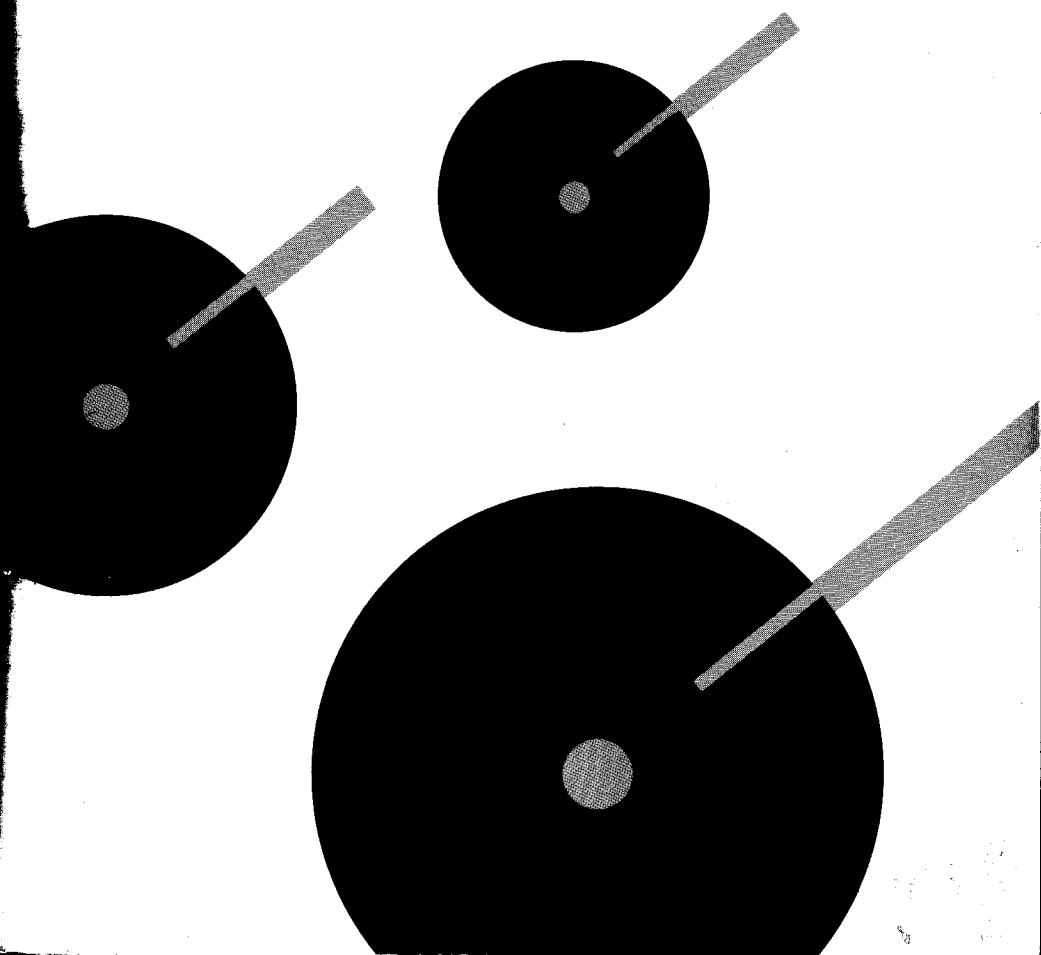


Paul Mace
SOFTWARE

*h*TEST • *h*FORMAT

Advanced Hard Disk Diagnostics designed by
Kolod Research

Delivers what your hard disk manufacturer promised ...
Control, Reliability and Speed



Emergency Help Index

If your computer announces: See this page:

"Abort, Retry, Ignore?"	4-12
"Bad or missing command interpreter"	2-16
"Error loading operating system"	2-16
"File allocation table bad"	2-19
"General failure reading drive c:"	2-19
"Invalid drive specification"	2-4
"Invalid partition table"	2-4
"Missing operating system"	2-16
"Non-system disk or disk error"	2-16
"Sector not found error reading drive c:"	4-12
"17XX" errors (during power-on testing)	2-2

**For free technical support (weekdays only)
call Paul Mace Software at (503) 488-0224
from 8:00 to Noon and 1:00 to 4:00 Pacific Time.**

HTEST/HFORMAT

Version 2.0

**Copyright © 1984-1988 Kolod Research, Inc.
All Rights Reserved**

**Manual © Paul Mace Software, Inc.
Revised June 1988**

Kolod Research, Incorporated, makes no representations or warranties with respect to the contents of this manual or the associated programs, and specifically disclaims any implied warranty of merchantability or fitness for any particular purpose. Furthermore, Kolod Research, Incorporated, reserves the right to revise this manual and its associated programs, and to make any changes from time to time in their content, without obligation to notify any person of such revisions or changes.

No part of this manual may be reproduced, stored in any retrieval system, or transmitted in any form or by any means, whether electronic, mechanical, photocopy, recording, or otherwise, without the prior written consent of Kolod Research, Incorporated.

The following are trademarks owned by the listed companies: "HTEST/HFORMAT," "DFT," "XFDISK," Kolod Research, Incorporated. "Mace Utilities," "Mace Vaccine," Paul Mace Software, Incorporated. "IBM," "Personal Computer DOS," "PS/2," International Business Machines Corporation. "AT&T," American Telephone and Telegraph, Incorporated. "MS-DOS," Microsoft Corporation. "Speedstor," Storage Dimensions. "Vfeature," Golden Bow Systems. "Disk Manager," Ontrack Computer Systems. "Bernoulli Box," Iomega Corporation. If any other trademark has been used without acknowledgment, we apologize and request that we be notified in order that we may correct our mistake.

Table of Contents

Emergency HELP Index	Inside Front Cover
Preface	v
Changes from version 1.5	vii
Supported Hardware List	viii
Friendly Advice	ix
 Chapter I. Hard Disk Maintenance Guide	 1-1
Format a Hard Disk	1-2
Test a Hard Disk	1-8
Measure Disk Performance	1-13
Change Disk Interleave	1-16
Create and Save a Partition Table	1-22
 Chapter II. Hard Disk Recovery Procedures	 2-1
Power-On Testing Errors	2-2
Invalid Drive or Partition Table	2-4
Operating System Startup Failure	2-16
Bad Directory or FAT	2-19
Programs Crashing Under DOS 3.3	2-21
 Chapter III. Program Reference	 3-1
CRC	3-2
GETSEC AND PUTSEC	3-4
HFORMAT	3-8
HOPTIMUM	3-16
HPERF	3-23
HTEST	3-27
RELOCATE	3-33
33TRAP	3-35
XFDISK	3-37
 Chapter IV. An Introduction to Hard Disks	 4-1
Physical Description	4-1
Formatting and Partitioning	4-5
Hard Disk Problems	4-9

Table of Contents, continued

Appendices

PC's and Primary Storage	A-1
PC's and Secondary Storage	B-1
Disk Encoding Schemes	C-1
Disk(ette) Primer	D-1
DOS Disk Space	E-1
Interleaving and Performance	F-1
Disk Error Codes	G-1

Preface

HTEST/HFORMAT is a unique set of machine-level utilities for fixed disks and hard cards and their controllers. The programs allow you to operate on three levels of involvement:

- normal preparation and testing of hard disks
- recovery from hard disk catastrophes
- sensitive analysis/optimization for intensive applications

This manual is organized along these three levels. Chapter one, Hard Disk Maintenance, guides you through the steps necessary for installing and checking new hard disks and optimizing their performance. Chapter two, Recovery Procedures, explains step-by-step the applications of HTEST/HFORMAT in curing major disk disasters. Chapter three, Program Reference, details the options and operations of each of the programs in alphabetic order. Each of these chapters contains an introduction which is required reading, as is the Friendly Advice that follows this preface.

Chapter four, Introduction to Hard Disks, presents a straightforward explanation of the physical and logical structures of hard disks. If you're new to hard disks, I recommend that you read this chapter first. Appendix G explains the error codes that HTEST and HFORMAT report when testing hard disks.

The HTEST/HFORMAT set is not a panacea; it is a tool. And like any tool, it will reward you for the time you invest in learning how to use it. Its publication is part of a growing awareness that the standard pieces no longer fit together as snugly as they once did, and that with the right tools you can work on your problems more effectively than anyone else.

This page is blank.

Changes from version 1.5

This release of HTEST/HFORMAT version 2.0 includes the following major changes:

- HPERF, a hard disk performance analyzer, is new.
- XFDISK, which replaces DOS Fdisk, is also new.
- HFORMAT can now be run non-destructively, with the /NV switch, although no sane person would do this without a backup.
- HOPTIMUM is also non-destructive now, although again it is insane to use these programs without a backup.

Please note that other minor changes and additions to the version 1.5 programs are not noted here. Previous users of HTEST/HFORMAT should check Chapter 3 (program reference) to familiarize themselves with the differences.

Supported Hardware List

The HTEST/HFORMAT programs work on IBM and IBM-compatible PC, XT, and AT computers with standard MFM or RLL controllers and disks of up to 16 heads and 1024 cylinders. They deal with controllers and with hard disks as physical units, not as logical DOS drives.

The HTEST/HFORMAT programs have been tested and shown to work correctly with the following controllers:

- Adaptec
- Data Technology Corp. (DTC)
- Compaq ESDI
- IBM PS/2 ESDI¹
- IBM PC, AT, XT
- Scientific Micro Systems (OMTI)
- Sysgen (removable cartridge controller)
- Western Digital Corp. (WDC)
- Xebec

The programs will not work the following controllers:

- SCSI²
- Iomega Corp. (Bernoulli Box)
- Tallgrass

¹ HFORMAT and HOPTIMUM will not work on PS/2 ESDI controllers.

² HTEST and HFORMAT may work on SCSI controllers that contain ROM BIOS extension firmware.

Friendly Advice

1. **WARNING:** disk files are potentially ephemeral and your best insurance is a current and safe backup of your work. The HTEST/HFORMAT programs go to great lengths to protect you from accidentally overwriting your files, but no program can guard against the unexpected. Back up early and back up often.

2. Use the CRC program to check your HTEST/HFORMAT diskette when you receive it. Place the diskette in drive A: and enter

```
A>CRC *.*
```

and then

```
A>TYPE CRC.LST
```

and check that the listings from CRC match this file's listings. Any files that do not match have been altered and should not be used. See page 3-1 for more information on CRC.

3. I recommend that you make a copy of the HTEST/HFORMAT diskette for everyday use. Use DOS's DISKCOPY or COPY *.* command and run CRC *.* again against the new diskette to check the files, then store the original diskette in a safe place.

4. When a new drive is installed it should be left OFF for 24 hours to adjust to the temperature in its new environment. Any computer or disk subjected to repeated heating and cooling will develop problems. Also insure that the disk will not be bumped or shaken while in operation, and protect it from dust, smoke, vibration, power surges, and strong magnetic fields such as vacuum motors.

This page is blank.

Chapter 1: Hard Disk Maintenance Guide

This chapter shows how to use the HTEST/HFORMAT programs in the preparation and maintenance of a hard disk. It is designed to be easy to follow, and generally uses the programs' default values. If you encounter unfamiliar concepts or terms, read the introduction to hard disks in Chapter four; for detailed information on each program and its options, consult the program references in Chapter three.

The examples in this chapter assume that the HTEST/HFORMAT diskette (or a copy of it) is in drive A:, and that drive A: is the current drive. The command line examples show the DOS prompt (which can vary from the one shown) as "A>" and the command as you would type it immediately following. For example,

A>HTEST

directs you to type "HTEST" at the DOS prompt and then press ENTER. You may enter commands in upper or lower case.

To make certain that the programs don't trip over anything in your CONFIG.SYS file, and that there are no "terminate and stay resident" programs (TSR's) in memory, rename CONFIG.SYS and AUTOEXEC.BAT:

A>RENAME C:\CONFIG.SYS CONFIG.TMP

A>RENAME C:\AUTOEXEC.BAT AUTOEXEC.TMP

and then reboot. Restore the filenames and reboot after using HTEST/HFORMAT:

A>RENAME C:\CONFIG.TMP CONFIG.SYS

A>RENAME C:\AUTOEXEC.TMP AUTOEXEC.BAT

Format a Hard Disk

HFORMAT performs the physical formatting which is necessary to prepare a new hard disk for DOS's partitioning and logical formatting. In addition, HFORMAT tests the newly formatted disk by repeatedly writing and reading varying bit patterns. You can limit the area formatted and specify the number of times to test the surface; you can also specify the sector interleave to use when formatting.

HFORMAT displays all formatting and testing information on the screen. To print a copy of the format parameters, start printer echoing with the Ctrl-PrtSc key combination at the DOS prompt before starting the program.

Also before starting, you must remove any BUFFERS, FILES, or FCBS commands from your CONFIG.SYS file (or just rename the file temporarily) and reboot, and make sure that no memory-resident programs (including disk-caching programs) are running.

Important: If you intend to use HFORMAT on a disk that is in use you must first back up all your files. Even though the program can be run non-destructively, normal formatting will destroy all disk information. You may also want to use HOPTIMUM first to determine the best interleave for your system.

Important: If you have an auto-configure controller (which includes most RLL controllers) you must not format cylinder zero of head zero. These controllers rely on special drive information written between the sectors by the controller's own formatting program. HFORMAT starts by default at cylinder one: use extreme caution when changing the starting cylinder.

Starting HFORMAT

If you have only one hard disk, enter HFORMAT at the DOS prompt (include drive and path names if HFORMAT.COM is not in the current directory):

A>HFORMAT

If you have two (physical) hard disk drives, add the DOS name of the drive to use. To format the second drive, for example, enter:

A>HFORMAT D:

HFORMAT begins by displaying a version and copyright message, and pauses for a few seconds while it locates the drive controller. It displays this information as "Signature value x,y" where "x,y" is two hexadecimal numbers which indicate the beginning of the controller's program, or BIOS. If nothing happens at this point the program has locked up while trying to get around DOS to work directly with the controller; you will need to reboot the computer. Check that no resident programs start running during the reboot. Restart HFORMAT and include the "No Big Deal" option on the command line:

A>HFORMAT /NBD

If you have two (physical) hard disk drives, add the NBD option after the drive's DOS name:

A>HFORMAT D: /NBD

This directs the program to ignore minor (and correctible) errors, which slightly lessens HFORMAT's error-spotting abilities but allows the program to work with otherwise recalcitrant controllers.

Defining the Options

HFORMAT now displays two lists, a list describing the number of heads, cylinders, and sectors per track on this drive, and a list of format parameters. These options determine the area to be formatted; in this case the program defaults will direct HFORMAT to

- format the entire drive, starting at cylinder one
- format with an interleave of two (AT's) or six (XT's)
- test the formatted area three times
- lock out any tracks that return read/write errors

Now HFORMAT asks you if you wish to change any test parameters. If you elect to change the test, the program displays each option and asks for a new value. You can enter a different value, or you can preserve the current value by just pressing ENTER. For information on all these test options consult the program reference for HFORMAT on page 3-8.

After you have edited all the options HFORMAT again displays the disk drive description and the format parameters, and asks again whether you want to change them. You can change these options as many times as you like before proceeding.

Now HFORMAT will ask you if you want to begin formatting. If you respond by pressing "Y" it will ask a second time, just to make sure. **Important:** stop HFORMAT by pressing "N" at either prompt unless you are certain that you want to format the disk and destroy all information on it.

Important: If you have an auto-configure controller (which includes most RLL controllers) you must not format cylinder zero of head zero. These controllers rely on special drive information written between the sectors by the controller's own formatting routine.

Testing the Controller

Before HFORMAT begins working it performs three quick tests of the drive controller, displaying the following messages if the controller responds normally:

- Performing drive recalibrate...
- Drive recalibrate performed successfully
- Performing internal drive controller test...
- Internal controller test performed successfully
- Performing internal drive reset test...
- Internal drive reset performed successfully

If HFORMAT encounters any problems with the controller it will ask you whether to proceed with the test. You should cancel the test, as these errors indicate a need for expert service on the controller or the drive (or both).

Running HFORMAT

As HFORMAT works it displays a heading which shows which cylinder and surface are being formatted. You can suspend the program by pressing the space bar (HFORMAT may take a moment to respond); you can then continue formatting by pressing C, or abort HFORMAT and return to DOS by pressing A. When running HFORMAT over large areas of the disk you should turn down the display brightness to guard against screen burnout.

After formatting, HFORMAT begins testing the disk. It will list errors as they occur, noting the cylinder and head and a hexadecimal error code returned by the controller (Return Code, abbreviated R/C). If you press the SPACE bar, HFORMAT will suspend test-

ing (it may take a moment to respond) and wait for a response from you. Press:

- A to Abort (finish) testing and return to DOS
- C to Continue testing
- N or P to view the Next or Previous screenful of errors
- H (Help) to view a list of error (R/C) codes

HFORMAT will also display these choices when it finishes testing, adding the option of pressing

- B to begin the Add Bad Tracks routine

You can print all the errors by paging through them and using the Shift-PrtSc key combination. Print the Help listing as well.

If you are formatting a new disk, look for a list of bad tracks on the drive case and use HFORMAT's "add bad tracks" routine to lock out any of them that were missed during testing. HFORMAT will add these tracks to its list of errors, using the return code "00h." Although this is not an error code, HFORMAT has in fact marked these tracks unusable. If you enter a track that has already been locked out HFORMAT ignores it and asks for another.

If HFORMAT reports errors in the first track (cylinder zero, head zero) then your disk needs service or replacement. You will not be able to create a partition table, and the disk is useless.

What Next?

The next step in preparing a new hard disk for use is partitioning. You can use XFDISK to create DOS and other partitions and save a copy of the partition table as insurance against future disk

problems. Following partitioning, you will need to use DOS's **FORMAT** or the Mace Utilities **FORMAT-H** to prepare the DOS partition for use. For a detailed explanation of the entire process read Chapter four.

Test a Hard Disk

HTEST checks the surface of a physically formatted hard disk that is in working order. You can direct it to test only the ability to read the disk, in which case it will not alter any of your files, or you can direct it to try writing information and then reading it, which will destroy information already on the disk. You can also specify the particular heads and cylinders to test, the number of times to repeat the test, and which of three head movement patterns to use.

HTEST displays all testing information on the screen. To print a copy of the information, start printer echoing with the Ctrl-PrtSc key combination at the DOS prompt before starting the program.

Also before starting HTEST you must remove any BUFFERS, FILES, or FCBS commands from your CONFIG.SYS file (or just rename the file temporarily) and reboot, and make sure that no memory-resident programs (including disk-caching programs) are running.

Starting HTEST

If you have only one hard disk, enter HTEST at the DOS prompt (include drive and path names if HTEST.COM is not in the current directory):

A>HTEST

If you have two (physical) hard disk drives, add the name of the drive to use. To test the second drive, for example, enter:

A>HTEST D:

HTEST begins by displaying a version and copyright message, and pauses for a few seconds while it locates the drive controller. It displays this information as "Signature value x,y" where "x,y" is two hexadecimal numbers which indicate the beginning of the controller's program, or BIOS. If nothing happens at this point the program has locked up while trying to get around DOS to work directly with the controller; you will need to reboot the computer. Check that no resident programs start running during the reboot. Restart HTEST and include the "No Big Deal" option on the command line:

```
A>HTEST /NBD
```

If you are testing one of two drives, add the NBD option after the drive name:

```
A>HTEST D: /NBD
```

This directs the program to ignore certain (correctible) errors, which slightly lessens HTEST's error-spotting abilities but allows the program to work with otherwise recalcitrant controllers.

Defining the test

HTEST now displays two lists, a list describing the number of heads, cylinders, and sectors per track on this drive, and a list of test parameters. These options determine the type of test that will be performed, and in this case they will be the program defaults:

- test the entire drive (all surfaces and all tracks)
- test without harming information on the disk (read only)
- test from the first to the last track in order (quick test)
- test the drive once only ("repeat loop counter" equals one)

HTEST asks you if you wish to change any of the test parameters. Check that the "read/write test type" option is "READ" before proceeding. If it is, press "N" to accept these defaults. If it is "WRITE" and you have ANY valuable information on the drive, press "Y" to change the test.

If you elect to change the test, HTEST will display each option and ask for a new value. You can enter a different value, or you can preserve the current value by just pressing ENTER. For information on all these test options consult the program reference for HTEST on page 3-27.

To change the test to READ, skip the first four options by pressing ENTER, and enter "R" at the "Enter new test type" prompt. Press ENTER again at the "loop counter" prompt to finish with these changes. HTEST again displays the disk drive description and the test parameters, and asks whether you want to change them. You can change these options as many times as you like before proceeding.

Now HTEST asks you if you want to begin the test. Press "Y" to begin (or press "N" and HTEST will end, returning you to DOS).

Before HTEST begins testing the disk surface, it performs three quick tests of the drive controller. If HTEST encounters any problems with the controller it will ask you whether to proceed with the test. These errors indicate a need for expert service on the controller or the drive (or both).

Testing the surface

When the disk surface test begins, HTEST clears the screen and displays a heading which, as the test progresses, will show which

cylinder and head are being tested. Beneath that are three columns in which HTEST will record errors as they occur, noting the cylinder and head and a hexadecimal error code returned by the controller (Return Code, abbreviated R/C). If you press the SPACE bar, HTEST will suspend the test and wait for a response from you. Press:

- A to Abort (finish) testing and return to DOS.
- C to Continue testing.
- N or P to view the Next or Previous screenful of errors.
- H (Help) to view a list of error (R/C) codes.

These choices will also be displayed after HTEST finishes testing the disk. You can then page through the errors and use Shift-PrtSc to print them: print the Help listing as well. When running extensive tests you should turn down the screen brightness to guard against screen burnout.

Using the results of HTEST

HTEST results can help in locating the source of hard disk problems. For example, errors concentrated in one area are often due to a head crash, while small scattered problems are more likely to reflect manufacturing defects. If periodic tests begin to find increasingly greater numbers of such errors you should have the drive serviced or replaced immediately; whatever is degrading the surface is not going to get better on its own.

If HTEST reports increasing errors toward either the first few or the last few cylinders this could indicate trouble with the armature positioning mechanism. If test results vary from run to run the problem may be with the controller or the drive; try testing the

same drive with a different controller, and vice versa, to isolate the problem.

You can use HFORMAT (with the BEGIN and END options) to lock out selected problem areas found by HTEST but missed when the disk was first formatted.

Measure Disk Performance

HPERF allows you to measure your hard disk's speed in two ways: seek time, which is the time required for the heads to move from one track to another, and transfer rate, which is the speed at which information can be moved from the disk to the computer's memory. Together these factors determine the maximum rate at which the system can transfer data from the disk to the controller and then to memory. Not included in these measurements are the factors of operating system access methods or caching software.

Before starting HPERF you must remove any BUFFERS, FILES, or FCBS commands from your CONFIG.SYS file (or just rename the file temporarily) and reboot, and make sure that no memory-resident programs (including disk-caching programs) are running.

Starting HPERF

If you have only one hard disk, enter HPERF at the DOS prompt (include drive and path names if HPERF.COM is not in the current directory):

```
A>HPERF
```

If you have more than one physical drive, add the number of the drive to use (0-9) and a colon. For example, to evaluate the second hard drive (drive number 1), enter:

```
A>HPERF 1:
```

When HPERF starts, it divides the screen into four boxes. In the lower right corner it lists its Testing Parameters, which define the

size and range of its tests. In the upper left corner it displays the characteristics of the drive to be tested (heads, cylinders, and sectors per track).

In the upper right corner HPERF lists its options (and their codes) and a prompt. If you enter a two-letter option code the program prompts you for a new value for that option. For information on all the options see the program reference for HPERF on page 3-23.

Beginning an HPERF Evaluation

To accept the default test, which is a quick test limited to the next to last cylinder, just enter "GO" to begin the test. HPERF then begins its measurements, and displays the results in the lower left corner of the screen. After finishing the tests, the program waits for further instructions. You can use the Shift-PrtSc key combination to print a copy of the results, and then enter "EX" to quit the program and return to DOS.

Interpreting the Results

HPERF measures four aspects of hard disk performance:

- track to track seek time
- average seek time
- test track transfer rate
- random transfer rate

In addition, HPERF measures and reports the disk's rotational speed in revolutions per minute.

Track to track seek time is the time (in milliseconds) required for the disk to move its heads from one cylinder to the next (from cylinder zero to cylinder one in this case), and this limits the speed with which the drive can read (or write) consecutive cylinders, as it must when loading large files.

The average seek time is computed from random cylinder to cylinder movements within the test range, and demonstrates the drive's performance when reading many small files or reading a file that is badly fragmented (stored on widely scattered tracks). Because of the larger head movements needed, the average seek time will be greater than the track to track seek time.

The test track transfer rate is the speed (in Kilobytes, or thousands of characters) at which the system can transfer a disk file into the computer's memory, assuming that the information is stored in sequential sectors (or tracks) beginning at the test track.

The random transfer rate measures the same movement from disk to memory but selects random groups of sectors within the test area. Again, because of the additional head movements, this rate will be slower than the test track transfer rate.

Change Disk Interleave

HOPTIMUM evaluates the performance of your hard disk under different interleave schemes to determine which interleave would yield the fastest disk performance. In addition, it can reformat the entire disk to the best interleave, lifting your files off the disk and replacing them as it works.

HOPTIMUM measures the time needed to read an area of the hard disk at each interleave, to arrive at comparative measures of efficiency. You can define the area of the disk and the range of interleaves to use, as well as the number of times to read the area in evaluating each interleave.

If your hard disk is not yet physically formatted, you can use HOPTIMUM to find the best interleave and then run HFORMAT to format the disk at this interleave. If you make changes to your system (an accelerator board, for example) that affect processor or disk speeds, you can use HOPTIMUM to check whether the current interleave is still best, and reformat the disk to a new interleave if necessary.

Important: Do not use HOPTIMUM without a current and correct backup of all important hard disk files. HOPTIMUM's careful approach to moving large numbers of sectors is no guarantee against mishaps such as voltage sags.

Before starting HOPTIMUM you must remove any BUFFERS, FILES, or FCBS commands from your CONFIG.SYS file (or just rename the file temporarily) and reboot, and make sure that no memory-resident programs (including disk-caching programs) are running.

On two-speed computers the efficiency of an interleave will depend on the processor speed. For accurate results, run HOPTIMUM with the computer set at the most often used speed.

Starting HOPTIMUM

If you have only one hard disk, enter HOPTIMUM at the DOS prompt (include drive and path names if HOPTIMUM.COM is not in the current directory):

A>HOPTIMUM

If you have more than one physical drive, add the number of the drive to use (0-9) and a colon. For example, to evaluate the second hard drive (drive number 1), enter

A>HOPTIMUM 1:

HOPTIMUM begins by displaying a version and copyright message, and asks whether any background (resident) programs are running. If you are not certain, enter "N" here and at the next prompt ("Do you wish to continue") and the program will end; then reboot and check that your AUTOEXEC.BAT file does not start any resident programs.

Checking the Drive Description

HOPTIMUM next displays a list describing the number of heads, cylinders, sectors per track, total bytes, and controller type for this drive, and asks whether the list is correct. Enter "Y" if it looks correct, or enter "N" to edit the list.

If you elect to change the list, HOPTIMUM displays each option and asks for a new value. You can enter a different value, or you can preserve the current value by just pressing ENTER. There should be no need to change these values unless you have an unusual disk or controller. In addition, HOPTIMUM asks you to identify a Write Pre-compensation Cylinder, which some types of drive use to adjust the write current strength, and a Reduced Write Cylinder, which serves a similar purpose. Press ENTER at these options to retain the default values of -1 (which means "ignore") unless your drive manual specifically mentions these items.

Defining the Test

Now HOPTIMUM displays a list of testing parameters, which describe the interleave testing it will perform. In this case the list will use the program defaults, which direct HOPTIMUM to:

- limit the test area to surface zero
- limit the test area to the next to last cylinder
- evaluate all possible interleaves
- conduct READ tests only
- use the TRACK timing assumptions
- take 250 samples at each interleave

The program asks whether you want to change any of these parameters: check that the "read/write test" option is "READ" before proceeding. If it is, enter "N" to accept these defaults. If it is "WRITE" and you have ANY valuable information on the drive, enter "Y" to change the list.

If you elect to change the test, HOPTIMUM displays each option and asks for a new value. You can enter a different value, or you

can preserve the current value by just pressing ENTER. For information on all these test options consult the program reference for HOPTIMUM on page 3-16.

To change the test to READ, skip the first six options by pressing ENTER, and enter "R" at the "Enter read or write test type" prompt. Press ENTER again at the next two prompts to accept the default values and finish with these changes. HOPTIMUM again displays the test parameters, and asks whether you want to change them. You can change these options as many times as you like (and print a copy of them with the Shift-PrtSc key combination) before proceeding.

Determining the Current Interleave

Now HOPTIMUM asks you if you want to proceed. Enter "Y" and it asks whether you want to determine the current interleave and transfer rate. Enter "Y" again; HOPTIMUM takes a minute to issue its read requests to the test track and display the results. The "transfer rate" is the speed at which the system can move information from the disk into memory, and is given in kilobytes (thousands of characters) per second.

HOPTIMUM computes an average transfer rate based on the sample read requests it made, and computes the probable interleave of the test track based on its timing information and knowledge of the disk. HOPTIMUM will later compare its testing results to the present setup (you can make a copy of this information with the Shift-PrtSc key combination).

Evaluating Interleaves

Before beginning its interleave tests, HOPTIMUM asks whether you want to continue. Enter "Y" to begin the testing (or enter "N" to quit and return to DOS). HOPTIMUM takes a minute or more to report its progress and its results between interleaves.

These delays increase rapidly as you increase the test area and the sample size; use a small run such as this default test first to develop a sense of the time the program requires, and then run it again with a test area of at least ten cylinders and all heads. This slightly larger test will give you a more accurate idea of the transfer rates your applications are likely to achieve.

HOPTIMUM lists the results of its sampling (and any errors it encounters) as it works. It reformats the test area (first moving all disk information into memory and then replacing it) to each new interleave value and then issues random read requests, averaging them to arrive at a transfer rate for that interleave.

You can halt HOPTIMUM by pressing the SPACE bar; it finishes testing the current interleave and then asks whether you want to abort (stop) the program. If you enter "Y" HOPTIMUM restores the test area to its original interleave and returns control to DOS. No files will be harmed during this bailout.

Important: Some errors can be expected, and HOPTIMUM will not reformat tracks that it cannot read correctly. But if the program encounters frequent errors you should stop it and run HTEST to determine the problem.

After finishing its interleave tests, HOPTIMUM summarizes the results in a table, listing each interleave's transfer rate and the time (in seconds) that it took to complete all its read requests. HOP-

TIMUM also marks the fastest interleave with an asterisk (you can make a copy of this table with the Shift-PrtSc key combination).

Changing the Interleave

Finally, HOPTIMUM reports the increase in disk speed that you could expect if you reformatted the disk to the best interleave, and asks whether to begin the reformat. If that best value is the current one, there is no advantage to proceeding. If you enter "Y" the program asks again, just to be sure.

Important: Check that you have a current and correct backup of all important hard disk files before proceeding. Enter "Y" one last time to begin the reformat, and HOPTIMUM will display the cylinder and head numbers as it works, from the last cylinder to cylinder one. The program will not reformat cylinder zero under any circumstances.

You can halt the reformat by pressing the SPACE bar; HOPTIMUM then asks whether you want to abort (stop) the program. If you enter "Y" HOPTIMUM quits and returns control to DOS, leaving part of the disk formatted under the new interleave. This result (a disk with areas that use different interleaves) will not cause any problems, as DOS and the controller always read or write to sectors by number and without concern for how they are laid on the tracks. No files will be harmed during this bailout.

Note that HOPTIMUM does not do any track testing and it will reformat every track that it can read. You should find and lock out any bad areas of the disk before running HOPTIMUM; see "Test a Hard Disk" on page 1-8 for information on HTEST, or use Mace Utilities REMEDY if you already have files on the disk.

Create and Save a Partition Table

Unlike other partitioning programs, XFDISK allows you to preserve a copy of the first and most important sector on your hard disk, the one that allows the computer to find the DOS partition (not to mention the files in that partition). And the recovery procedures in chapter two show how XFDISK can use that copy to resuscitate a disk. For that reason alone, you should complete the steps in this section.

This discussion assumes that you understand the concepts of disk sectors, partitions, and the master boot record. If not, please read Chapter four before running XFDISK.

Also before starting XFDISK you must remove any BUFFERS, FILES, or FCBS commands from your CONFIG.SYS file (or just rename the file temporarily) and reboot, and make sure that no memory-resident programs (including disk-caching programs) are running.

Starting XFDISK

These examples assume that you will save the master boot record in a diskette file called HD0BOOT.REC, preferably on the same diskette as the XFDISK program. This insures that you can get to the file to recover the use of the hard disk. You should also keep a backup copy of HD0BOOT.REC.

Start the program with the following command:

```
A>XFDISK
```


XFDISK begins by displaying its Main Menu. If you have more than one hard disk drive, XFDISK will allow you to choose which of them to use.

If This is a New Disk

The advantage of using XFDISK when setting up your partitions is that it saves a copy of the master boot record, which can be invaluable if you need to replace it. And unlike DOS's FDISK, it does not destroy existing FAT or directory information in the partitions.

First, create a bootable DOS partition with Main Menu choice 1. Choose "Create a DOS or a DOS Extended Partition" on the next menu, and choose the DOS version you're going to use from the menu after that. XFDISK will prompt you through the definition of the partition. On disks of 32 Megabytes or less, the booting partition can use the entire disk. On larger disks, DOS 3.3 allows you to divide the disk into a booting and an extended partition, which XFDISK will also define.

Set the partition's first cylinder to 1, rather than 0. Remember that the first tracks of a partition store the FAT and the root directory. Since DOS looks at the FAT every time it reads or writes to the disk, these tracks get a lot of use. If they share a cylinder with the MBR, the MBR bears the same risk of damage even though it's not being read. Considering the MBR's importance, losing ten or twenty kilobytes of storage is a small price to pay for the insurance of starting the partition on cylinder 1. And to leave a safe parking spot for the heads, don't include the last cylinder in any partition.

After defining the partition you must return to the main menu and use choice 2 to activate it (unless you're working with the second of

two drives). This partition will then be marked as the boot partition, so that the system loads DOS from it when the machine is turned on.

If This Disk is Already in Use

If your disk is already formatted, you can use main menu choice 4 to display the partition table and then print a copy of it using the Shift-PrtSc key combination. If you have no valuable information on the disk, or if you don't mind restoring it from a backup, you may want to use XFDISK to move your partition to cylinder one, as outlined in the previous paragraph.

Saving the Partition Table

The next step is to tell XFDISK to save the partition table. Select main menu choice 7, and choice 1 on the next menu. When the program presents its default filename, backspace over it and enter

A:HD0BOOT.REC

so it will be saved to diskette (you can use your working copy of HTEST/HFORMAT). XFDISK will let you know if it has any trouble writing the file.

Placing the Table Onto the Disk

If this is a new hard disk you will also need to direct XFDISK to write the master boot record, containing the partition table, to the disk's first sector (cylinder zero, track zero, sector one).

Warning: Do not take this step if you already have information stored on the disk.

Select main menu choice 7, and choice 2 on the next menu. When the program asks whether you want to proceed, enter "Y" and XFDISK will let you know if it has any trouble writing to the disk. Then quit the program and reboot the computer with a DOS system diskette in drive A:. DOS should now recognize drive C:, although there won't be any files in it yet.

The remaining step in getting your hard disk running is logical formatting. Use DOS FORMAT or Mace Utilities FORMAT-H, and include the /S (system) switch to create a bootable partition.

Disk Insurance

In addition to this backup copy of the MBR, you should be using RXBACK from the Mace Utilities to guarantee that you can recover your files after disk reformats or directory mangling. And for more information on the uses of HD0BOOT.REC, read the procedures for recovering from an "Invalid Drive or Partition Table" error, in Chapter two.

This page is blank.

Chapter 2: Hard Disk Recovery Procedures

This chapter is designed to help you recover from major disk problems. The procedures use the full power of the HTEST/HFORMAT programs and some of them are complex, but they are presented in careful detail. If you are unfamiliar with the terms used in these remedies you should first read Chapter four.

The procedures generally assume that drive A: is the current drive. Command line examples show the DOS prompt as "A>" and the command as you would type it immediately following. For example:

A>HTEST

directs you to type "HTEST" and then press ENTER. Note that the GETSEC and PUTSEC command examples should be typed on one line, not on two as printed in this manual. In addition to the HTEST/HFORMAT diskette (or a working copy) you will need the main DOS program diskette (use a copy if you have one, instead of the original).

Important: the programs can't abide buffers, caches, and resident programs. The simplest way to avoid difficulties is to reboot the computer with the DOS program diskette in drive A:.

Keep in mind that when you need to restore files, you should use the original copies of commercial programs. And after recovering from catastrophic damage to your disk, look for new hardware or software that may have caused the problem.

Finally, remember that most problems have solutions, although the time and equipment costs may begin to rise dramatically. Call us for more advice if you don't succeed with these procedures.

Power-On Testing Errors

If you turn on your computer and, during its power-on self testing (POST) routines, you see messages like these (where "XX" is two numerals):

- Error 17XX
- 17XX

then the system has found a problem with the hard disk or controller. For PC and XT machines, any such problem will result in error 1701. AT's can return nineteen different errors, and none of them is good news.

If you recently installed the disk or added any other hardware, turn off the power, remove the cover, and check that the controller is seated in its expansion slot and that all cables are securely (and correctly) connected. See your computer or disk manual for cabling instructions, and make sure that no connectors have been reversed or flipped.

If you have an AT machine, run its Setup program to check that the disk type is correct. Start the computer with the DOS diskette containing SETUP.EXE and run the program by entering

A> SETUP

Check your drive type against the one shown by the program (your computer or disk manual, or your dealer, will have the correct value). After you've corrected (or accepted) the setup and the program ends, restart the computer and check for POST errors in the 1700's. You should also check your machine's battery.

Also note that if the drive hasn't been low-level formatted, PC or XT machines may display error 1701, while AT machines may display error 1780 (for the first fixed disk) or 1781 (for the second). This error should disappear after you use HFORMAT.

If you have a formatted disk and you still get POST errors, put the HTEST/HFORMAT diskette in drive A:, press Ctrl-PrtSc to start printer echoing, and enter

A>HTEST

If you have two hard disk drives, add the name of the drive to be tested. To test drive D:, for example, enter

A>HTEST D:

Press "N" when the program asks "Do you wish to change any of the parameters" and press "Y" in response to "Do you wish to start the testing?" HTEST performs three quick checks of the controller, and its results should look like this:

- Performing drive recalibrate...
- Drive recalibrate performed successfully
- Performing internal drive controller test...
- Internal controller test performed successfully
- Performing internal drive reset test...
- Internal drive reset performed successfully

If the controller passes these tests you can halt HTEST by pressing the SPACE bar and then pressing "A" (abort) to return to DOS. But if you receive any errors with these tests then your controller or drive needs professional help.

Invalid Drive or Partition Table

If you turn on your computer and, after its power-up tests, it displays either of the following messages:

- Invalid drive specification
- Invalid partition table

no amount of pleading or cursing will allow you to read your files. But don't despair. The HTEST/HFORMAT programs are designed to deal with this problem, which can also happen if you've booted with a diskette and you then try to change over to C:.

These messages indicate that the system can't read the master boot record (MBR), and so it can't find the DOS partition. Either the MBR is garbled (which is not hard to fix) or else the track or sector marks in cylinder zero are bad (this requires a little more work), or the drive can't accurately locate cylinder zero (this is serious).

Before You Panic

If the room is colder than normal (or if it was, and the computer's still cold) then the drive may be seriously out of calibration. Let the computer idle for an hour while it and the room warm up, and then try restarting it. Often this is all that's needed to get the drive back. Now back up everything and use HTEST to check the disk, especially cylinder zero. What you learn from this near-catastrophe is to keep the computer comfy, because large temperature swings will affect tolerances and can change the size of the disk surfaces enough to make the disk unreadable.

If temperature is not the problem then start the computer with a DOS diskette in drive A:. If you have an AT machine, use the DOS diskette that has the program SETUP.EXE on it and run the program by entering

A> SETUP

Check your drive type against the one shown by the program (your computer or disk manual, or your dealer, will have the correct value). After you've corrected (or accepted) the setup and the program ends, try to reboot the computer with no diskette in A:. If you succeed, offer a prayer and check your computer's battery.

If you originally partitioned the hard disk with a special program such as Speedstor, Vfeature, or Disk Manager, the trouble may lie with that program's device driver, which is needed to read the disk. If you have important files in the higher partitions try using the partitioning program's diskette to install the device driver on A:. If you can do that, you may be able to read some of the partitions and back up your files before proceeding.

The Easy Way Out

If you still can't get the hard disk to boot, and if you have a current, correct, and complete backup of your files, you can take the direct approach: wipe out everything and start over.

If you have an MFM controller:

Use HFORMAT (from the diskette) to reformat and test the disk. See "Format a Hard Disk" on page 1-2, and start formatting at cylinder zero, head zero.

If you have an Auto-configure (RLL) controller:

You will need to use its own formatting routine. If the disk manual says nothing about low-level formatting, or tells you that the disk must be shipped back to the factory to do it, try the following. Run DOS's DEBUG (it's on one of the DOS diskettes) by entering

A>DEBUG

and, when the "_" prompt appears, try to transfer control to the address where the formatter should be located by entering

-G=C800:5

Warning: this will destroy all information on the disk. The command works on Western Digital controllers. If you have another type, and you really do want to reformat the disk, call for help.

After formatting:

Use XFDISK to create partitions, and use DOS FORMAT or Mace Utilities FORMAT-H to prepare the DOS partition. See "Create and Save a Partition Table" on page 1-22. Finally, restore your programs and files.

Again, be certain that your backup is a good one before taking this approach. I don't trust any backup system (soft or hardware based) unless I've used it before and have been able to restore files with it.

Before We Start

If you aren't familiar with the concept of partitions and boot records, or with sectors and surfaces, read the introduction to hard

disks in Chapter four before beginning the recovery. The knowledge isn't essential, but it will help you proceed with confidence. I'll assume that you've booted from a DOS diskette and that you've put the HTEST/HFORMAT diskette in drive A:

Things may be a lot easier if you have a diskette copy of the hard disk's master boot record (MBR), created with XFDISK or GET-SEC. If not, but you can find a machine that has the same controller and drive and the same partitions as yours, use XFDISK to copy its MBR onto a diskette, and take that diskette back to the problem machine. If you have this MBR copy, begin at step two. Otherwise, begin at step one.

Attention autoconfigure (most RLL) controller owners:
Please read the note on page 11 before doing anything. It will help you decide whether to attempt this recovery procedure.

STEP 1: Creating a Copy of the MBR

Start XFDISK by entering

A>XFDISK

and try to display the partition table by selecting choice 4 from XFDISK's menu. The program will try to read the table from the disk's first sector, and if it reports any errors, you'll need to reformat. Press ESCAPE to quit XFDISK, and jump to step three.

If XFDISK can display the table, check that there is a DOS partition starting on cylinder zero (or perhaps cylinder one) and that it

shows "BOOT" in the "Status" column. If there is no partition, use choice 1 on the main menu to create a partition for your DOS version, starting on cylinder zero unless you know that it started further in, and make it as large as possible if you don't know how big it was. Use choice 2 to activate this partition, and then use choice 4 to display it again.

Once the table looks okay, choose main menu choice 7, and then choose option 1, to save it to diskette with the default filename HD0BOOT.REC (the program writes to the default drive., which is A: in this case). Now continue with step two.

STEP 2: Placing the MBR on the Disk

If you used XFDISK in step one to create an MBR, use main menu choice 7, option 2, to write it directly to the disk, and watch for error messages. If XFDISK reports errors in writing to the hard disk, quit the program and jump to step three.

If you had a diskette copy of the MBR before the trouble began, you can write it onto the disk with main menu choice 6. Change the default filename if you saved the MBR under a different name. Again, jump to step three if XFDISK has trouble writing to the hard disk.

If you can write the MBR (either a new or original version) without errors using either of these approaches, reboot the computer with no disk in A:. If C: works, jump to step seven. If you still get the "invalid drive" or "invalid partition table" message, continue with step three. If you get a different error, the partition may be damaged or defined incorrectly; see "Operating System Startup Failure" on page 2-16 for help.

STEP 3: Saving Track Zero and Cylinders 0-4

In an attempt to make the master boot record readable, we're going to reformat the track so it can be written and read reliably. If we can't do that, there are alternatives, although they're not simple.

First, use GETSEC to save just the track under the MBR. To make sure it fits on the diskette, use a blank (but formatted) diskette, copy GETSEC.COM and PUTSEC.COM onto it, and at the DOS prompt enter

```
A>GETSEC A:ZERO.TRK /IC=0, MC=0, IH=0, MH=0, IS=1,  
MS=17, D=C
```

GETSEC will try to copy all the sectors from track zero on surface zero. It will pad the image file ZERO.TRK with zeros for any sectors that it can't read, and will report the error.

Now you need to know how many heads, and how many sectors per track, your drive has. If you don't know, run XFDISK again, select main menu choice 5, and make a note of the values. Then press ESCAPE to quit XFDISK, and use GETSEC to save cylinders zero through four a cylinder at a time. For cylinder zero you will enter:

```
A>GETSEC A:ZERO.CYL /IC=0, MC=0, IH=0, MH=5, IS=1,  
MS=17, D=C
```

Warning: this command assumes that there are six heads (zero through five) on your drive, and 17 sectors per track. Make sure that you use the correct "MH" value (one less than the number of heads) and "MS" value (the number of sectors per track). Make a note of any sector errors that GETSEC displays: they indicate tracks that may need reformatting.

To save the next cylinder (cylinder one) enter

```
A>GETSEC A:ONE.CYL /IC=1, MC=1, IH=0, MH=5, IS=1,  
MS=23, D=C
```

Increment the image file's name (TWO.CYL, THREE.CYL, FOUR.CYL), and the "IC" and "MC" options, to save cylinders one through four (or more, if you want extra insurance) and remember to use the correct "MH" and "MS" values. Again, make notes of any sector errors that GETSEC displays.

If you use one diskette for each cylinder you won't have to worry about running out of room, and you should copy GETSEC.COM and PUTSEC.COM onto each diskette to keep things easy. After saving the cylinders, go on to step four.

STEP 4: Reformatting under the MBR

Now to format the track. For MFM controllers HFORMAT makes things easy. For RLL controllers things are a little rougher, but don't despair.

If you have an MFM controller:

Put the HTEST/HFORMAT diskette back in drive A: and enter

```
A>HFORMAT /B=0,0 , E=0,0 , NV
```

The options tell HFORMAT to begin at cylinder zero, head zero, and end at cylinder zero, head zero, so that the only track formatted will be the disk's first one, where the master boot record lives. The "NV" option tells HFORMAT to just format the track, not to test it.

HFORMAT displays a list which defines the area to be formatted. Check that the beginning and ending cylinders and surfaces are zero and that "format retry count" is also zero. Press "N" when HFORMAT asks "Do you wish to change any of the above testing parameters?" unless you need to correct these values.

Press "Y" when HFORMAT asks whether to proceed and press "Y" again when it asks whether you're certain. As HFORMAT tests the track it will display a list of "return codes" (errors). When HFORMAT finishes testing press "A" to abort (stop) the program and, if it found no errors, proceed with step five.

Any formatting errors on this track are serious. You can try running HFORMAT again, but if that fails, and you're determined to salvage your files, call us for advice.

If you have an Auto-configure (RLL) controller:

The approach is a lot less delicate. Since the original format wrote special drive information *between* sectors on the first track (cylinder zero, head zero) we have to use the same routine to reformat that track, and then stop the program before it does any harm.

For autoconfigure controller (most RLL) owners only: before you proceed, you should be aware that the approach outlined here (known as the "big red switch" game) is Paul Mace's approach and it's risky, no doubt about it. It exposes your files to the possibility of a low-level and completely destructive format. Of course, if you didn't need to recover files from the disk we wouldn't suggest it. Call us for advice on your alternatives — or for companionship while you try it.

If your computer or disk manuals don't tell you how to start the controller level format program, try the following approach. Run DOS's DEBUG (it's on one of the DOS diskettes) by entering

A>DEBUG

and DEBUG's "-" prompt will appear. You are going to transfer control to the address where the formatting program is stored on most RLL controller cards. When that program starts, it should display some information and ask you to choose or accept some options before starting to format the disk. Unfortunately, you can't tell it to format only certain cylinders, so you're going to have to start the process and then IMMEDIATELY kill the power. Enter the following line:

-G = C800:5

and, if the format program begins, accept its defaults (otherwise, call for help). Now take a deep breath, put one hand on the power switch, press ENTER with the other hand and immediately turn off the power. Given the speed of this program, the first track will almost certainly have been formatted. If you keep your eyes on the monitor, you might see some indication of the program's progress before the power drops, but don't delay hitting the switch. If PUTSEC (in step five) still can't write to the first track you can always try this again and cut the power a fraction less quickly. Now continue with step five.

STEP 5: Restoring the Formatted Area

If you have an MFM controller:

Use PUTSEC to replace the track you saved with GETSEC. Enter

```
A>PUTSEC A:ZERO.TRK /IC=0, MC=0, IH=0, MH=0, IS=1,  
MS=17, D=C
```

which matches the options used with GETSEC in step three. If PUTSEC works, continue with step six, but if it reports errors then the reformat didn't work. You can try step four again, or you can call us for help.

If you have an Auto-configure (RLL) controller:

If you have an auto-configure controller then use PUTSEC to replace each of the cylinders you saved. Refer back to step three and use the same options that you used with GETSEC, and remember to use the correct "MH" and "MS" values. PUTSEC may find that the formatter left a track unreadable when it stopped in mid-stride, so add any new sector errors to your list of GETSEC errors from step three. You can repair individual tracks or cylinders later; for now, continue with step six.

STEP 6: Restoring the MBR

Now we need to replace the disk's master boot record. Whether you had saved a copy of the MBR before the trouble began, or you created a new file in step one, start XFDISK again and use main menu choice 6 to write it to disk. Change the program's default filename if your MBR file isn't called HD0BOOT.REC, and add a pathname if it's not in the root directory of A:.

If XFDISK has any trouble writing the MBR, it will let you know. In that case, you can repeat the reformat in step five and try writing the MBR again, but a disk whose first cylinder has format or read/write trouble may not be usable without difficult (and expensive) service, if ever. Call us for advice.

But if XFDISK reports that it wrote the file without errors, you're almost home. Take five, then continue with step seven.

STEP 7: Recovering the C: Partition

Open drive A: and try restarting the computer. If it loads DOS from C:, congratulations! If you still get an invalid drive or partition table message, call us. You might also get a different error message, if the DOS partition has suffered any damage.

If there are problems with other cylinder zero tracks and you have a DOS partition starting on cylinder zero you'll still be getting operating system errors, and you will probably lose some information. You can use the GETSEC/HFORMAT/PUTSEC cycle in steps three through five to reformat individual tracks or entire cylinders (which you've already saved) as necessary, as long as you stay away from track zero, cylinder zero on RLL controllers.

If the tracks are good, but you have operating system errors, turn to "Operating System Startup Failure" on page 2-16 for advice.

Finally, assuming you've regained the use of your disk, check your directories and files for damage and, if necessary, use Mace Utilities UNFORMAT to recover them. But if you had higher partitions that aren't responding, see step eight before doing anything. And before you quit, back up the disk and use HTEST to check cylinder zero. See page 1-8 for instructions on running HTEST.

STEP 8: Recovering Higher Partitions

If you recovered by using XFDISK to construct a new partition table, you may or may not have specified the correct beginning and ending cylinders for each partition. Until you've defined them all correctly, don't save anything to the disk, or one partition could overwrite another one's files.

The first (C:) partition is the easiest to find. It probably starts at cylinder zero, unless you created it with XFDISK or a similar program and started it at cylinder one. But does it end with the cylinder that makes it DOS's maximum usable size of 32 Megabytes, or does it end somewhere else?

In fact, it's easier to define it much smaller, once you've backed up the C: files, and then start looking for the beginning of the next partition, while assuming that it ends on the last cylinder. This approach requires running XFDISK in the same way you used it to define and recover the C: drive. Use the program to move the beginning cylinder forward (or back) until a partition works (that is, you can view the root directory and files). Don't forget to reboot each time you redefine the partition table, to make the changes take effect.

Operating System Startup Failure

If you turn on your computer and, after its power-up tests, it displays one of the following messages:

- Non-system disk or disk error
- Missing operating system
- Error loading operating system
- Bad or missing command interpreter

it's either because the partition table shows no active (bootable) partition, or else because the active (C:) partition's operating system has suffered some damage.

The difference between these errors and the "Invalid drive specification" or "Invalid partition table" messages is that the computer can read the partition table, but it can't find an operating system to load. If you restart the computer with a DOS diskette in drive A:, you should then be able to read drive C:, and of course the first priority is to make sure you have current backups of all your work.

If you're unfamiliar with the concepts of boot records and FATs, read the introduction to hard disks in chapter four before beginning this recovery. The knowledge isn't essential, but it will help you proceed with confidence. I'll assume that you've booted from a DOS diskette and that you've put the HTEST/HFORMAT diskette in drive A:.

Checking the Partition Table

The first step is to check that there is a DOS partition marked active in the partition table. Start XFDISK by entering

A>XFDISK

and select choice 4 from the program's main menu to display the partition table. You should see a DOS partition marked "BOOT" in the status column. If not, use main menu choice 2 to activate the DOS partition and then use choice 7 to save the file to diskette, modifying the program's default filename to read

A:HD0BOOT.REC

If you need help running XFDISK, see page 1-22. Use choice 7 again to write the new table onto the disk, and then quit FDISK. Now cross your fingers and restart the computer.

Note: if you had saved HD0BOOT.REC before this, you should first try putting it back on the disk with XFDISK, using main menu choice 6.

If you just recovered from an "Invalid drive specification" error and you're having these problems, you should also try altering the partition's starting cylinder, as outlined on page 2-15.

If the partition table looked good, or if you activated the partition and the drive still won't boot, keep reading.

Replacing the System Files

The next step is to replace the system files in the C: partition. The only problem here is that they must be entered into the first two places in the root directory, or else DOS cannot load itself. When the partition was first created and formatted, these files were copied into their proper places and then hidden from sight, so

there should be room for them unless some errant program has managed to save something else on top of them.

Place the DOS diskette in drive A: and enter

A>SYS C:

To copy the system files into the C: partition. If this routine gives you any problem, you can try deleting files from the root directory to make room for the system files, or give up and reformat the partition. If SYS works, continue with the next step.

Replacing COMMAND.COM

The last step is to put a good copy of COMMAND.COM into the partition. With the DOS diskette in drive A: enter

A>COPY A:COMMAND.COM C:\ /V

to copy (and verify) the command processor from the diskette. By the way, you can use SYS and COPY COMMAND.COM to upgrade a disk to a new DOS version if the system files will fit. Now try rebooting on C:.

Taking The Extreme Approach

If the computer still can't load DOS from C:, or if it now displays a "File allocation table bad" message, you can try reformatting it as outlined in "Bad Directory or Fat" on the next page.

Bad Directory or FAT

If you turn on your computer and it's able to load DOS from the hard disk, but then you (or your programs) try to read the disk and DOS responds with either of the following:

- File allocation table bad
- General failure reading drive C:

then something has scrambled the partition's structures. These include the file allocation table, or FAT, the root directory, and the partition's boot record, which sit in the first two or three cylinders of the partition. This problem can also plague higher (DOS 3.3 extended) partitions.

The damage isn't as drastic as it sounds, however. Mace Utilities UNFORMAT can rebuild all these DOS structures and recover your files for you. See the Mace Utilities Recovery (red) manual for instructions.

If the UNFORMAT procedures won't work, it's possible that there is underlying damage to the tracks that hold these structures. If you don't know where your partition starts, run XFDISK to view it:

```
A>XFDISK
```

and select main menu choice 4. Note the starting cylinder for the partition, quit XFDISK, and run HTEST on the first four cylinders. For example, if the partition starts on cylinder one, and there are four heads on the drive, use:

```
A>HTEST C: /B=1,0 , E=4,3 , L=10
```

This will test cylinders one through four and surfaces zero through three, and will repeat ("loop") the test ten times. Change the "3" to the number of the last surface (head) on your drive. If HTEST finds any problems in the first few cylinders of the partition, you can use Mace Utilities REMEDY to move files out of bad sectors. If the damage is widespread, consider a more extreme solution, such as moving the partition (after backing it up, of course). See "Create and Save a Partition Table" in the maintenance guide.

One last hint if you haven't found any reason for the damage: sudden and unexplained destruction of FATS and root directories or is a strong symptom of a program virus. Contact Paul Mace Software for information on MACE VACCINE, which is designed to prevent intentional or accidental mischief by sick programs.

Programs Crashing Under DOS 3.3

If you use IBM Personal Computer DOS version 3.30 or above and you are using a non-IBM computer or hard disk controller you may find that some programs lock up during disk reads or writes, requiring you to reboot the computer. Unfortunately, these DOS's assume that all controllers can handle multiple-track requests, in which DOS asks the controller to read or write more than one track's worth of information at a time. If the controller can't handle the request, the system grinds to a halt.

33TRAP is a resident program designed to cure this problem by converting all multiple-track read, write, and verify requests into a series of single track requests. If you use a version of DOS below 3.3 or you don't have programs locking up on you then you don't need 33TRAP. If your programs ran without trouble until you upgraded to DOS 3.3 or greater, then you definitely need it.

You must run the program C33TRAP.EXE to create a 33TRAP program tailored for your computer. At the DOS prompt, enter

```
A> C33TRAP
```

and the program will take a few seconds to automatically create the program, which is stored on the diskette in a new file called 33TRAP.COM. Note that this 33TRAP program will not work if it is simply transferred to another computer; it must be created by C33TRAP for the machine on which it will run.

Copy the new program onto the hard disk, either into the root:

```
A> COPY 33TRAP.COM C:\ /V
```

or into an appropriate subdirectory (DOS, for example):

A>COPY 33TRAP.COM C:\DOS /V

You don't need to copy C33TRAP.EXE onto the disk; its work (for this machine) is done.

To start 33TRAP enter its name (preceded by drive and path names, if necessary) at the dos prompt. For example,

C:\DOS> C33TRAP

You can automate the process by including this command in your AUTOEXEC.BAT file so that 33TRAP starts whenever you turn on your computer.

Chapter 3: Program Reference

This chapter contains an entry for each of the HTEST/HFORMAT programs, and supplies the following information for each:

- Program Name
- Purpose (a brief functional description)
- Command Syntax (including optional parameters)
- List of Options (a brief description of each)
- Program Description (details of program operation)
- Command Examples
- Conflicts (notes on hardware and software problems)

Command line syntax uses the following conventions:

[option]	Items in square brackets may be added to the command line but are not required.
{choice choice}	Items enclosed in braces and separated by vertical bars represent mutually exclusive choices. If the choice list is enclosed in square brackets then the selection of a choice is optional.
m,n	"m" and "n" represent the need to enter an integer value in decimal notation.

You can enter any option in upper or lower case; you must use a slash ("/") before the first option, and you must separate options with either slashes or commas. You can add spaces between options except where noted, and the order of options is not important. You can specify an option with either its abbreviation or its long form.

This chapter assumes a familiarity with the physical and logical organization of hard disks. Chapter four supplies that information for those new to hard disks.

CRC

Purpose

CRC checks for alterations to a file by computing and displaying unique values based on the file's contents.

Syntax

CRC [drive]file

Options

- | | |
|-------|--|
| drive | Indicates which drive to search for the file(s) specified; must be a valid DOS drive name (letter plus colon). The default value is the current drive and the (sub)directory searched is always the current one. |
| file | Specifies the file to use in computing CRC values. Must be a valid filename and optional extension. The wildcard characters (?) and (*) may be used to extend the command to several files at once. |

Description

CRC looks at the contents of a file as a long bit string representing a polynomial, or multiple-power equation, which has a unique value. The program uses the standard international CCITT method, and also uses a shorter CP/M method as a second check.

The results of both checks are sufficiently dependent on the value of each bit in the file to serve as a valid check of the file's contents. If two runs of CRC against the same file produce differing results then the file's length or content has changed.

Note that the file specification for CRC does not accept subdirectory paths; the files specified must be in the current directory. If the drive is specified there must be no spaces or other characters between the drive name and the file name.

Examples

CRC *.*

Compute and list CRC values for all files in the current drive and directory. If run against the HTEST/HFORMAT diskette the results can be compared to the file CRC.LST as a check on the diskette's integrity.

CRC A:LETTER21.DOC

Compute and list CRC values for the file LETTER21.DOC in the current directory of drive A:.

GETSEC and PUTSEC

Purpose

Transfer the information on a sector or a range of sectors into a disk file (GETSEC). Transfer from a file onto a range of sectors (PUTSEC).

Syntax

GETSEC [drive]file /IC=n, MC=n, IH=n, MH=n, IS=n,
MS=n, D=X

PUTSEC [drive]file /IC=n, MC=n, IH=n, MH=n, IS=n,
MS=n, D=X

Options

- | | |
|-------|--|
| drive | Indicates the drive on which GETSEC will create an image file, and on which PUTSEC will look for it. It must be a valid DOS drive name (letter plus colon). If this option is not used, GETSEC and PUTSEC assume the current drive by default. |
| file | Supplies the name of an image file in which GETSEC will save, and from which PUTSEC will read, the sector contents. It must be a valid DOS filename (eight characters plus an optional period and three-character extension). No path may be given, and the current directory of the specified or default drive is always used by default. If the drive is specified then no spaces may separate the drive and file names. |
| IC=n | Specifies the Initial Cylinder of a range. |
| MC=n | Specifies the Maximum Cylinder of a range. |
| IH=n | Specifies the Initial Head of a range. |

MH = n	Specifies the Maximum Head of a range.
IS = n	Specifies the Initial Sector of a range.
MS = n	Specifies the Maximum Sector of a range.
D = x	Specifies the drive to use, using a letter (no colon).

Description

GETSEC creates a copy of specific sectors of the disk and saves it as a disk file, on the same or another disk. PUTSEC reverses this operation when it reads a file and copies its contents onto a specific area of the disk.

The file that GETSEC creates, and that PUTSEC reads, is called the image file, and is specified on the command line. If GETSEC finds that the image file already exists, it will ask whether to erase it. Press "Y" to continue, or "N" to stop the program. If PUTSEC cannot find the image file it will indicate that fact and quit.

The command options also define a set of sectors, called the sector group, that is the source (GETSEC) or target (PUTSEC) for the transfer. This group is the intersection of the three ranges specified on the command line. For example, the options

IC=0 MC=4 IH=2 MH=2 IS=1 MS=2

define a group composed of sectors one and two of tracks zero through four on surface two. The sector group can be defined as narrowly as a single sector, or it can be as large as the available storage on the image file's disk.

Aside from asking whether the image file and sector group are correct, PUTSEC does not issue warnings or offer a chance to avoid

overwriting disk information. **Important:** Check the sector group carefully before instructing PUTSEC to proceed.

Both programs begin by listing the image file and the command options, and asking whether the displayed parameters are correct. Press "N" to stop the program and return to DOS, or press "Y" to begin the transfer. The programs list cylinder, head, and sector numbers (in hexadecimal) as they work through them, and will display any read or write errors that occur.

If either program receives an error when reading or writing a sector it will try up to twenty times to succeed before giving up and moving to the next sector in the group.

If GETSEC cannot read a sector it will pad the image file with null (ASCII value 00) characters. This insures that the image file's length is equal to the requested number of sectors. If PUTSEC later copies this image file onto the same sectors it will either write the zeros in that problem sector or, if it cannot write, it will skip both the zeros and the sector. As a result, PUTSEC avoids the possibility of shifting sector contents among the group.

Examples

```
GETSEC A:HD0BOOT.REC / IC=0, MC=0, IH=0, MH=0,  
IS=1, MS=1, D=C
```

Copies sector one of the first track (cylinder zero, head zero) of the (first) hard disk to the file HD0BOOT.REC, thereby saving a copy of the drive's master boot record (MBR). This command can be saved on the diskette as a one-line DOS batch file named GET-MBR.BAT. When the batch is executed (enter "GET-MBR" at DOS's prompt) it will call GETSEC with the correct options.

PUTSEC A:PARTN.CYL / IC=0, MC=3, IH=0, MH=5, IS=1,
MS=17, D=C

Writes a copy of the file PARTN.CYL onto the hard disk. Assuming the drive has six heads and seventeen sectors per cylinder, PARTN.CYL will include all sectors of cylinders zero, one, two, and three, thereby saving the first partition's FATs and root directory.

HFORMAT

Purpose

Performs physical (controller, low-level, absolute) formatting of all or part of a hard disk, and can be run non-destructively. Tests each track and writes bad track marks as needed. Accepts (and marks) additional bad track list from keyboard.

Syntax

```
HFORMAT [drive] [/BATCH], [B = m,n], [E = m,n], [FT = n],  
[I = n], [M = {XT|AT}], [MC = n], [MH = n],  
[MS = n], [NBD], [NF], [NV]
```

Options

drive Indicates which drive to format; must be either C: (physical drive zero) or D: (physical drive one), regardless of partitions and their names. If only one hard disk is connected, HFORMAT ignores this option. The default value is C:

BATCH (no abbreviation): suppresses all HFORMAT questions requiring keyboard response, allowing formatting to begin immediately using the options and values specified (or defaults where necessary). By removing all interaction with the user, the BATCH option allows HFORMAT to run in automated disk setup routines. Only experienced disk preparers should use this option.

B = m,n **BEGIN**: defines the first cylinder (m) and first surface (n) of the area formatted. The HFORMAT default values are cylinder one (not cylinder zero) and surface zero. **DO NOT FORMAT SURFACE**

ZERO, CYLINDER ZERO OF DISKS WITH RLL CONTROLLERS.

- E = m,n** **END:** defines the last cylinder (m) and last surface (n) of the area formatted. The HFORMAT default values are the numbers of the disk's last cylinder and last surface.
- FT = n** **FMT_TRIES** (format tries): specifies the number of times to check the newly formatted area by writing and verifying a test pattern, changing the pattern with each repetition. The default value for repetitions is three; the minimum is zero, which preserves information on the disk, and the maximum is 9999.
- I = n** **INTERLEAVE:** specifies the interleave to create when writing sector information. The default value for a PC or PC-XT is six; the default value for a PC-AT is two. The minimum interleave is one and the maximum possible interleave equals the number of sectors per track minus one. An HFORMAT interleave of N is equal to a Microsoft "interleave factor" of 1:N.
- M = xx** **MACHINE:** defines the controller type. XT and AT are the only acceptable values. If this option is not supplied on the command line, HFORMAT will examine the controller and determine this value.
- MC = n** **MAX_CYL:** overrides the controller's value for the number of cylinders on the drive. The value must be an integer from 1 to 1024.
- MH = n** **MAX_HEAD:** overrides the controller's value for the number of heads on the drive. The value must be an integer from 1 to 16.

MS = n	MAX_SEC: overrides the controller's value for the number of sectors per track on the drive. The value must be an integer from 1 to 63.
NBD	("No Big Deal"): manipulates the controller through DOS's routines instead of directly. Used if HFORMAT has trouble locating the controller program (BIOS). This option has no long form.
NF	NO_FORMAT: directs HFORMAT to avoid writing track and sector information to the disk; proceeds directly to the track testing routine. Also suppresses the writing of bad track information if errors are found. This option does not suppress the initial HFORMAT options display and its cross-check before proceeding. Note that any track testing will still destroy information on the disk.
NV	NO_VERIFY: skips all testing of the formatted area, although HFORMAT will still accept additional bad track numbers and mark them unusable. If the NV switch is used then Format Tries equals zero and the program will perform a non-destructive format of each readable track in the range, copying and rewriting information as it works. It will not format tracks that it cannot read.

Description

HFORMAT physically formats all or part of a hard disk, writing the track and sector information needed for partitioning and logical formatting. Most DOS packages do not include this function; IBM only provides it as part of a Hardware Maintenance and Service package. In addition, HFORMAT (if run with FT=0) will lift and replace information on the disk, preserving all existing files.

HFORMAT also performs extensive tests of the surface. It writes information onto the entire newly formatted area and then reads and compares (verifies) the information. The FT option specifies the number of times to repeat this test. HFORMAT marks a track unusable if the testing of any sector in that track causes the controller to signal an error. The bad mark prevents a track's use by any operating system.

Following testing, HFORMAT allows the user to identify bad tracks found in other tests and adds bad track marks to them. If both the NF (no format) and NV (no verify) options are used, HFORMAT limits itself to this "add bad tracks" routine and accepts and displays bad tracks, but does not actually write them.

HFORMAT recalibrates the drive (by seeking cylinder zero) between operations, guaranteeing that the controller can read each cylinder independently. HFORMAT recalibrates after formatting each track, after sensing any track error, and after writing every bad track mark.

HFORMAT begins by locating the hard disk controller program (BIOS) normally hidden from direct access by DOS. It displays the message "signature value x,y" to identify the beginning of the controller's input-output routines.

The program next lists its format parameters, using defaults as necessary, and offering to edit them. If directed to change these options, HFORMAT begins presenting each one in turn, allowing them to be accepted or changed. After presenting all options for review, HFORMAT redisplayes the format parameters as revised. This edit-display cycle can be repeated indefinitely.

In the BEGIN and END options, the specification of surfaces (heads) is independent of the specification of cylinders. Specifying

the same surface in both BEGIN and END limits HFORMAT to one surface, while specifying the same cylinder and surface in both options limits HFORMAT to a single track.

If "N" is pressed in response to "change testing parameters" HFORMAT issues its only warning and asks twice whether to proceed. Pressing "Y" at both prompts allows HFORMAT to begin; pressing "N" at either prompt causes it to quit and return control to DOS.

Before HFORMAT begins formatting, it resets the controller and directs it to recalibrate the drive (find cylinder zero). This assures that the controller and drive will begin laying out tracks correctly. If the program encounters any problems with these tests it will display them and ask whether to continue with formatting, as these problems usually indicate a need for disk or controller service.

HFORMAT displays cylinder and head numbers as it formats. Once it has formatted the disk it begins testing, displaying errors as they occur and creating pages (screens) of error reports. Each error report takes the form

Cylinder	Head	R/C
-----	-----	-----
245	1	80h

where the cylinder and head are in decimal and the return code (R/C) is in hexadecimal. The program resets the controller after each error.

HFORMAT suspends work if the space bar is pressed (it may take a moment to finish a pass before responding). When HFORMAT is interrupted (and after it has finished formatting and testing) it displays a menu of options which can be selected by single key-

strokes. These choices allow the user to page through all error reports, call up an explanatory list of return codes, resume running HFORMAT, or exit the program and return to DOS. When HFORMAT has finished, it adds the "add bad tracks" option to this menu.

The "add bad tracks" routine asks for a cylinder and a head number (in decimal). For each track entered, it writes a mark (unless HFORMAT was invoked with the NF option) that will prevent the operating system from using the track. HFORMAT ends this routine in response to a null entry and again displays its menu.

Examples

HFORMAT

Formats the hard disk using all program defaults. Formats all but cylinder zero using the appropriate default interleave, and tests each track three times.

HFORMAT D: /NV, I=3

Formats the second hard disk to an interleave of three, lifting and replacing information track by track. HOPTIMUM measures disk performance under different interleaves and will suggest the best value to use. The HFORMAT default might be the best choice, but some drives and machine combinations will benefit from other values. Don't run this benign reformat without a backup!

HFORMAT /NF, FT=32

Skips the formatting (adding of track and sector marks) and begins testing drive C:. Repeats the test 32 times, reporting errors as they

occur, but does not write bad track marks. Note that this testing destroys all information on the disk.

HFORMAT /B = 100,2 , E = 100,2 , FT = 20

Formats only the track on cylinder 100 under head 2 and tests it 20 times.

Conflicts

Important: Do not use HFORMAT on surface zero of cylinder zero if you have an auto-configure controller. Most RLL controllers are of this type, and they rely on special information written between sectors in this first track. HFORMAT's default starting cylinder is one, not zero; use extreme care when changing this value.

Before starting HFORMAT, remove any BUFFERS, FILES, or FCBS statements from the CONFIG.SYS file and reboot. Also remove all resident programs from memory.

HFORMAT works on IBM and IBM-compatible PC, XT, and AT computers with standard MFM or RLL controllers and disks with up to 16 heads and 1024 cylinders. Detailed information is contained in the supported hardware list on page viii.

If HFORMAT locks up the computer when it starts, reboot and add the NBD option to the command line (it cannot be invoked within the program). This allows HFORMAT to use DOS's interfaces instead of trying to locate and talk directly to the controller. It will slightly lessen HFORMAT's error-checking abilities.

Certain drive and controller combinations may lock up after read/write errors or after bad-track marking; use the NV option to skip HFORMAT's testing routine.

The MAX_CYL, MAX_HEAD, and MAX_SEC options may allow HFORMAT to format a drive that is not recognized by DOS's input-output routines. The values supplied must reflect the actual number of cylinders, heads, and sectors on the drive, and the controller must allow software redefinition of its disk parameters. These overrides will last during HFORMAT only; HFORMAT does not make an otherwise unsupported disk usable.

The NF (no format) option will allow HFORMAT to use its powerful testing abilities on drive and controller combinations which use special formatting routines included with the controller (ESDI controllers, for example). The problems found by HFORMAT can be locked out after the controller's own format, during its "add bad tracks" routine.

Important: The MACHINE option is necessary with certain computers, such as the Toshiba 3100, which use XT controllers in AT style machines. Specify the *controller* type. This value can be changed while editing the format parameters.

Important: Do not use HFORMAT's non-destructive format without a current and correct backup of all important hard disk files. Its careful approach to moving large numbers of sectors is no guarantee against mishaps such as voltage sags or disk problems.

HOPTIMUM

Purpose

Evaluates the speed of disk operations under different interleaves. Reformats disk non-destructively to the best interleave.

Syntax

HOPTIMUM [drive] / [B = m,n], [E = m,n], [BI = n], [EI = n],
[M = {XT|AT}], [MC = n], [MH = n], [MS = n],
[SA = n], [{R|W}], [{T|S}]

Options

- | | |
|---------|---|
| drive | Indicates which drive to evaluate; must be a physical drive number (0-9) followed by a colon. If only one hard disk is connected, HOPTIMUM ignores this option. The default value is zero (the first physical drive). |
| B = m,n | BEGIN: defines the first cylinder (m) and first surface (n) of the interleave test area. The HOPTIMUM default values are the disk's next to last cylinder and first surface (surface zero). |
| E = m,n | END: defines the last cylinder (m) and last surface (n) of the interleave test area. The HOPTIMUM default values are the disk's next to last cylinder and first surface (surface zero). Note that some drive manufacturers leave the last track unformatted for diagnostic use. |
| BI = n | (Beginning Interleave): specifies the first of the range of interleaves to evaluate. The default value (and minimum) is one, and the maximum is one less than the number of sectors per track. Note that the |

normal values of six (for XT's) or two (for AT's) are most likely to be optimum. This option has no long form.

- El = n** (Ending Interleave): specifies the last of the range of interleaves to evaluate. The default value (and maximum) is one less than the number of sectors per track, and the minimum is one. This option has no long form.
- M = xx** **MACHINE**: defines the controller type. XT and AT are the only acceptable values. If this option is not supplied on the command line, HOPTIMUM will examine the machine and determine this value.
- MC = n** **MAX_CYL**: overrides the controller's value for the number of cylinders on the drive. The value must be an integer from 1 through 1024.
- MH = n** **MAX_HEAD**: overrides the controller's value for the number of heads on the drive. The value must be an integer from 1 through 16.
- MS = n** **MAX_SEC**: overrides the controller's value for the number of sectors per track on the drive. The value must be an integer from 1 through 63.
- SA = n** (Samples): specifies the number of times to read or write all tracks in the test area (in sequential order) as an evaluation of each interleave. The default value is 250; the minimum is 1 and the maximum is 65,535. The sample size affects HOPTIMUM's run time, but small samples (under 100) may contain enough variance to skew the results. This option has no long form.
- R** **READ**: directs HOPTIMUM to evaluate interleaves using only read requests. If neither READ nor WRITE is specified then HOPTIMUM READs by

default, preserving all information on the disk. As normal disk use consists of ninety percent reads, this is also the most accurate method of testing.

- W** **WRITE:** directs HOPTIMUM to evaluate interleaves using write requests, destroying information in the test area. The program will not, however, write to cylinder zero.
- T** **TRACK:** directs HOPTIMUM to evaluate interleave performance with requests for entire tracks at a time, simulating the normal loading or saving of files under DOS. If neither SECTOR nor TRACK is specified then HOPTIMUM conducts TRACK tests by default.
- S** **SECTOR:** directs HOPTIMUM to evaluate interleave performance with separate sector requests. This approach may yield better results for applications which do not use standard DOS requests.

Description

HOPTIMUM performs two major tasks: it analyzes disk performance under different interleaves, and it reformats the entire disk to the relative best of the interleaves tested. In pursuit of these goals it will:

- check whether any resident programs are running
- display the characteristics of the drive to be tested and correct them if necessary
- display its proposed interleave testing options and change them as directed
- determine the current interleave and transfer rate of the disk
- determine the transfer rate for interleaves in the test range
- report the performance gain possible under the best interleave

- reformat the entire disk at the best interleave

HOPTIMUM conducts an extensive conversation with the user, always checking whether to proceed with the next task and displaying all results onscreen. In particular, HOPTIMUM asks whether to proceed before beginning its interleave tests and before beginning a reformat to the best interleave, and it will quit and return to DOS unless directed to continue. It pauses frequently, allowing use of the Shift-PrtSc key combination to print copies of displayed results.

HOPTIMUM will also stop in response to a keystroke during its testing or reformatting, and then ask whether to continue or quit. This bailout will not harm any of the disk files. If HOPTIMUM stops during a reformat, it leaves the disk with one section under the new interleave and one section under the old.

HOPTIMUM first displays the drive characteristics (number of heads, cylinders, sectors per track, controller type) and asks whether they are correct. If "N" is pressed in response, HOPTIMUM presents each characteristic in turn and prompts for a new value. In addition, HOPTIMUM prompts for Write Pre-Compensation and Reduced Write Cylinder numbers. A few drive types use these values to adjust the write current strength, and those drives' manuals will specify them. For the majority of drives, the default value of -1 (which means "ignore") is correct. HOPTIMUM then redisplay the drive characteristics; it will repeat this edit and display cycle as necessary.

The program next lists its testing parameters, using defaults as necessary, and offering to edit them. If directed to change these options, HOPTIMUM begins presenting each one in turn, allowing them to be accepted or changed. After presenting all options for review, HOPTIMUM redisplay the test parameters as revised. This edit-display cycle can be repeated indefinitely.

In the BEGIN and END options the specification of surfaces (heads) is independent of the specification of cylinders. Specifying the same surface in both BEGIN and END limits HOPTIMUM to one surface, and specifying the same cylinder and surface in both options (as the defaults do) limits HOPTIMUM to a single track.

To insure that its results approximate those that application programs will encounter, HOPTIMUM should use a test area of at least ten tracks and all heads, to introduce the normal delays of head switching and (slower) track switching. Note that the time required to run HOPTIMUM also increases directly with test area and number of samples.

Before beginning its interleave evaluations HOPTIMUM asks whether to examine the test area to determine the current interleave. This optional routine issues random requests to the disk and displays the transfer rate and an estimate of the current interleave.

Following a check on whether to proceed, HOPTIMUM begins its evaluations. For each interleave tested it reformats the test area to the new interleave, removing and replacing information to preserve it, and then times itself while repeatedly reading the area.

The program reports its results (and any errors received) as it works, and then summarizes them in a table. If the program encounters more than a few errors while working it should be stopped and the disk should be checked with HTEST.

If HOPTIMUM determined the current interleave, and it finds a different test interleave that is better, it suggests a reformat. If directed to proceed, it reformats the entire disk, working from the last track toward the first, removing and replacing information as it works, and then returns control to DOS.

Tip: you can use HOPTIMUM to reformat a drive to any interleave by setting the beginning and ending test interleaves to the same value. You can also use this approach to periodically "refresh" a drive, reinitializing the format fields and data areas.

Examples

HOPTIMUM

Tests the hard disk using all program defaults. Evaluates all possible interleaves by issuing 250 full-track requests to read the next to last track on surface zero.

HOPTIMUM 1: / BI = 5, EI = 7

Tests the second hard disk, evaluating interleaves five through seven. On an XT machine the optimum interleave is usually six, and this example limits the test (and run time) to the most likely candidates.

HOPTIMUM /B = 100,0, E = 120,3

Tests the first hard disk from cylinders 100 through 120 and surfaces zero through three, which incorporates head movements into the evaluations. The larger test area will greatly increase run time.

Conflicts

Important: Do not use HOPTIMUM without a current and correct backup of all important hard disk files. HOPTIMUM's careful approach to moving large numbers of sectors is no guarantee against mishaps such as voltage sags or disk errors.

Before starting HOPTIMUM, remove any BUFFERS, FILES, or FCBS statements from the CONFIG.SYS file and reboot. Also

remove all resident programs from memory. If HOPTIMUM reports the same transfer rate for each possible interleave, quit the program and check for disk caching programs. If no caching (or other resident) programs are running and the problem persists, add the DT switch to the command line to force HOPTIMUM to use DOS's timing routines in its evaluations.

On two-speed computers the efficiency of an interleave will depend on the processor speed. For accurate results, run HOPTIMUM with the computer set at the most often used speed.

HOPTIMUM works on IBM and IBM-compatible PC, XT, and AT computers with standard MFM or RLL controllers and disks with up to 16 heads and 1024 cylinders. Detailed information is contained in the supported hardware list on page viii.

The MAX_CYL, MAX_HEAD, and MAX_SEC options may allow HOPTIMUM to format a drive that is not recognized by DOS's input-output routines. The values supplied must reflect the actual number of cylinders, heads, and sectors on the drive, and the controller must allow software redefinition of its disk parameters. These overrides will last during HOPTIMUM only; HOPTIMUM does not make an otherwise unsupported disk usable.

Important: The MACHINE option is necessary with certain computers, such as the Toshiba 3100, which use XT controllers in AT style machines. Specify the *controller* type. This value can be changed while editing the format parameters.

HPERF

Purpose

HPERF runs statistical tests that measure hard disk performance, using test ranges and sample sizes defined by the user.

Syntax

HPERF [drive]

Options

drive Indicates which drive to test; must be a physical drive number (0-9) followed by a colon. If only one hard disk is connected, HFORMAT ignores this option. The default value is zero (the first physical drive).

Options chosen while using HPERF

BC Beginning Cylinder: specifies the first of the range of cylinders in the tests. The default value is cylinder zero.

GO Begins performance measurement using displayed values.

EC Ending Cylinder: specifies the last of the range of cylinders in the tests. The default value is the next to last cylinder. Note that some drive manufacturers leave the last track unformatted for diagnostic use, so HPERF may not work on it.

EX Exit: quits HPERF and returns control to DOS.

RN	Random Seed: supplies the value HPERF uses to generate random track selection during the testing. The default value is 17, the minimum is 1, and the maximum is 65,535.
RA	Range: specifies the number of cylinders in the tests. The default value is from the beginning cylinder to the next to last cylinder on the drive.
SA	Samples: specifies the number of random repeats (the sample size) of each test. The default value is 500, the minimum is 10, and the maximum is 16,000. Note that sample sizes under 50 may generate unreliable results.
SD	Select Drive: selects the drive to be tested, using physical drive number (zero to one less than the number of drives). The default value is zero (drive C:).
ST	Sectors per Transfer Request: specifies the number of sectors HPERF will read in each sample transfer. The default value is the number of sectors per track, the minimum is 1, and the maximum is 68.
TC	Test Cylinder: selects (in combination with Test Head) the test track, which functions as a control in seek and transfer timing. The default value is the next to last cylinder on the drive. Note that some drive manufacturers leave the last track unformatted for diagnostic use, so HPERF may not work on it.
TH	Test Head: selects (in combination with Test Cylinder) the test track. The default value is head zero.

Description

HPERF creates a screen display divided into four parts. In the upper right corner of the screen it lists the options and their mnemonic codes, and prompts for a choice. When one of the two-letter codes (other than GO or EXit) is entered, HPERF prompts for a new value for that option. If it is given an illegal value or nothing then HPERF substitutes its default value for that option.

After any option is changed HPERF redisplay the current test definition in the lower right corner of the screen. In the upper right corner it displays the characteristics (number of cylinders, heads, and sectors per track) of the test drive.

Note that values for the test Range and the Beginning and Ending Cylinders are interdependent; HPERF adjusts one or more of these values if necessary when any one of them changes.

Once given the GO option HPERF begins testing, using the current test definition. The message "METRICIZING..." appears in the option prompt area, and the program cannot be interrupted (except by rebooting) until it finishes the tests. HPERF is a read-only tester and does not harm any information on the disk.

HPERF displays the results of its work in the lower left corner of the screen, and the Shift-PrtSc key combination can be used to print a copy of the results of each test. Any number of tests can be defined and run before selecting EXit to quit HPERF and return to DOS.

Under the heading "Fixed Performance Metrics" HPERF displays the time in milliseconds required to move the heads from cylinder zero to one (a track to track seek) and the average time needed to move the heads from one to another random track within the test

range (average seek). HPERF also measures and displays the disk's rotational speed.

Under the heading "Variable Performance Metrics" HPERF displays the rate in Kilobytes at which information can be read into memory from the test track and from random tracks within the test area, and the interleave of the test track. The tests that determine these values (which are "variable" in that the interleave value will affect them) do not write to the disk.

Conflicts

Before starting HPERF, remove any BUFFERS, FILES, or FCBS statements from the CONFIG.SYS file and reboot. Also remove all resident programs from memory.

HPERF works on IBM and IBM-compatible PC, XT, and AT computers with standard MFM or RLL controllers and disks with up to 16 heads and 1024 cylinders. Detailed information is contained in the supported hardware list on page viii.

HTEST

Purpose

Tests the ability of a hard disk drive and controller to read, or to write and read, all or a selected part of the disk; displays a list of track numbers and errors.

Syntax

HTEST [drive] /[B = m,n], [E = m,n], [L = n], [MC = n], [MH = n],
[MS = n], [NBD], [{Q|A|S}], [{R|W}]

Options

- drive** Indicates which drive to test; must be either C: (physical drive zero) or D: (physical drive one). If only one hard disk is connected, HTEST ignores this option. The default value is C:
- B = m,n** BEGIN: defines the first cylinder (m) and first surface (n) of the test area. The default values are cylinder zero and surface zero.
- E = m,n** END: defines the last cylinder (m) and last surface (n) of the test area. The default values are the numbers of the disk's last cylinder and last surface.
- L = n** LOOP: specifies the number of times to repeat the test. The default value is one, and the maximum is 9999.
- MC = n** MAX_CYL: overrides the controller's value for the number of cylinders on the drive. The value must be an integer from 1 to 1024.

MH = n	MAX_HEAD: overrides the controller's value for the number of heads on the drive. The value must be an integer from 1 to 16.
MS = n	MAX_SEC: overrides the controller's value for the number of sectors per track on the drive. The value must be an integer from 1 to 63.
NBD	("No Big Deal"): manipulates the controller through DOS's routines instead of directly. Used if HTEST has trouble locating the controller program (BIOS). This option has no long form.
Q	QUICK: tests the sectors in ascending cylinder and head order. This is the quickest of three possible test patterns. If no pattern is specified then HTEST uses QUICK by default.
A	ACCORD: tests the sectors beginning at the mid-point of the testing range and alternately testing the next lower and then the next higher untested cylinder, in an accordion-like pattern.
S	STAGGER: tests the sectors in ascending cylinder and head order, but move the heads to both the first and last cylinders of the drive after testing each sector. This is the longest of the three possible test patterns.
R	READ: limits testing to non-destructive read requests. If neither READ nor WRITE is specified then HTEST READs by default.
W	WRITE: tests by writing and then reading each sector. This testing destroys all information in the areas tested.

Description

HTEST runs on physically formatted disks with valid track and sector marks, and can be run as a read-only test to protect existing files. If it is run as a write test, it destroys existing information in the area tested.

HTEST works directly with the controller, bypassing the normal DOS input-output routines. It ignores partitions and FATS and works at the physical level with all or any part of a disk.

HTEST begins by locating the hard disk controller program (BIOS) normally hidden from direct access by DOS. It displays the message "signature value x,y" to identify the beginning of the controller's input-output routines.

The program next describes the test to be performed, using defaults as necessary, and offering to edit them. If directed to change these options HTEST begins presenting each one in turn, allowing them to be accepted or changed. After presenting all options for review, HTEST redisplayes the test parameters as revised. This edit-display cycle can be repeated indefinitely.

In the BEGIN and END options, the specification of surfaces (heads) is independent of the specification of cylinders. Specifying the same surface in both BEGIN and END limits the test to one surface, while specifying the same cylinder and surface in both options limits HTEST to a single track.

If "N" is pressed in response to "change testing parameters" HTEST asks whether to begin; it will quit and return control to DOS if instructed not to proceed. Please note that HTEST does not issue a warning in the case of a WRITE (destructive) test.

HTEST begins testing by resetting the controller and directing it to recalibrate the drive (find cylinder zero). This assures that the controller and drive will respond predictably to the test. If the program encounters any problems with these tests it will display them and ask whether to continue with formatting, as these problems usually indicate a need for disk or controller service.

HTEST displays cylinder and head numbers as it works, and displays read/write errors as it encounters them, creating pages (screens) of error reports. Each error report takes the form

Cylinder	Head	R/C
-----	-----	-----
245	1	80h

where the cylinder and head are in decimal and the return code (R/C) is in hexadecimal. The program resets the controller after each error.

HTEST suspends testing if the space bar is pressed (it may take a moment to finish a pass before responding). This interrupt does not harm disk files. When HTEST is interrupted (and after it has finished testing) it displays a menu of options which can be selected by single keystrokes. These choices allow the user to page through all error reports, call up an explanatory list of return codes, resume testing, or exit the program and return to DOS.

Examples

HTEST

Tests the hard disk using the program defaults. Testing will be read only, quick (sequential) head movement, over all disk surfaces, with no repetitions. This is the fastest possible test of the entire disk.

HTEST D: /W,S,L=10

Tests the second hard disk using write (destructive) testing and the **stagger** pattern of head movement. Repeats the test ten times, **testing** the entire surface by default. This combination of write and **stagger** testing is the most stringent (and most time-consuming) test **possible**.

HTEST /BEGIN=0,0 , END=0,3

If the hard disk has four heads this command tests all of the first **cylinder**; testing begins and ends on cylinder zero, beginning with **head** zero and ending with head three. The test will be read-only and quick by default.

HTEST /B=0,2 , E=614,2

If the hard disk has 615 cylinders this command tests all of surface **two**, and will not test any other surfaces. The test will be read-only and quick by default.

Conflicts

Before starting HTEST, remove any BUFFERS, FILES, or FCBS statements from the CONFIG.SYS file and reboot. Also remove all resident programs from memory.

HTEST works on IBM and IBM-compatible PC, XT, and AT computers with standard MFM or RLL controllers and disks with up to 16 heads and 1024 cylinders. Detailed information is contained in the supported hardware list on page viii.

The MAX_CYL, MAX_HEAD, and MAX_SEC options may allow HTEST to test a drive that is not recognized by DOS's input-

output routines. The values supplied must reflect the actual number of cylinders, heads, and sectors on the drive, and the controller must allow software redefinition of its disk parameters. These overrides will last during HTEST only; HTEST does not make an otherwise unsupported disk usable.

If HTEST generates a lot of "address mark not found" errors then you may be testing an unformatted disk or a disk and controller combination that was not used to format the disk. Use HFORMAT to perform the (low-level) formatting.

If HTEST locks up the computer when it starts, reboot and add the NBD option to the command line (it cannot be invoked within the program). This allows HTEST to use DOS's interfaces instead of trying to locate and talk directly to the controller. It will slightly lessen HTEST's error-checking abilities.

RELOCATE

Purpose

RELOCATE moves the hard disk heads to a specified cylinder on the drive to protect the surfaces during power-off and the subsequent restart.

Syntax

RELOCATE [drive] [/LZ = n]

Options

- drive** Indicates which drive to park; must be either C: (physical drive zero) or D: (physical drive one). If only one hard disk is connected, **RELOCATE** ignores this option. The default value is C:
- LZ = n** Specifies which cylinder to use. The default value for LZ ("landing zone") is the drive's last cylinder.

Description

RELOCATE moves the hard disk heads on the specified drive to the cylinder specified, displays the cylinder number, and returns control to DOS. After **RELOCATE** is run the computer can still be used, although subsequent disk commands will move (unpark) the heads. This program should be run just prior to turning off the computer.

The disk heads, which normally hydroplane above the platters on a stream of air, make contact with them when the drive is off. When the drive is turned on, the heads grind along the surfaces while waiting for the drive to get up to speed and float the heads. Consis-

tently parking the heads on the last cylinder minimizes this inevitable damage and protects the vital first few cylinders of the drive.

Conflicts

Do not try to park the heads with RELOCATE in a batch file. The system will move the heads after executing RELOCATE as it attempts to find the next batch command (or the end-of-file marker).

33TRAP

Purpose

A resident program that intercepts DOS multiple track read/write requests and separates them into single track requests.

Syntax (no options)

33TRAP

Description

IBM Personal Computer DOS versions 3.30 and above assume that the hard disk controller can handle multiple track requests. Some IBM "compatibles" and some controllers do not support this feature and will freeze the system when DOS issues such a request. The most telling symptom is application programs that begin locking up during disk read or write operations after a DOS upgrade from 3.2 or earlier versions.

33TRAP "cascades" multiple track requests and feeds them as single track requests to the hard disk controller. It runs as a terminate and stay resident (TSR) program which uses about 1K of memory; it does not degrade system performance.

33TRAP.COM is not supplied on the HTEST/HFORMAT diskette: it is created by running C33TRAP. C33TRAP creates 33TRAP.COM, tailoring it to the machine and controller combination on which it will run. Note that this 33TRAP.COM cannot be run on machine/controller combinations other than the one it was created for.

C33TRAP has no command options, and can be rerun as necessary on other machine/controller combinations to create additional 33TRAP.COM programs.

XFDISK

Purpose

Creates and changes partition tables. Partitions disks by writing the master boot record to the disk. Saves backup copies of partition tables in disk files.

Syntax (no options)

XFDISK

Description

XFDISK is an expanded and non-destructive version of DOS FDISK. In addition to DOS partitions, XFDISK can define other operating system partitions, including KRI DFT, KRI XDISK, KRI XBOOT, Compaq, CP/M, PCIX, XENIX, UNIX, Speedstor, and Vfeature.

Like DOS FDISK, the program can be used to partition a hard disk by writing the master boot record (which contains the partition table) onto the disk. But XFDISK will also save a copy of the master boot record in a disk file, creating a backup which gives the user unprecedented recovery power. Furthermore, XFDISK does not perform the destructive cylinder testing that makes DOS FDISK useless for hard disk recovery.

Having a backup copy of the master boot record in a disk file, and being able to re-write it to the hard disk with XFDISK, provides an unprecedented level of recovery power. At least one copy of the file should be kept on a diskette, ideally with XFDISK, for emergency use.

The file is called HD0BOOT.REC in the examples in this manual, using the default filename that XFDISK suggests for the first fixed disk. Techniques that use the backup file to recover from an "Invalid Drive Specification" error are outlined in chapter two.

XFDISK displays a sequence of menus and prompts for all necessary choices and decisions. When it starts it displays its Main Menu, which features these functions:

- add partitions
- designate the active partition or change partition types
- remove partitions
- list the partitions
- display or change the disk's characteristics
- restore the master boot record from a file
- save a copy of the master boot record as a disk file
- write the new table to the hard disk's master boot record
- select a drive, if more than one is connected

The program reads a disk's current partition table from memory when it starts, and allows unlimited changes to it. It does not automatically save these changes, however. Unless it is directed to write the new table into the master boot record, or to save it as a disk file, all changes are lost when XFDISK returns control to DOS.

When directed to save the master boot record as a file (or to read a previously saved copy), XFDISK asks for a filename and offers HD0BOOT.REC as a default. In the case of a second fixed disk, it offers HD1BOOT.REC, using DOS's physical drive numbering. The filename can be changed, and a valid drive and path can be added, up to a maximum of 26 characters. XFDISK uses the current drive and path by default. Note that use of the default filename means that the file is easily identified without notes.

The main menu choice "Modify a Partition" includes both designating the active (booting) partition and altering partition types. The program's ability to change partition types is based on the partition table's use of one byte to indicate each partition's type. XFDISK automatically sets this indicator byte for a wide range of partition types, but other types can also be defined, by using XFDISK to directly define the indicator byte.

This same feature can function as a simple safety device. If a DOS partition's indicator byte is changed, and the machine rebooted, DOS will no longer recognize the partition. From DOS's point of view, in fact, it will not exist. This ability to hide an entire partition can be useful for security, although it won't stop anyone who knows about indicator bytes.

XFDISK will also redefine a drive's characteristics (number of heads, cylinders, and sectors per track). This feature allows it to create partition tables for drives that DOS does not support, just as HFORMAT will format them. The values supplied must reflect the actual number of cylinders, heads, and sectors on the drive, and the controller must allow software redefinition of its disk parameters. However, such drives require special software to work with DOS.

Conflicts

Before starting XFDISK, remove any BUFFERS, FILES, or FCBS statements from the CONFIG.SYS file and reboot. Also remove all resident programs from memory.

XFDISK works on IBM and IBM-compatible PC, XT, and AT computers with standard MFM or RLL controllers and disks with up to 16 heads and 1024 cylinders. Detailed information is contained in the supported hardware list on page viii.

XFDISK will only alter a disk's partition table if specifically directed to do so, and only after confirmation. The changes to the partition table will not take effect until the machine is rebooted, when the master boot record is read into memory. At that point, any changes to a partition table will make a disk's previous files unreadable. Replacing the partition table with a copy of the original, or reconstructing it, will restore the files, if no other disk operations have been performed, and if XFDISK was used in both steps.

Chapter 4: An Introduction to Hard Disks

This chapter provides an introduction to hard (fixed) disk drives for those who are unfamiliar with the breed. It discusses disk design and operation, the formatting process, and common disk problems. Some of the more powerful HTEST/HFORMAT applications (especially the recovery procedures) cannot be fully understood without this background. If you are reluctant to subject your disk files to any process you don't understand, this information is a must.

Description

This section describes (very briefly) the physical components and the logical organization of hard disks. It lays the groundwork necessary to an understanding of the sections that follow, on hard disk formatting and problems. You should also read the information on disk drives that came with your computer (or with your drive).

Platters and Heads

Hard disks for microcomputers use the Winchester drive design, in which several rigid platters rotate rapidly like records on a common spindle, but with spaces between them. A comb-like armature holds a magnetic recording head over (and under) each platter, and the platter rotation generates an airstream which allows the heads to hover just above the surface. A motor controlling the armature can move the heads radially, in or out of the stack, to any of a number of predetermined positions, "stepping" them in unison from near the outer edges of the platters to near their centers.

Both sides of a platter (called surfaces) carry a thin coating which can accept and preserve magnetic charges. In essence, your files are simply sequences of magnetic charges on the disk's surfaces.

Tracks and Cylinders

The narrow band of recording surface that passes under a head as it hovers above a rotating platter is called a track, and the sequence of positions assumed by a head as the stepper motor moves it back and forth over the platter surface defines a set of concentric tracks on the platter. The head can write information onto a track by creating a series of magnetic pulses in the coating as the platter spins beneath it, and it can read that information later, sensing the same pulses as they pass under it.

When the system controlling the drive writes or reads information it must specify which surface, and which track on that surface, to use. The convention for specifying this information, strangely enough, does not refer to either surfaces or tracks. Instead, since there is a head for every surface, we will specify which head to use. And instead of referring to a track, we will specify a cylinder. Remember that the armature moves all the heads together, so when a head is at a particular track, the set of tracks under all the heads forms a cylinder.

The heads and cylinders are numbered beginning with zero; cylinder zero is the one nearest the outer edges of the platters. When information is written to an unused area of the disk it is done as efficiently as possible, filling the tracks in one cylinder sequentially before moving the heads to the next cylinder. So a large file on a four-surface drive might cover, in order, track 45 on sur-

faces 0, 1, 2, and 3, then track 46 on surfaces 0, 1, 2, and 3, then track 47 on surface 0, etc.

Sectors

In addition to the physical division of the disk's platters into tracks, formatting subdivides tracks into sectors, which are short lengths of track that can be read or written independently of one another.

The formatting routine writes information in each track at regular intervals, marking the beginning of each sector and the sector's number. Sectors are numbered beginning with one, not zero (as with heads and cylinders). The operating system makes read and write requests in groups of sectors, called clusters, and the number of sectors per cluster varies from four to sixteen, depending on the disk and the version of DOS in use.

Formatting is accomplished by the controller, which uses one of several different methods for coding information when writing it. MFM (modified frequency modulation) coding, which divides each track into 17 sectors, is the most common method. The RLL (run length limited) method, recently adapted from mainframe technology, creates either 23 or 27 sectors per track, depending on the specific controller. Other methods include ST-506, ESDI, and SCSI.

Interleave

The final wrinkle in this scheme is sector interleave. To understand the reason for interleave, we'll walk through a read request for sectors one and two of a track, using the head's point of view, assuming that sector two follows sector one on the track.

DOS begins by asking the controller to read sector one and copy it into memory. To find the first sector, the head reads the sector numbers as they spin past until it finds the beginning of sector one. It reads sector one and the controller copies the information into memory. By the time that DOS has indicated where it wants the next sector placed, sector two is already passing under the head, and everyone has to wait for one revolution of the platter, when the beginning of sector two will again pass under the head.

But the information marking the beginning of a sector includes that sector's number, so we don't need to place them on the track in sequence. Instead, we could start by marking sector one, then skipping a sector's worth of track and marking sector two. Continuing in this fashion (and assuming 17 sectors per track), we will lay sector nine just before we again encounter sector one, which we skip, laying sector ten between sectors one and two, and eventually filling the track. We have now created an interleave of two: the track distance from sector one's beginning to sector two's beginning is equal to two sector lengths.

Now, in our two-sector read request, the system has a little time between sectors while sector ten passes the head. If this is enough time to prepare for the second read, it saves waiting for a disk rotation, and our reads (and writes) will be much faster. Whether the interleave really does achieve this depends on your configuration (processor speed, operating system, hard disk and controller).

If each track is divided into a prime number of sectors (17, 23, etc.), we can create any interleave from one (laying the sectors down sequentially) to 16, and the distance between consecutive sectors will be constant around the track. With a non-prime number of sectors, some interleaves would require a one-sector deviation from the pattern.

Read and write requests will vary, so no interleave can assure that the next sector always arrives just after your system is ready for it. But for a given system, the difference between a good interleave and a poor one can be a factor of ten; that is, the best value may move data ten times as fast as the worst.

HTEST/HFORMAT offers you two ways to achieve this best value: HOPTIMUM and HFORMAT. HOPTIMUM will test different interleaves to determine the best one, and it can change the interleave without harming your files. HFORMAT will, when formatting a disk, allow you to specify the interleave to use.

Formatting

This section outlines the steps that prepare a hard disk for use: physical formatting, partitioning, and logical formatting. These concepts are crucial to an understanding of HTEST/HFORMAT.

Bad Tracks

The surfaces of your hard disk are the product of a manufacturing process and are, of course, not perfect. The tolerances for the surface coating allow enough variation that some spots on the disk will not be usable, so in compensation the drive is built with slightly more than its nominal capacity. The manufacturer tests writing to and reading from the disk in order to locate tracks that are out of tolerance, and to guarantee that the first cylinder (cylinder zero) is good, for reasons that will become obvious. Notice that the testing identifies suspect tracks, not sectors, since a track is the smallest accessible portion of the disk at this stage. HFORMAT allows you to do your own (and more exhaustive) testing, and lock out bad tracks.

Physical Format

The first step in formatting a hard disk is to number the tracks and sectors. The program performing the physical formatting (also called low-level, controller, or absolute formatting) writes information onto the platter surface, identifying each track and sector. When the disk is in use, this information will tell the controller where to begin reading or writing a sector.

The formatting program will ask for a list of bad (or at least suspect) tracks, and will mark them unusable; no information will ever be stored in them. At this stage in the hard disk's life it's more efficient to freeze out entire tracks, even if only a small part of the track has a problem. The program will also do its own testing and may lock out additional tracks.

Since the program writes sector numbers onto the disk, the interleave is established during physical formatting. If you purchased a hard disk that was formatted by the manufacturer or the retailer, the interleave may or may not be set at the optimum for your system. If you are already using your disk, the good news is that HOP-TIMUM allows you to determine whether a better interleave is possible, and then change the interleave without harming any information on the disk.

A class of controllers called "autoconfigure" controllers (which includes most RLLs) adds another wrinkle to this scheme. Before it can work with the disk, an autoconfigure controller must read information about the disk which was written *between* sectors on the first track of the drive (cylinder zero, head zero). This information allows it to set (configure) itself so as to be able to read the disk. The controller contains its own formatting routine, which writes the special information when it formats the disk.

Partitions

When IBM developed fixed disks for the PC their design specified that multiple operating systems should be able to share the disk without requiring backup and restore. In other words, the operating systems must leave one another's files alone. Partitioning the disk achieves this objective, dividing the disk into pieces and giving each operating system the exclusive use of one piece.

When DOS's Fdisk (or a similar program) creates the partitions it assigns an uninterrupted span of cylinders to each of them, and allows you to designate one of them the active (or bootable) partition. It creates a partition table, listing the first and last cylinder in each partition and identifying the active partition. The program adds the table to a small bootstrap program, creating the "master fixed disk boot record," which it writes onto the disk in the first sector (head 0, cylinder 0, sector 1; remember that sectors numbers begin with 1).

To understand this boot record, let's walk through the sequence that your computer uses when first turned on. After it finishes its self-testing routines, if drive A: is empty it gets the master boot record from the first sector of the fixed disk (if you have two fixed disks, it reads the first one). The program contained in this boot record looks up the beginning of the active partition (in its partition table) and loads another boot record from that partition.

DOS always refers to the active partition on the first fixed disk as drive C:, and its boot record contains a program which actually loads and starts the operating system, using COMMAND.COM and two hidden system files which have to be the first two files in the root directory (they are placed there during logical formatting).

Note that even if there is only one partition on the disk, occupying all of it, the disk still uses a master boot record and partition table. The ability to read the master boot record is crucial, which explains why manufacturers certify that cylinder zero is free of defects. Unfortunately, problems can appear, but chapter two shows how the HTEST/HFORMAT programs can cure most of them.

Logical Format

Logical (also called high-level or relative) formatting enables a partition to store and retrieve files, and most references to “formatting” refer to this final step in disk preparation. For a DOS partition, the formatting program will create the following structures:

- a boot record
- a file allocation table, or FAT
- a backup copy of the FAT
- a space reserved for root directory entries

and will write them on the disk, in that order, beginning at the first sector of the partition. These structures fill several tracks; the exact number depends on the DOS version in question.

The boot record contains a program that can load and start the operating system. It also contains information DOS needs, including the DOS version used to format the partition, the number and sizes of disk surfaces, tracks, and sectors, the number and size of the FATs, and the number of entries permitted in the root directory.

DOS's file allocation table (FAT) is long enough to contain an entry for each cluster in the partition. The value of an entry indicates whether that cluster is free (unused), bad (unusable), or in

If the cluster is marked "in use" then the entry either points to the next in a chain of clusters or identifies itself as the last in the chain.

The root directory can hold 512 entries (1024 for a 20 megabyte disk formatted with DOS version 2). Each directory entry, in addition to the information displayed by DOS's Directory command, contains the starting cluster for the file. DOS uses this cluster number to find the beginning of this file's chain in the FAT (it may be as short as one cluster), and locates the file by following the FAT chain.

Obviously DOS cannot find a file without the FAT (and neither can you, short of examining every cluster in the partition and knowing what you're looking for). Since the information in the FAT allows DOS to read the disk, DOS maintains a backup copy. Unfortunately, it doesn't automatically take advantage of the copy when the first FAT is unreadable.

The Format program supplied with DOS is a logical formatter which also works with floppies. A similar program from Mace Utilities, FORMAT-H, is designed for hard disks alone. When formatting an active (bootable) partition, the program must also copy the operating system onto the disk. It places two system files (which DOS normally hides from you) in a reserved area near the beginning of the partition, and copies the file COMMAND.COM into the root directory.

Problems

Unless you are the exception, you don't back up often enough. How often is enough, you ask? Well, how long will it take you to

recreate your files? Put down this manual, now, and back up your hard disk. If you consistently postpone backing up because it's too time-consuming, buy a faster backup system.

Now try to restore at least some of the files, just to check that your backup worked. Somebody at our office didn't, and found out too late that his backup was worthless. Why the emphasis on backup? Because hard disks pack a lot of technology into a small space, and Murphy was an optimist.

Problem Solving

If the problem occurs at power-up, make sure the machine and the room are at their normal temperature (a difference of twenty degrees will make any disk cranky). New and interesting whines, grinds, and other impolite mechanical sounds are really cries for help, and they indicate that it's time to back up everything and have the drive serviced. It will not get better on its own.

You should make a habit of testing the disk (HTEST or Mace Utilities DIAGNOSE or REMEDY) regularly. And any analysis of sudden disk or file problems must now consider the possibility of a computer virus (a.k.a. Trojan Horse, worm, bomb). Some apparently useful programs incorporate unpleasant surprises, such as an unannounced reformat of your hard disk. Contact Paul Mace Software for information on the preventive program MACE VACCINE.

Whatever happens, please don't panic. If your hard disk won't cooperate, have a cup of tea (or something stronger) and read on, and remember that help is available.

Disk Error Messages

The following list is not comprehensive, but should help you decide where to turn next. We'll assume that your hard disk has an active (booting) DOS partition and the drive actually runs (the light comes on when you turn on the computer).

1. If you turn on the power and, during the computer's self testing routines, you see messages like these (where "XX" is two numerals):

- Error 17XX
- 17XX

then the system has found a problem with the hard disk or controller. See "Power-On Testing Errors" in chapter two.

2. If you turn on the power and, after the computer's self testing routines, you see either of these:

- Invalid drive specification
- Invalid partition table

then you may have a corrupt or unreadable master boot record. The computer cannot transfer control to your DOS partition (drive C:) and therefore cannot load DOS. See "Invalid Drive or Partition Table" in chapter two.

3. If you turn on the power and, after the computer's self testing routines, you see one of these:

- Non-system disk or disk error
- Missing operating system
- Error loading operating system
- Bad or missing command interpreter

then either the partition table has marked the C: partition inactive or else the partition itself has problems. The hidden system files, or the command interpreter `COMMAND.COM`, may be unreadable or garbled (or missing). See "Operating System Startup Failure" in chapter two.

4. If DOS starts from the hard disk, but subsequent disk reads or writes produce either of these:

- File allocation table bad
- General failure reading drive c:

then retry (press R at DOS's prompt) several times, and press A to abort if the problem persists. DOS can't read its own structures, but the files may still be okay. See "Bad Directory or FAT" in chapter two.

5. If DOS starts from the hard disk, but subsequent disk reads or writes produce messages like these:

- Sector not found error reading drive c:
- Write fault error writing drive c:

then areas of the disk's surface may be unreadable. Another symptom of this is a repeated "whirr-click" while DOS tries to read a bad sector. When these errors occur, DOS offers:

- Abort, Retry, Ignore?

Press R to retry. If one or two retries fix the problem then finish what you were doing, but if the problem persists press "A" to abort. In either case, consult the Mace Utilities for instructions on using `REMEDY` to check the disk, move files, and lock out bad sectors.

Appendix A: PCs and Primary Storage

The members of the IBM Personal Computer family, like any other digital microcomputers, are “stored program” devices; the architecture is a variation on the von Neuman computer. This von Neuman architecture includes:

- a linear organization of fixed size memory cells
- a one level address space
- a low level machine language (instructions that perform simple operations on elementary operands)
- sequential and centralized control of computation
- processor, communication, and memory incorporated into a single unit

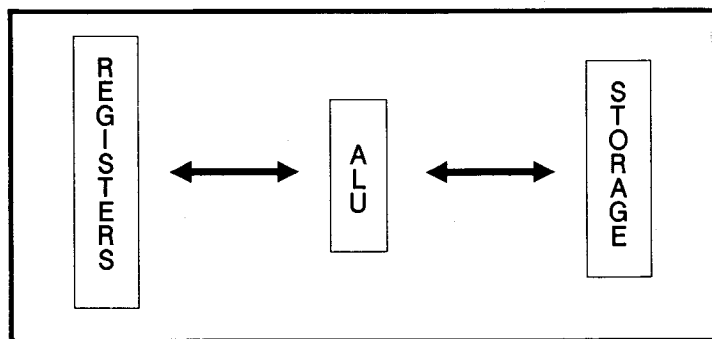


Figure A.1

With this design in mind, the IBM PC can be functionally described by the block diagram in Figure A.1. A standard “stored program” computer is composed of the following functional elements:

- an arithmetic and logic unit (ALU) which performs functions on data elements and generates results
- a set of high speed internal registers which are the vehicle used to bring data into the domain (in the mathematical sense) of the ALU
- a primary memory unit, capable of operating at electronic speeds, whose sole function is to provide high speed access to programs and data during execution (the memory subsystem of a digital computer is functionally a set of such registers where programs and data are stored)
- secondary storage, where large amounts of data that are not frequently needed, or that overflow the available primary memory, may be kept

The instructions from the programs stored in memory are retrieved by the control unit of the machine (digital computer) and are decoded to perform the appropriate operations on the data stored either in memory or in a set of registers in the processing unit.

The access methods are represented in Figure A.1 with arrows. Access to the machine's functions occurs only through use of the registers. Data is brought into registers through random access memory (RAM) or through the I/O ports in a machine. The source for data in RAM is either the registers or an external or peripheral device capable of controlling the electronic signals necessary to support the addressing and writing of data into RAM (a CPU normally performs these functions); or data may be accessed from RAM via the registers, with the CPU generating the necessary control signals, or via a peripheral device which can generate and orchestrate the signals necessary to read data from RAM.

The designers of the IBM PC chose the INTEL 8088 (or the 80286 in the case of the PC/AT) as the primary controller of activities within the PC. It is within this unit that the ALU and high speed registers reside. Intrinsic to the 8088 is a set of fourteen 16-bit registers. These are the fastest and sparsest high speed memory in the PC. Five of these registers (ES, CS, SS, DS, and IP) are used primarily in conjunction with the Bus Interface Unit of the 8088. ALU functions performed on data elements are normally processed in the registers associated with the execution unit (AX, BX, CX, DX, SP, BP, SI, DI), notably the logical operations.

For simplicity's sake, and to facilitate calculations, this leaves a total of $4 \text{ (AX, BX, CX, and DX)} * 16 \text{ bits} = 48 \text{ bits}$ available for computations upon data. I have limited the number of processing registers to these four because the other registers are normally associated with the manipulation of data in preparation for receipt by one of these four. Depending on the definition chosen for processing, such as whether or not all overhead incurred in doing a calculation is considered processing, this would of course alter the number of bits which are involved in the actual "processing."

Looking back at Figure A.1, the registers are a filter, in the sense that all data must pass through them in order for computations on that data to occur. They are also a potential bottleneck for processing.

One of the most important parameters of any device used to hold information (RAM, ROM, cassette, disk, etc.) is its "access time." The access time of a storage device is defined as the amount of time that elapses between the moment a request for data stored in the device is issued and the moment at which that data is available to the requester. Access times range from a few seconds down to a few billionths of a second. Another very important parameter of

storage devices is their cost. This ranges from a few dollars per bit of storage down to less than one-thousandth of a cent per bit.

Readers will not be surprised that the faster devices are the more expensive ones. Compare 20 nanosecond ECL static RAMs to floppy diskettes or even 450 nanosecond dynamic RAMs. A large part of the practice (art) of computer engineering, as in all engineering fields, involves cost-performance tradeoffs, and storage speed versus price is an important consideration in any design.

The primary, directly addressable memory space available in the PC is 1 megabyte (8 megabits). This amount of storage is the limitation imposed by the 20 address pins of the 8088 (this of course does not include "bank switching" implementations such as the Intel/Lotus/Microsoft scheme, nor does this include the 80286 protected mode of processing).

This primary memory can be used to store both programs and data. While the timing of functions performed on data in registers is strictly a function of the gate delay and, in the case of the 8086 family, the micro-code design, operands and instructions which are stored in memory are first hindered by access time to the memory (access time is defined as the time between the request of information from the storage device and the presence of that information in a stable form on the lines) and then by the delays internal to the CPU.

As an aside, some very interesting schemes have been drawn up to increase the overall performance of primary memory. Some implementations include "cache" memory, with access times in between register access and RAM, and memory interleaving, in which requests can be issued to memory in such a way that when data word one is available from bank one of memory, on the next clock

cycle data word two is available from bank two without waiting for the normal memory access delay associated with bank one.

To return to the discussion though, this “cost” in access time is compensated for by the fact that RAM is cheaper and can be added far more readily than can another microprocessor or additional registers within the microprocessor. But there is still a limitation on the amount of RAM that can be directly addressed by the CPU. Also, RAM is volatile in nature—unless battery backup is provided, the data represented by the presence or absence of charges will be lost when power is removed from the machine.

The two primary forms of data storage mentioned so far, registers and primary memory, can be modeled in another way as shown in Figure A.2. In this figure, the ALU is represented by the peak of

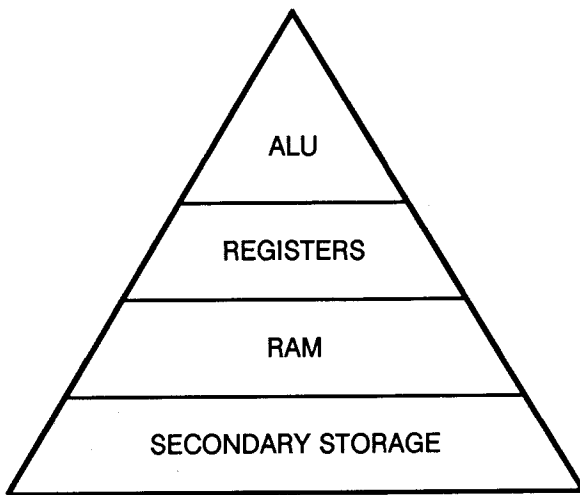


Figure A.2

the triangle. Registers are directly underneath, and they rest in turn upon a base composed of primary memory (RAM) and secondary memory.

Appendix B: PCs and Secondary Storage

Secondary memory completes our “cone” of addressable storage. This form of memory is used to store infrequently used data or instructions, or the overflow from primary memory. This is not to say that what is normally called secondary memory couldn’t be used as primary memory. Conceivably one could configure a system in which references normally associated with RAM access could be reconfigured to correspond to an address on a peripheral storage device. This augmenting of primary storage occurs in the form of “virtual storage” provided by some machines (IBM S/370, 80286) and some operating systems (MVS/VS, UNIX, and others). Economic considerations come into play in opting to go with secondary storage.

Of the technologies available to provide secondary storage, the engineers at IBM chose magnetic media as the primary vehicle. In particular they chose magnetic disk systems. In contrast to registers and RAM, which are “active” devices, magnetic disk systems fall into the category of passively stable devices; they do not dissipate energy just holding the information.

Magnetic recording is a storage mechanism that employs magnetic fields and relies on the fact that moving a magnet past a coil will induce a voltage in that coil and that the sense of that voltage will depend on whether the magnet is moving along north pole first or south pole first. Devices which employ this moving magnet principle include disks, drums, and magnetic tape.

Imagine that there is a strip of ferromagnetic (read: capable of being magnetized) material that may be moved under a fixed coil called the “head.” If a pulse of current is passed through the head, a magnetic pattern will be created on the material as it goes by.

Positive current pulses will induce one pattern, and negative pulses will induce its opposite. If sometime later the material is passed again under the same (or a similar) head, a current or pulse will be generated that will be of the same polarity as the original "writing" pulse.

As the name implies, the storage medium in a magnetic disk system consists of one or more disks (or diskettes, in the case of floppy disk drives) stacked onto each other. A thin film of ferrous material is deposited on each disk, usually on both sides. The disks are mounted on a rotary drive such that the magnetic surfaces move in close proximity to a series of read/write heads. The disks are rotated at a uniform speed and are not started or stopped during the actual access operations. Each head consists of a magnetic yoke and a magnetizing coil.

Digital information can be stored on the magnetic film by applying current pulses of suitable polarity to the magnetizing coil. This causes the magnetization of the film in the area immediately underneath the head to be switched into a direction parallel to the applied field. The same head can be used for reading the stored information. In this case, changes in the magnetic field in the vicinity of the head resulting from the movement of the film relative to the yoke cause an induced voltage in the coil, which now serves as a sense coil. The polarity of this voltage is monitored by the control circuitry to determine the state of magnetization of the film. It should be noted that only changes in the magnetic field under the head can be sensed during the read operation. Therefore, if the binary states 0 and 1 are represented by two opposite sides of magnetization, a voltage is induced only at 0 to 1 and 1 to 0 transitions in the bit stream. A long string of 0's or 1's will cause an induced voltage only at the beginning and end of the string. In order to determine the number of consecutive 0's or 1's stored, it is neces-

sary to provide information for synchronization, or a clock. This clocking information can be recorded on a separate track, where a change in magnetization on the data track, plus the clock signal, will allow correct reading of the stored data. Many commercial disk units are based on the scheme of storing the clock information on a separate track. In the case of the disk subsystems currently being delivered by IBM, the clocking and synchronization information resides within the data bits.

This page is blank.

Appendix C: Disk Encoding Schemes

Encoding schemes are designed to maximize the usable channel capacity of a device. Claude Shannon, of Bell Labs, in the 1940's rigorously defined the quantization of information, channel capacity, and transmission. The concept of data encoding originated in conjunction with the design of the telephone operating system.

If one looks at a diskette or disk drive as a communications channel, the notions created and defined by Shannon show great applicability. Associated with each channel is a maximum capacity. In a perfect channel, data can be transmitted (stored) at this maximum. However, noise exists. In the case of a diskette or disk, the ability to generate and detect signals is limited by numerous factors, including the magnetic flux "changes per inch" that the media can support, the size of the magnetic field created by the read/write head, and the error in drive rotational speed, among others. Except for mechanical aberrations, the major physical limitations in magnetic recording have to do with the packing density, that is, how many bits per inch can be recorded linearly and still be distinguishable. The fewer overall changes in state there are, the closer the bits can be packed together. Shannon stated that a maximum utilization of the channel could be achieved by the introduction of codes, although he did not show how to accomplish this.

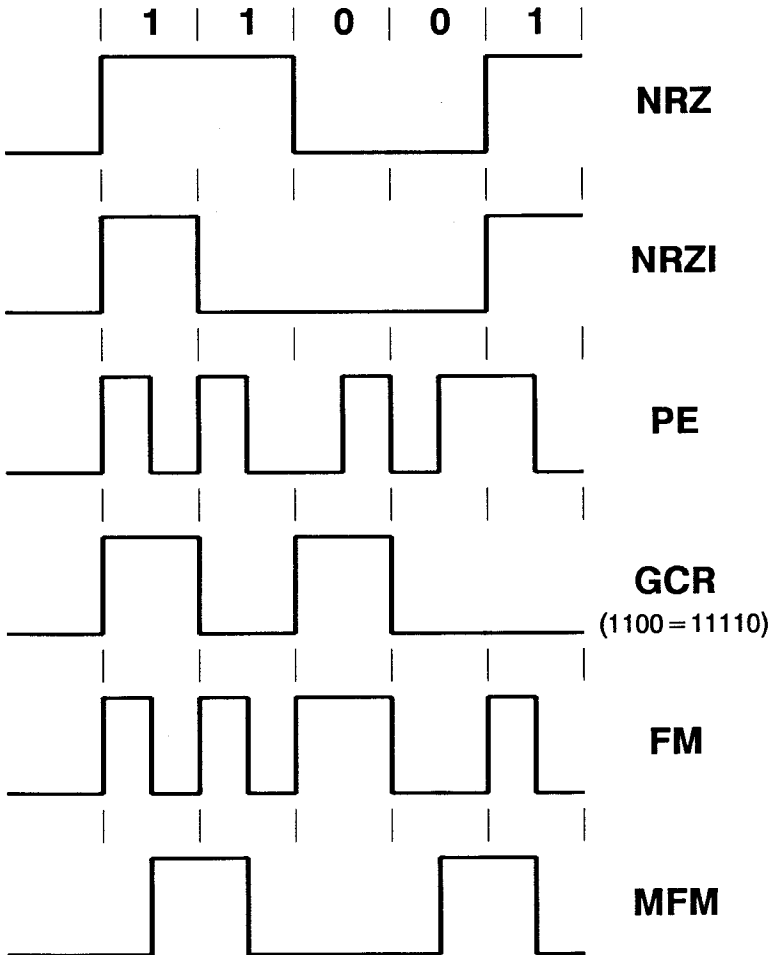
There are a variety of recording techniques in common use. Figure C.1 illustrates some of these. In the non-return-to-zero (NRZ) format, when a 1 follows a 1 or a 0 follows a 0, no change of state can be detected on output. Otherwise, there is a change. A variation of this is non-return-to-zero inverted (NRZI), where there is a change of state only when there is a 1. A third format, phase encod-

ing (PE), changes state in the middle of every bit. If it is a 1, it generates a negative pulse; if it is a 0, a positive pulse. This of course means that if two 0's or two 1's follow each other, there is a change of state between them.

NRZI, with its fewer changes in state, allows a better packing density than NRZ. However, since it is impossible to tell where some bits start and stop, a clock trace must also be recorded with the data. This is not true of PE, which is self-synchronizing and therefore allows higher densities. However, the many reversals still limit the density that PE can achieve.

A recording format called group code recording (GCR), based on NRZI, achieves much higher densities by encoding the data bits. In NRZI, a string of zeros will have no change of state, causing synchronizing problems. GCR overcomes this by encoding four data bits into five-bit patterns, none of which have more than two zeros adjacent nor more than one zero at either end. In the thirty-two combinations possible with five bits, there are seventeen which meet this requirement. Sixteen of them are used to encode data, and the seventeenth (11111) is used as a synchronization pattern.

Another type of recording format is called frequency modulation (FM), or Manchester code. For this type of encoding there is a change of state in the middle of a 1, but not in a 0. There is also a change of state at the beginning of every bit. This last change makes this format self-synchronizing; there is always a pulse, either positive or negative, when the next piece of information passes under the head. A variation of this is modified frequency modulation (MFM), which also has the change in the middle of a 1, but only has a change of state at the beginning of a bit when two 0's are adjacent. This in effect can double the density, because the bit size can be cut in half.

Figure C.1: Comparative Encoding of a Bit Pattern

It is this last format, MFM, that IBM uses to define the storing of data in both their floppy disk drives and their fixed disks.

This page is blank.

Disk(ette) Primer

One class of simple, small, and cheap disk units can be constructed by using a flexible plastic diskette coated with magnetic material. Such devices are called floppy disks. The diskette is enclosed in a cardboard or plastic jacket. The jacket has an opening where the read/write head comes into contact with the diskette. A hole in the center of the cartridge allows a spindle mechanism in the disk drive to position and rotate the diskette. The standard rotational speed is 300 rpm. For the quad-density drive(s) of the PC/AT when used in quad-density mode, the rotational speed is 360 rpm.

Each surface is divided into concentric tracks, and each track is divided into sectors. The track density of the drives initially available with the IBM PC, PC/XT, Portable PC, and PC Jr. was 48 tracks per inch (tpi). With the introduction of the PC/AT IBM also started distributing 96 tpi drives.

Data bits are stored serially on the floppy diskette in fixed "sectors." The number of bytes in a sector is 512. This type of floppy is called "soft sectoring," because the creation and maintenance of the sectors is based on software. The software needs one marker, a hole in the disk, to give it a starting point for the first sector. As the disk spins, the light reaches the sensor when the hole passes in front of the sensor. A pulse is generated and this pulse is sensed by the floppy disk controller. In this way the beginning of each track is found; the remaining sectors on the track are found based on pulse counts away from the index hole.

The great majority of the fields on a given track are devoted solely to enabling differentiation between the different data sectors. Under PC-DOS 2.0 and greater, the only difference between a 9 sector per track diskette and an 8 sector per track diskette is the

number of sectors per track that are in use. DOS formats all standard density diskettes with 9 sectors, but only maps 8 of them.

Addressing (locating) data on the disk is accomplished by specifying the surface number, the track number, and the sector number. The set of all corresponding tracks on all surfaces forms what is called a cylinder.

The effective data capacity for a floppy is calculated in the following manner: bytes/sector x sectors/track x tracks/side x number of recording surfaces. For the quad-density drives of the PC/AT the capacity is $512 \times 15 \times 80 \times 2 = 1,228,800$.

The maximum effective data transfer rate for a 512 bytes/sector, 8 sector/track, diskette is calculated as follows: assuming a rotational speed of 300 revolutions per minute, a complete track can be read or written in $60/300 = 0.2$ seconds. This corresponds to a byte transfer rate of $(8 \times 512)/0.2 = 20,480$ bytes/second. For a 9 sector/track diskette, the byte transfer rate is $(9 \times 512)/0.2 = 23,040$ bytes/second. The rotational speed for a 15 sector/track diskette (the highest supported IBM density in the PC/AT) is 360 rpm. Consequently, the rate is $(15 \times 512)/0.16 = 46,079.9$ bytes/second.

Floppies constitute a form of peripheral storage called removable media. In other units the disk cannot be removed.

Some advantages can be derived if the disk pack and the read/write heads are placed in a sealed, filtered-air enclosure. This approach is known as Winchester technology. In such units the read/write heads can be placed closer to the magnetized track surfaces because no difficulties are encountered with dust particles that are a problem in unsealed assemblies. The closer the heads are to a track surface, the more densely data can be packed along the track. When the disks are moving at their steady rate, air pressure

develops between the surface and the head and forces the head away from the surface. This force can be counteracted by a spring loaded mounting arrangement for the head that allows it to be pressed toward the surface. The flexible spring connection between the head and its arm mounting permits the head to "fly" at the desired distance away from the surface despite any small variations in the flatness of the surface. Moreover, the tracks can be narrower. Thus, Winchester disks exhibit greater density and have a larger capacity for a given physical size. This ultimately results in lower cost units. Another advantage of Winchester technology is that data integrity tends to be greater in sealed units where the storage medium is not exposed to contaminating elements.

Whereas the different floppy diskette formats supported by IBM are derived primarily from software formatting parameters, with extensions to two sides or 15 sectors/track with 80 tracks as opposed to 40, the IBM PC/XT and PC/AT support a variety of disk types. Tables D.1 and D.2 illustrate the different drive types supported by both the IBM PC/XT and PC/AT.

The principles of organization and addressing are the same for both floppies and hard disks. Addressing data on the fixed disk is accomplished by specifying the surface number, the track number, and the sector number.

The effective data capacity for a given fixed disk drive is calculated in the same manner as for a floppy. For a 512 byte/sector, 17 sector/track, 306 track/side, 4 sided drive, the capacity is $512 \times 17 \times 306 \times 4 = 10,653,696$ bytes. For the standard IBM supplied fixed disk drive in the PC/AT, the capacity is $512 \times 17 \times 615 \times 4 = 21,411,840$ bytes.

The maximum effective data transfer rate for a 512 bytes/sector, 17 sector/track, disk is calculated as follows: assuming a rotational

speed of 3600 revolutions per minute, a complete track can be read or written in $60/300 = 0.016$ seconds. This corresponds to a byte transfer rate of $(17 \times 512)/0.016 = 522,239$ bytes/second, very close to the maximum data transfer rate possible on the PC bus with standard RAMs.

In the same sense that the registers are a filter through which all data must pass prior to execution upon within the ALU, so the PC family uses a standard interface to secondary storage. There are two levels to this control, with the first one being at the hardware level.

Communications between a diskette and the CPU is done via a controller chip. This chip is the 8272 Single/Double Density Floppy Disk Controller (FDC) from Intel.

Transfer of data to and from the diskette systems is done via Direct Memory Access (DMA), and that in turn is handled by an Intel 8257-A 64K DMA arbitration device and the corresponding support circuitry. Note that this is a 64K device, and not a 1 megabyte device (which is the addressing range of the 8088). Data transfers can only occur within hardware 64K boundaries.

The ROM Basic I/O System (BIOS) provides the standard access to this chip and its associated functions. The functions provided by the BIOS are:

- Read data: data from the requested location on diskette is written to memory, or put through the processor, depending on DMA implementation.
- Write data: data from memory or the processor, depending on DMA implementation, is written to the specified portions of the diskette.
- Reset: the drive is recalibrated to track 00.

Table D.1: Built-In Drive Types Supported by PC/XT BIOS

	OLD XT				NEW XT			
Drive Type	0	1	2	3	0	1	2	3
Cylinders	306	375	306	306	306	615	306	612
Heads	2	8	6	4	4	4	8	4
MB Capacity	5.3	26.1	15.9	10.6	10.6	21.4	21.2	21.3

Table D.2: Built-In Drive Types Supported by PC/AT BIOS

Drive Type	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Cylinders	306	615	615	940	940	615	462	733	900	820	855	855	306	733
Heads	4	4	6	8	6	4	8	5	15	3	5	7	8	7
MB Capacity	10.6	21.4	32.1	65.4	49.0	21.4	32.1	31.9	117.5	21.4	37.2	52.0	21.3	44.6

- Read status: the status of the FDC is read and transmitted.
- Verify the desired sectors: the specified sectors on diskette are compared to an onboard buffer.
- Format: a track is formatted to user-defined specifications, which is how multiple formats can be supported by this chip.

In the case of the hard disk systems on the PC/XT and PC/AT, IBM has used different hardware implementation schemes. In the PC/XT, data is transferred to and from the hard disk via DMA (with the same 64K boundary limitation). In the case of the PC/AT data is transferred via polled I/O instructions of the 80286 (no DMA).

The functions supported under both systems are similar, however, and they in turn are similar to the floppy disk functions provided for by the Intel chip. The ROM BIOS also provides similar access mechanisms.

This page is blank.

Appendix E:

DOS Disk Space

Tables E.1 and E.2 summarize the different diskette and disk sizes for all diskettes and the standard fixed disk drives available from IBM. This section is an overview of that information.

In PC-DOS versions 1.x, we have to give addresses to a total of 640 sectors. The first 320 sectors are on the first side of the disk and have the same addresses for both single and double sided disks. The second side of a double sided disk has addresses 320 to 639 (140h to 27Fh).

In versions 2.x, the logical sector numbering scheme has changed. All the sectors on a track (or cylinder) are assigned numbers before moving on to the next track. To get a clear picture of this, we will compare the logical sector numbering between versions 1.x and versions 2.x on a double sided diskette formatted for eight sectors per track. This change in versions 2.x minimizes read/write head movement and consequently increases effective data throughput since data access from the other side of the diskette is only limited by the electronic switching speeds of the read/write heads as opposed to mechanical movement time associated with the armature.

Track 0, head 0, has logical sector numbers 0–7. In versions 1.x, the sectors accessed by the second head on that track (track 0, head 1) are numbered 320–327. In versions 2.x, they are numbered 8–15. The consequences of this difference in logical sector numbering are usually hidden from the user, but sometimes you need to know that the difference exists.

In versions 3.x, the logical sector numbering scheme is consistent with that implemented in versions 2.x. The difference here is that

the increase in the number of sectors per track (15 possible with a quad-density derive) causes a corresponding increase in the logical sector numbering, 0–14 for track 0, head 0, and sector numbers 15–29 for track 0, head 1.

Logical sector numbering on a hard disk follows the pattern established with versions 2.x. All the tracks on one cylinder are numbered before moving on to the next cylinder. The sectors read by head zero are numbered 0–16, those read by head one are numbered 17–33, those read by head two are numbered 34–50, and those read by head three are numbered 51–67. The second cylinder begins at logical sector 68.

Tables E.3 and E.4 summarize the overheads and usable capacities of different media as supported by the different releases of PC-DOS. In addition, they show the increase in the size of PC-DOS over the releases. The format of the the overhead with any disk or diskette is the same. There is a boot sector allocated, one File Allocation Table (FAT), and the root directory. Two copies of the FAT are maintained for integrity and so the amount of space allocated to the FAT must be doubled when calculating the amount of available space. The formula is: total number of sectors [from media size calculations before] – (1 boot sector) – (2 x the number of FAT sectors) – (directory space sectors) = total number of usable sectors.

The amount of usable space can be misleading, though. In DOS there is not necessarily a one-to-one correspondence between physical sector and addressable space. The smallest addressable unit of space in PC-DOS is the “cluster.” This “cluster” corresponds to an entry in a vector, the FAT, which is a map of the available space on the medium.

The implications can be seen from Table E.3. Though the total amount of space on a double-sided, nine sector/track diskette is greater than that of a single-sided, eight sector/track diskette, the amount of addressable space in clusters is not that much greater, 354 versus 313. With a cluster size of 1 sector, the smallest amount of space a file can take up is 512 bytes. With a cluster size of 2 sectors, the smallest amount of space a file can take up is 1024 bytes.

This limitation in part stems from the original specification for a FAT entry. The original specification called for a 12 bit number. There can at most be 4093 unique sector numbers. A maximum FAT would occupy 6,144 bytes in storage and 12,288 on disk (two copies). Such a FAT could only track the allocation of 2,095,616 bytes of data if a sector is 512 bytes. Obviously a 10 megabyte disk surpasses this limitation. The cluster size under DOS 2.0 for a hard disk became 8 sectors. Here again, although the smallest physical addressable unit is a 512 bytes sector, the smallest possible file size is 4096 (8 x 512) bytes.

With the introduction of DOS 3.0, and continuing through DOS 3.1, the file allocation table entry size has been modified to support either 12 bits or 16. The maximum addressable number of clusters can now be 65,535. In addition, on the PC/AT hard disk, when it is formatted using PC-DOS 3.x, the cluster size is 4 sectors (2048 bytes). This provides a dramatic increase in the amount of usable space on the PC/AT.

Interestingly enough, if a PC/AT hard disk system has been formatted under DOS 3.x and DOS 2.x is booted on the system, the hard disk will be an invalid drive to DOS. This is because the partitioning specification has also changed under DOS 3.x. The FDISK utility distributed as part of DOS 3.x now can put an indicator into the partitioning header indicating whether or not 16 bit

clusters are used. If, however, DOS 3.x is used to format a standard PC/XT hard disk, the format will be compatible with DOS 2.x.

Table E.1: Diskette Characteristics

FAT byte	# of sides	# of tracks	sectors /track	total sectors	FAT size	dir sectors	dir entries	sectors /cluster	supported by DOS versions
FE	1	40	8	320	1	4	64	1	1.0,1.1,2.0,2.1,3.0,3.1
FF	2	40	9	640	1	7	112	2	1.1,2.0,2.1,3.0,3.1
FC	1	40	9	360	2	4	64	1	2.0,2.1,3.0,3.1
FD	2	40	9	720	2	7	112	2	2.0,2.1,3.0,3.1
F9	2	80	15	2400	7	14	224	1	3.0,3.1

Table E.2: Diskette Storage With and Without Operating Systems Installed

# of sides	sectors per track	DOS version	sectors available		sectors available		clusters available		clusters available		bytes available		bytes available	
			w/o system	w/system	w/o system	w/system	w/o system	w/system	w/o system	w/system	w/o system	w/system	w/o system	w/system
1	8	1.0	313	289	313	289	313	289	160,256	147,986	160,256	147,986	160,256	146,432
1	8	1.1	313	286	313	286	313	286	160,256	146,432	160,256	146,432	160,256	146,432
2	8	1.1	630	603	313	301	313	301	322,568	308,224	322,568	308,224	322,568	308,224
1	8	2.0	313	235	313	235	313	235	160,256	120,320	160,256	120,320	160,256	120,320
2	8	2.0	630	552	315	275	315	275	322,568	281,600	322,568	281,600	322,568	281,600
1	9	2.0	351	273	351	273	351	273	179,712	139,776	179,712	139,776	179,712	139,776
2	9	2.0	708	630	354	314	354	314	362,469	321,536	362,469	321,536	362,469	321,536
1	8	2.1	313	234	313	234	313	234	160,256	119,808	160,256	119,808	160,256	119,808
2	8	2.1	630	551	315	275	315	275	322,560	281,600	322,560	281,600	322,560	281,600
1	9	2.1	351	272	351	272	351	272	179,712	139,776	179,712	139,776	179,712	139,776
2	9	2.1	308	629	354	314	354	314	362,496	321,356	362,496	321,356	362,496	321,356
1	8	3.0	313	196	313	196	313	196	160,256	100,352	160,256	100,352	160,256	100,352
2	8	3.0	630	513	315	256	315	256	322,560	262,144	322,560	262,144	322,560	262,144
1	9	3.0	351	234	351	234	351	234	171,712	119,808	171,712	119,808	171,712	119,808
2	9	3.0	708	591	354	295	354	295	362,496	302,080	362,496	302,080	362,496	302,080
2	15	3.0	2371	2254	2371	2254	2371	2254	1,213,952	1,154,048	1,213,952	1,154,048	1,213,952	1,154,048
1	8	3.1	313	193	313	193	313	193	160,256	98,816	160,256	98,816	160,256	98,816
2	8	3.1	630	510	315	254	315	254	322,560	260,096	322,560	260,096	322,560	260,096
1	9	3.1	351	231	351	231	351	231	179,712	118,272	179,712	118,272	179,712	118,272
2	9	3.1	708	588	354	293	354	293	362,496	300,032	362,496	300,032	362,496	300,032
2	15	3.1	2371	2251	2371	2251	2371	2251	1,213,952	1,152,512	1,213,952	1,152,512	1,213,952	1,152,512

Table E.3: Fixed Disk Operating System Overhead

FAT byte	# of sides	# of tracks	sectors per track	total sectors	FAT size	directory sectors	directory entries	sectors per cluster	supported by DOS versions
XT hard disk, PC-DOS 2.0, 2.1									
F8	4	306	17	20723	8	32	512	8	2.0, 2.1, 3.0, 3.1
XT hard disk, PC-DOS 3.0, 3.1									
F8	4	306	17	20723	8	32	512	4	2.0, 2.1, 3.0, 3.1
AT hard disk, PC-DOS 3.0, 3.1									
F8	4	615	17	41735	16	32	512	4	3.0, 3.1

Table E.4: Fixed Disk Storage With and Without Operating Systems Installed

fixed disk type	DOS version	sectors available w/o system	sectors available w/system	clusters available w/o system	clusters available w/system	bytes available w/o system	bytes available w/system
XT	2.0	20,674	20,596	2,584	2,572	10,584,064	10,534,912
XT	2.1	20,674	20,595	2,584	2,572	10,584,064	10,534,912
XT	3.0	20,674	20,557	2,584	2,568	10,518,528	10,534,912
XT	3.1	20,674	20,554	2,584	2,568	10,518,528	10,534,912
AT	3.0	41,670	41,553	10,417	10,387	21,334,016	21,272,576
AT	3.1	41,670	41,550	10,417	10,386	21,334,016	21,270,528

Appendix F: Interleaving and Performance

In the case of a moving head disk system, there are two time components involved in the delay between receiving an address at the disk controller and the beginning of the actual data transfer. The first, called seek time, is the time required to move the read/write head to the proper track. This obviously depends on the initial position of the head relative to the track specified in the address.

Average values in the 30 millisecond range are typical. The second component is the rotational delay, also called latency time. This is the amount of time that elapses after the head is positioned over the correct track until the starting position of the addressed sector comes under the read/write head. On the average, this is the time for half a rotation of the disk, that is, $0.5 \times (60/3600) = 8.3$ milliseconds, using our specification for a hard disk from before. The sum of these delays is usually called the disk access time. Note that if only a few sectors of data are moved in a single operation, the access time is about an order of magnitude longer than the actual data transfer period.

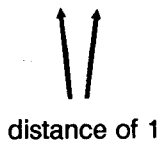
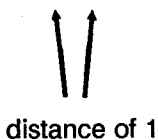
The mechanical components of a disk drive are not alterable, but the physical sequencing of sectors on a disk is. And that is the crux of hard disk interleaving. To define sector interleaving, picture a disk platter spinning in a disk drive. The sectors in the track over which the head is positioned are passing by the head one after another—sector 1, sector 2, and so on—until the disk has turned one complete revolution. Then the sequence repeats itself. A standard IBM fixed disk spins at 3600 rpm. One revolution of the diskette takes $60/3600$ seconds, about 0.0166 seconds per track, or 0.9 milliseconds a sector.

Now imagine DOS loading a program or accessing a file from such a disk. DOS takes a finite amount of time to read and process each sector. DOS normally requests each sector in sequence from within a cluster. It takes a finite amount of time for each request. Assume that it has to make repeated reads to load the file or program. By the time DOS has read and loaded sector n , it will be too late to read sector $n + 1$. This sector will have already passed by the head and will not come around to position for another 16.6 milliseconds. Proceeding in this fashion, almost 2 tenths of a second are needed to read one complete track (this small amount may seem ludicrous, but these amounts do add up over time).

This problem can be overcome by simply numbering the sectors so that there are several physical sectors between each numerically contiguous sector.

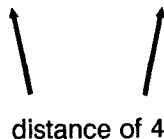
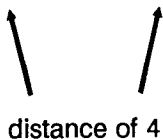
For example, with an interleave factor of 1, the distance between sectors is 1:

Sector #: 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17.



With an interleave factor of 4, the distance between each sector is 4:

Sector #: 01 14 10 06 02 15 11 07 03 16 12 08 04 17 13 09 05



Depending on the clock speed of the system, the processor can now do either a little or a lot after a sector transfer. Normally, the higher the clock speed the smaller the interleave. IBM has specified interleaves for their clock speeds. For those users who have increased their clock speeds, or are running clones at higher clock speeds, these interleave factors are not adequate.

Depending on the size and type of files, the wrong interleave can strongly affect performance. It is not a gradual effect, either: if you "miss" the interlace, the perceived performance will be very slow. One of the nice features about PC-DOS is its use of physical sector numbers in addressing media. This feature leaves the interleaving separate and independent of DOS.

This page is blank.

Appendix G: HTEST/HFORMAT Error Codes

The controller will return one of the following hexadecimal codes to HTEST or HFORMAT if an error occurs while reading or writing a sector. The programs list these errors in the "R/C" (return code) column of their displays, adding an "h" to indicate that they are in hexadecimal (for example, "0A" appears as "0Ah").

01 Bad command passed to disk I/O

The controller received an invalid command, either because it is incompatible with the program, or because it wasn't initialized.

02 Address mark not found

The controller could not find a given surface, track, or sector, usually because the value supplied was outside the disk's range. Make certain that the parameter list accurately reflects the drive's characteristics. This error can also occur if an area of the disk is so badly damaged that the controller can't sense the head position relative to cylinder zero.

04 Record not found

The controller failed to find a sector mark. This could be caused by a surface defect or by a failure of the controller's circuitry.

05 Reset failed

The drive failed to respond to a reset (seek cylinder zero) command. Turn off the power, wait twenty seconds, and turn the power back on. If this error persists, the controller or disk needs service.

07 Drive parameter activity failed

The controller cannot retain the drive characteristics in its memory, and it definitely needs service.

09 Attempt to cross 64K DMA boundary

The controller tried to copy a sector into memory across an address boundary, because the program was loaded too high into memory. Check that no memory-resident programs are running before starting HTEST or HFORMAT.

0A Bad sector flag detected

A previous program marked this sector to prevent its use. This is a normal occurrence.

0B Bad track flag detected

A previous program marked this track to prevent its use. HFORMAT and other formatters write these marks on tracks that have problem sectors.

10 Bad ECC on disk read

The Error Checking and Correction (ECC) routines store a value in a sector which is used to check whether it has been written or read correctly. In this case, the check value didn't match the sector contents, indicating unreliable reads or writes, and the sector should be locked out.

11 ECC corrected data error

The Error Checking and Correction (ECC) routines store a value in each sector as a quick check on each read or write. In this case, the controller's routines found the check value was inaccurate, but succeeded in reading the sector correctly. The sector should be locked out.

20 Controller has failed

The controller did not pass its own testing routines. Turn off the power, wait twenty seconds, and turn the power back on. If this error persists, the controller needs service.

40 Seek operation failed

The drive could not locate a given cylinder, usually because the value supplied was outside the disk's range. Make certain that the parameter list accurately reflects the drive's characteristics. This error can also occur if an area of the disk is so badly damaged that the controller can't sense the head position relative to cylinder zero, or the controller may need service.

80 Attachment failed to respond

The drive is not responding to the controller's commands. If it is properly connected then the drive probably needs service.

AA Drive not ready

The controller cannot get a "ready" signal from the drive, usually because of a faulty cable or power connection, but the drive or controller might also need service.

BB Undefined error occurred

The controller noted an error but could not determine its cause. Both the drive and controller should be checked.

CC Write fault

The controller noted an error during a write or format operation, usually because the drive's printed circuitry is failing and needs service. Replacing the controller board with an **identical** one will usually cure the problem.

FF Sense fail

The controller noted an error, but either the drive could not supply information as to its cause, or else the error is unknown. Both the drive and controller should be checked.

*h*TEST · *h*FORMAT is an advanced hard disk diagnostics program. *h*TEST allows exhaustive testing for, and locking out of bad sectors at the controller level. *h*FORMAT allows flexible low-level formatting of the hard disk, interleave adjustment, and easy entry of manufacturer's bad track information. Included is *h*OPTIMUM, which tests for optimum disk interleave, without re-formatting! *h*PERF performs a series of hard disk performance tests. *h*TEST · *h*FORMAT is compatible with all variations of PC and MS-DOS, version 1.1 through 3.3 and supports disks up to 1024 cylinders and 16 heads. Now supports disks using RLL encoding!

Paul Mace Software, Inc.
400 Williamson Way
Ashland, OR 97520

(503) 488-0224

*h*TEST · *h*FORMAT runs on IBM PC, XT, AT, and compatible computers. *h*TEST · *h*FORMAT works with standard MFM or RLL hard disks with up to 16 heads and 1024 cylinders. Supports Adaptec, DTC, OMTI, Western Digital and Xebec controllers. *h*TEST is not copy protected.

Requires 64K RAM and MS or PC-DOS version 1.1 or higher.