR *[dir_name]*	Synonym for CD.
CHKDSK [c:]*[filespec]* [/F] [/V]	Verify integrity of the disk. /F can also repair disks. /V shows files. See page 86.
CLS	Clears the screen, sends cursor to upper left corner.
COMMAND [/C] *[program]*	Load a new copy of the command processor. See page 69.
COPY *source destination* [/A | /B] [/V]	Copies one or several files to a new location. Can use wild cards for source if the destination is a path name. /A means ASCII, observes the end of file marker (character 26), while /B does a binary copy. /V verifies that the copy is accurate.
CTTY COM1 | CON	Change the keyboard and display to an external device.
DATE *[mm-dd-yy]*	Display or set the date. Use TIME to change the time.
DEL *filespec*	Delete one or several files.
DIR *[filespec]* [/W][/P]	Display a directory listing. /W for wide, /P to pause. See page 42.
DISPCTL [+C|-C] [+K|-K]	±C cursor tracking. ±K **ALT**+cursor movement.

`ECHO [ ON | OFF ]`	Turns command echoing on or off, displays current status. See page 71.
`ECHO text`	Display a text message. See page 71.
`ERASE`	Synonym for DEL.
`EXIT`	Exit the command processor and return to Filer.
`FOR %%a IN (set) DO cmd`	Repeat a command for each item in *(set)*. Most often used in batch files.
`FORMAT A: [/E] [/P] [/Q] [/R:ss] [/V]`	Format a RAM card. /Erase all sectors, /Pseudo-floppy, more space, not PCMICA compatible. /Quiet suppress prompts. /R: number of sectors for root directory file information (16 files per sector); default is 4. An internal command on the HP 95.
`GOTO label`	Used in batch files to transfer control to a label. Labels begin with a colon, as in ":LABEL". See page 74.
`IF text == text cmd`	Conditional test used in batch programs. See page 75.
`MD [C:]dir_name`	Create a new directory.
`MKDIR`	Synonym for MD.
`PASSWORD [/A][/M][/D]`	Create or change the startup password. /A=autolock, /M= manual lock, /D = disable password protection.
`PATH [path_name]`	Displays or changes the PATH environment variable. See page 64.
`PAUSE [>NUL]`	Suspends a batch file until the user presses a key. See page 78.
`PROMPT [prompt_text]`	Changes the command prompt. See page 65.
`RD dir_name`	Removes a directory. There must be no files in the directory.
`REM comment`	Add comments to batch files. Can also begin comments with a colon and a space, which runs faster and is easy to read. See page 78.
`REN old_name new_name`	Rename a file or group of files.
`RENAME old_name new_name`	Synonym for REN.
`RMDIR`	Synonym for RD.
`SERCTL /O | /W | /I`	Turns off the serial port (/O), changes to wire (/W) or infrared port (/I).
`SET`	Display all environment variables. See page 63.
`SET var=text`	Allocate or remove an environment variable. See page 63.
`SHIFT`	Shift position of replaceable parameters left for batch files. See page 72.
`TIME [hh:mm:ss]`	Display or set the time. See also DATE.
`TYPE`	Display the contents of a text file.
`VER`	Display the version of the MS–DOS operating system. On the HP 95 this is "MS–DOS Version 3.22."
`VERIFY [ON | OFF]`	Display, enable or disable verify. When enabled, MS–DOS verifies that files are written to disk correctly. The default is off. On slows the computer slightly, and adds little security.
`VOL [c:]`	Display the volume label. The HP 95 can only create labels when formatting disks, it does not include LABEL.COM.

PROMPT Symbols

This table lists the symbols available for creating DOS command prompts. Here is the syntax for the PROMPT command:

```
PROMPT $P$G
```

Symbol	Meaning
$$	The dollar sign.
$_	This symbol adds a carriage return and linefeed. This allows your path to extend over two or more lines.
$B	The piping character (|).
$D	Today's date, in the format "Sat 09-04-1991" or the format set with COUNTRY statement in CONFIG.SYS.
$E	The escape character, ASCII code 27.
$G	Greater than (>) symbol.
$H	Backspace. This is a destructive backspace, it erases the character under the cursor then moves the cursor to the left one position.
$L	Less than (<) symbol.
$N	The current drive letter (C:). This is much less useful than the $P command, which includes not just the drive letter, but also the complete path.
$P	The drive letter and path, in the format "C:_DAT"
$Q	Equals sign (=).
$T	This symbol inserts the current time in the format 22:33:44.55, complete with hundredths of a second.
$V	Operating system version. With the HP 95, this symbol displays the string "MS–DOS Version 3 22"

File Name Extensions

While MS–DOS doesn't restrict how we may name files, several file name extension conventions are commonly observed. This table lists file name extensions often found on MS–DOS computers, and especially the HP 95.

$$$	Temporary file.
ABK	HP 95 Appointment book file.
AI	Adobe Illustrator.
ARC	Archive. Compressed file containing packed information.
ASM	Assembly language source code.
BAK	Back-up copy of a file. The previous version of a file. May be erased.
BAS	BASIC programming language source code.
BAT	Batch file. Contains DOS commands.
BI	Include file for BASIC language.
C	C programming language source code.
CHK	File recovered by CHKDSK.
CNF	Configuration. Lotus uses 123.CNF.
COM	Executable program file. Exact image of program loaded in memory.
CTF	Code translation file. Used by HP 95 Comm program to translate characters.
CPP	C++ programming language source code.
DAT	Data.
DCF	HP 95 Comm program settings.
DEF	Linker definition.
DOC	Documentation. May contain formatting codes.
DRW	Drawing file for Micrografx.
ENV	Environment. Like CNF.
EQN	HP 95 Calc Solver file.
EPS	Encapsulated PostScript.
EXE	DOS Executable program file. Relocatable. The disk file is not an exact image of what the program will require in memory.
EXM	HP 95 System Manager Compliant program.
FCF	HP 95 Filer configuration file.
FOR	FORTRAN programming language source file.
H	C programming language include file.
HLP	Help. Several are in the C:_SYS\ directory.
HP	HP Graphics Language (plotter).
ICO	Icon (Microsoft Windows).
IMA	Zenographics image.
INC	Assembly language include file.
INI	Initialization.
INL	Inline code (C++). Like H, but can include code.
LCF	HP 95 Comm logon script.
LIB	Compiled programming code library.
LNK	Linker response file.
LST	Listing file created by a compiler. Also APNAME.LST for assigning HP 95 System Manager keys.
MAC	Assembly macros (Borland).
MAK	Make. Programming utility script.
MAP	Reference map. Created by programming linkers.
OBJ	Compiled program code intermediate file (before linking).
OVL	Program overlay module.
PAS	Pascal language source code.
PBK	HP 95 Phone Book.
PCF	Printer configuration.
PCX	Graphics file. Paintbrush format.
PIC	Lotus 1–2–3 graph file.
QLB	Quick Library for Microsoft QuickBasic.
SCR	Script. Used with DEBUG.EXE.
SYS	System file. Device drivers and utilities. Also CONFIG.SYS.
TIF	Tag Image File Format (graphics).
TMP	Temporary file. Used like $$$.
TEX	Text (ASCII).
TXT	Text (ASCII) file. Often created by HP 95 Memo program.
WKQ	Borland Quatro (spreadsheet)
WKS	Lotus 1–2–3 Release 1, 1A
WK1	Lotus 1–2–3 Release 2.0, 2.01, 2.2, 2.3.
WK3	Lotus 1–2–3 Release 3, 3.1, 1–2–3 for Windows 1.0.
WMF	Windows Meta File. Combined picture file for MS Windows.
WRK	Lotus Symphony Release 1, 1.01
WR1	Lotus Symphony Release 1.1, 1.2, 2.0.
XIP	HP 95 execute-in-place program.
XLM	Microsoft Excel Macro.
XLS	Microsoft Excel spreadsheet.
ZIP	Archive. Similar to ARC.