HDB-DOS 1.0

User Manual

Manual Revision 1.1 - June 18, 2002

Cloud-9

Cloud-9
3749 County Road 30
Delano, MN 55328
Telephone 763.972.3261
www.cloud9tech.com

Cloud-9

Cloud-9
3749 County Road 30
Delano, MN 55328
Telephone 763.972.3261
www.cloud9tech.com

C/H/S vs. LBA Mode

In order to accommodate the widest range of drives, HDB-DOS supports the C/H/S (Cylinder/Head/Sector) method of accessing IDE drives. While using C/H/S mode does add some overhead when processing each read/write, it insures the greatest compatibility with a wide range of IDE drives.

Unlike C/H/S mode, the LBA mode is a more direct approach to accessing a hard drive's sectors, and requires very little overhead, but is only supported on drives of larger (> 2GB) capacity. Since CoCo users are inclined to use smaller drives, we felt that supporting C/H/S mode was priority. In the near future we may support LBA mode as well.

Table Of Contents

13. IDE Notes

Supported IDE Interfaces

HDB-DOS currently supports the following IDE interfaces:

- Glenside Color Computer Club IDE interface

An IDE interface, unlike SCSI, is not an intelligent controller. It is merely an interface to the IDE drive(s) on the IDE bus. This distinction is important, and affects the way that HDB-DOS starts up for IDE vs. SCSI.

HDB-DOS will wait approximately 5 seconds for the IDE device to spin up before it fails with a HARD DRIVE NOT FOUND message. This timeout occurs whether or not the controller is plugged into the CoCo. It may appear that the CoCo is locked up, but HDB-DOS is actually waiting for the IDE drive to spin up and become available for detection.

You can change the amount of time by changing the byte at $D93D. Each value represents $1/10^{th}$ of a second, so a value of 10 at the address would equal a 1 second wait before timing out (be aware that running at high-speed will cut this time in half).

ATA-2 vs. ATAPI

HDB-DOS supports ATA-2 compliant devices only. ATAPI devices such as CD-ROMs and ZIP drives will not work with HDB-DOS.

12. SCSI Notes

Supported SCSI Controllers

HDB-DOS currently supports the following SCSI controllers:

- TC^3 SCSI Controller from Cloud-9
- Ken-Ton SCSI Controller from Ken-Ton Electronics (out of production)
- Disto Hard Disk II Adapter (out of production)
- Disto 4-N-1 Hard Disk Adapter (out of production)

If you are using HDB-DOS on a SCSI controller, please note the following:

1. Upon startup, HDB-DOS will attempt to detect the presence of the supported SCSI controller by doing a write/read check on the controller's data port. Since the controller is immediately available on power-up, the detection happens quickly. However, it may take time for a hard drive to spin up, so HDB-DOS waits indefinitely until the default hard drive has powered up. Once it is ready, a beep will be emitted and the hard drive will be activated.

2. Some controllers may not support the full 0-7 SCSI ID range, nor do all controllers support parity, which is necessary for some drives. This is not a flaw in HDB-DOS. It is an inherent flaw in the design of some SCSI controllers.

1. Congratulations!

Thank you for purchasing HDB-DOS, another one of the fine products offered by Cloud-9. This product was designed to be used with the Radio Shack TRS-80 Color Computers 1, 2, and the Tandy Color Computer 3.

HDB-DOS (Hard Disk BASIC Disk Operating System) started out in 1986 as RGB-DOS, a ground-breaking product of Roger Krupski and RGB COMPUTER SYSTEMS. It was paired with the fine SCSI controller from Ken-Ton Electronics, and versions were also available for other hard drive controllers.

With the introduction of the TC^3 SCSI controller from Cloud-9, renewed interest in RGB-DOS began to appear. In 2002, Acadian Embedded Solutions acquired all rights to the product from Ken-Ton and now licenses the product exclusively to Cloud-9 as HDB-DOS.

HDB-DOS is reliable and easy to learn, but you have to KNOW HOW TO USE IT if you expect maximum performance. We suggest that some time be spent reading the User Manual before attempting to use HDB-DOS with your hard drive system. Over the years, several people have spent long hours at the keyboard writing the software and the User Manual, so please spend a few minutes and read the manual first. It will save you time, money and frustration.

HDB-DOS contains the best ideas of many CoCo enthusiasts. Countless ideas were given and countless

suggestions made. Some ideas were great, and became part of HDB-DOS. Most ideas were rejected for technical reasons. Although we feel HDB-DOS is a fine product, we never consider it "finished". There is always something that can be improved. If you have an idea, comment or suggestion, please pass it along to us.

Features

- Allows access to modern SCSI and IDE hard drives under BASIC.
- Can address up to 8 SCSI devices or 2 IDE devices.
- Up to 256 “virtual floppy” disks can be accessed.
- Works flawlessly with Cloud-9’s TC^3 SCSI Controller.
- Also supports Disto and Ken-Ton SCSI controllers, and the Glenside IDE interface.

Be aware that in order for this to work properly, your OS-9 boot track must contain a “boot” module that understands the controller that you are using. If you do not do this, then the boot track will attempt to find the OS-9 boot file on the floppy disk, not the hard drive.

NOTE!! Under OS-9, RBF sees sectors as 256 bytes in size. If you have a drive with 512 byte sectors, then you will need to double the sector value from 176,448 to 352,896 in order to get the full use of the OS-9 side of the hard drive. Also, it is imperative that you set the device descriptor for the ZIP-100 disk to reflect 352,896 as the number of sectors. If you specify a larger value, you run the risk of OS-9 writing over the HDB-DOS partition of the hard drive.

The good news is that the included WIZARD.BAS program will calculate this offset for you automatically, based upon the number of sectors of your hard disk and the number of HDB-DOS hard disks you wish to have.

"No Floppy" OS-9 Boot

HDB-DOS comes with a BASIC program named LINK, which, when run, prompts you for an HDB-DOS hard drive number where an OS-9 boot disk resides on. It then sets up the OS-9 partition to boot from that OS-9 boot disk, which is actually located on the HDB-DOS partition.

To use this feature, create a 35 track, single-sided OS-9 boot disk. Reboot into HDB-DOS and use the BACKUP command to copy the disk from the floppy drive to one of the HDB-DOS hard disks. Then, run LINK3.BAS (LINK12.BAS if you have a Color Computer 1 or 2) and provide the drive number of the HDB-DOS hard disk where the newly backed-up disk resides (for example, hard disk 20). Once the program is finished, you can type `DOS 20` to boot into OS-9 from the image on HDB-DOS hard disk 20.

2. EPROM Installation

If you have received HDB-DOS on an EPROM, then follow these steps to install the EPROM on your Color Computer Disk System. If you don't have an EPROM, then skip to the next chapter.

IMPORTANT !!! NEVER, NEVER Insert or remove ANY PROGRAM PAK or ROM PAK from the Color Computer while the POWER IS ON! To do so may damage the Computer and/or the Program Pak!

(1) Turn OFF the Computer, Multi-Pak Interface, and ANY OTHER equipment attached to the computer.

(2) Disconnect the cable from the DISK CONTROLLER to the FLOPPY DISK DRIVE(S) (34 conductor ribbon) and remove the disk controller.

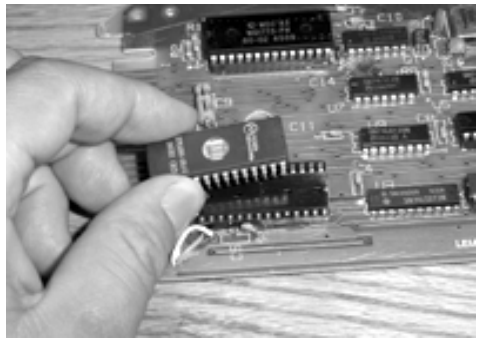
(3) Using a finger, feel the LABEL on the DISK CONTROLLER, looking for a small indentation which is the single screw holding the 2 parts of the plastic case together.

(4) Using a PHILLIPS screwdriver, puncture the label and remove the screw by unscrewing it right out of the hole you made in the label.

(5) Carefully unsnap the top and bottom pieces of the controller case, exposing the printed circuit board inside.

(6) Locate the ROM chip. The ROM chip is the SMALLER of the 2 LARGE CHIPS. Some disk controllers will have 2 "Large" chips and several "Small" chips. The largest chip will have either 28 or 40 pins (count them!). The smaller of the 2 will have 24 pins. This is the ROM chip. Most "after-market" floppy disk controllers (non-Radio Shack) have a 28 pin ROM socket, or even have several ROM sockets. The Tandy FD-502 also has a 28 pin ROM Socket. If you are not ABSOLUTELY sure which chip to remove, DO NOT DO ANYTHING; contact Cloud-9 for assistance!

(7) Take note where PIN 1 of the ROM CHIP is. It is absolutely ESSENTIAL that the NEW CHIP be inserted properly, that is PIN 1 to PIN 1. A small dot or notch will indicate where pin 1 is. Installing the chip backwards will DESTROY THE CHIP and VOID THE WARRANTY! Draw a sketch if necessary!



(8) Using a small screwdriver, CAREFULLY insert the tool between the CHIP and the SOCKET and pry up JUST A BIT, then pry the other end and SLOWLY, CAREFULLY rock the chip up and out.

1. How many sectors does your hard drive have?
2. How many HDB-DOS Virtual Floppy Disks do you wish to have?

Example

Let's take an example using a ZIP-100 disk, which has 196,608 sectors. We wish to allocate enough space for 32 Hard Disks, and devote the rest to OS-9.

An HDB-DOS Hard Disk contains 630 sectors. Since we wish to allocate 32 Hard Disks, we multiply the number of sectors of one Virtual Floppy, 630, by 32, and get the result: 20,160.

We then subtract the number of HDB-DOS Hard Disk sectors, 20,160, from the total number of sectors, 196,608:

196,608 – 20,160 = 176,448

The result is the OS-9 offset, or number of sectors from the beginning of the drive to the FIRST HDB-DOS Virtual Floppy. However, the number must be converted to hexadecimal.

176,448 decimal = $02B140

This 3 byte value can then be placed into offsets $1938-$193A of the HDB-DOS ROM (assuming a base address of 0).

11. OS-9 and HDB-DOS

For OS-9 users, HDB-DOS offers several advantages. It also allows both OS-9 and HDB-DOS partitions to exist on the same physical hard drive. It even allows full booting to OS-9 from a hard drive without the need for a floppy drive.

HDB-DOS Offset Feature

For OS-9 users, HDB-DOS offers a feature that allows both an OS-9 partition AND HDB-DOS Hard Disks to reside on the same physical Hard Drive. The 3-byte sector offset, which is located at $D938-$D93A, is added to all Hard Disk accesses under HDB-DOS, and represents the number of sectors from the beginning of the hard drive to the FIRST Virtual Floppy on the Hard Drive.

Please note that the included WIZARD.BAS file will calculate the necessary OS-9 offset value for you automatically. The following information is only presented as a thorough technical explanation of the feature. Also, if you are using multiple hard drives on your system, the OS-9 offset is applied to ALL of the hard drives. This may not be what you want, so be aware. An alternate solution would be to leave the OS-9 offset at $000000 and devote a separate hard drive just for OS-9.

In order to properly calculate the HDB-DOS offset, you must first determine several things:

(9) Remove the NEW ROM CHIP from the protective foam and place the OLD ROM CHIP into the foam for safe keeping.

(10) Insure that no pins on the ROM chip are bent, then place the chip in the socket and firmly press to seat it. DO NOT FORCE IT! Be absolutely sure that no pins are bent before seating the ROM CHIP.

(11) Visually inspect the installation: Is the chip BACKWARDS? Are any pins BENT? Is the chip in the RIGHT PLACE?

When you are satisfied with the installation, replace the cover of the DISK CONTROLLER and re-install the screw.

3. Using the HDB-DOS Wizard

The HDB-DOS Wizard is a BASIC program supplied on the HDB-DOS distribution diskette. Its purpose is to set up a custom copy of HDB-DOS for your system in a very short time, by asking you a series of questions about your controller type, addressing and a few other options.

To run the program, insert a backup copy of your HDB-DOS disk into floppy drive 0 and type:

```
RUN "WIZARD:0"
```

NOTE: It is important that you run WIZARD.BAS from a system with RS-DOS 1.1. DO NOT run WIZARD.BAS from HDB-DOS or another DOS.

Sit back, relax and answer the few questions that the Wizard puts your way. In no time, you will have a custom loadable HDB-DOS.BIN file.

While running HDB-DOS in this manner will work, we recommend that you burn the created EPROM.DOS image into an EPROM and insert it into your disk controller ROM socket at some point. This is the best way to take advantage of HDB-DOS.

DSK2DSK.BAS

A copy program for floppy/hard drive copies. *Runs on a 512K Color Computer 3 ONLY.*

FINDFILE.BAS

A handy file find utility that searches for a filename across the virtual floppys of the hard disk. *Runs on all Color Computers.*

ERRMSG.BIN, ERRMSG.SRC

A patch which gives detailed disk I/O related error messages. *Runs on all Color Computers.*

RAMDISK.BIN, RAMDISK.ASM

A patch for HDB-DOS users which turns drives 2 and 3 into RAM Disks under Disk BASIC. Source is included. *Runs on a 512K Color Computer 3 ONLY.*

TWPATCH.BAS, TWPATCH.BIN, TWPATCH.ASM

A patch for Telewriter 80 to allow use of HDB-DOS drives. Source for the patch program is included. *Runs on the Color Computer 3 ONLY.*

FIX 128.BAS

A BASIC program which patches Telewriter 128 to allow use of HDB-DOS drives. *Runs on the Color Computer 3 ONLY.*

10. HDB-DOS Utilities

HDB-DOS comes with a set of BASIC and machine language utilities that you will find useful. Please note that some programs are for specific models of the Color Computer.

WIZARD.BAS

This BASIC program allows you to configure a custom copy of HDB-DOS to either load into memory via the `LOADM` command, or burn onto an EPROM to put in your disk controller. You will be prompted with several configuration questions, and once completed, you will have two files: EPROM.DOS and HDB-DOS.BIN. The former is intended to be burnt into an EPROM, and the latter is to be loaded into memory and executed. *Runs on all Color Computers.*

LINK3.BAS

This program "links" the OS-9 partition to an OS-9 Hard Disk on the HDB-DOS side of the hard drive. This allows booting completely from the hard drive without needing a boot floppy. *Runs on the Color Computer 3 ONLY.*

LINK12.BAS

This program performs the same function as LINK3.BAS, but is for the 64K Color Computer 1 and 2.

4. Choosing A Hard Drive

When it comes to choosing a hard drive to work with HDB-DOS, it is important to understand the limitations that Disk BASIC and HDB-DOS impose. At the time of this writing, in 2002, hard drives in excess of 40 gigabytes can be routinely purchased for under $100, with 20 and 10 gigabyte drives going even cheaper. This amount of storage was barely conceivable when HDB-DOS started out in 1986. Consequently, HDB-DOS will utilize only a small portion of today's hard drives -- most of the space will go unused.

Sector Sizes

Virtually all IDE and SCSI drives utilize 512 byte sectors, while the Color Computer's Disk BASIC was designed with 256 byte sectors in mind. Short of a complete rewrite of Disk BASIC, it is difficult to support full 512 byte sector hard drives AND maintain compatibility with Disk BASIC programs. This does not mean that HDB-DOS will not work with 512 byte sector hard drives. It does mean, however, that HDB-DOS will only utilize the first 256 bytes of each sector, leaving the remaining 256 bytes unused.

Virtual Floppy Disks

In order to maintain compatibility with Disk BASIC, HDB-DOS breaks up a hard drive into a number of "virtual floppy disks." Each one of these virtual floppy disks resembles a 35 track, single sided floppy drive in size, for a total of 630 sectors. Up to 256 virtual hard disks (0 – 255) can be accessed, depending upon the size of the drive. HDB-DOS

will query the drive for its size and will adjust the maximum number of virtual hard disks accordingly.

Putting It Together

With this information, let's look at an example of how HDB-DOS utilizes the space on a Quantum LPS52 SCSI hard drive.

The Quantum LPS52 SCSI has a total of 102,136 sectors at 512 bytes per sector, for a grand total of approximately 52 megabytes. Assume that we want to allocate the entire hard drive for HDB-DOS. When HDB-DOS starts up, it queries the hard drive and finds that it has 102,136 sectors. It then divides that number by 630, the number of sectors per virtual floppy disk (remember, 630 is the same number of sectors of a 35 track, single-sided floppy disk).

102136 / 630 = 162.1206

HDB-DOS ignores the fractional part, and sets the number of virtual floppy disks to 162 for that hard drive (0 – 161).

Now, since HDB-DOS only utilizes half of the hard drive, the actual number of bytes being used is really only approximately 25 megabytes. Remember, HDB-DOS only utilizes the first 256 bytes of each sector.

So how many sectors does a hard drive need in order to utilize the full 256 virtual floppy disks under HDB-DOS? Multiply the maximum number of virtual disks by the number of sectors per virtual disk:

256 * 630 = 161,280

$0151	IDNUM	1 byte containing the SCSI or IDE ID number. This byte is normally set to 0 but may be changed to allow access to other devices on the SCSI or IDE bus. The DRIVE # command controls this byte.

		Note that SETUP [$D932] only sets up 6 byte commands. If extended (10 or more) byte commands are used, the Command Descriptor Block must be set up by the user.
$014A - $014B (IDE ONLY)	CYLINDERS	2 bytes containing the number of cylinders for the current device.
$014C (IDE ONLY)	HEADS	1 byte containing the number of heads for the current device.
$014D (IDE ONLY)	SECTORS	1 byte containing the number of sectors for the current device.
$014E	HDFLAG	1 byte containing the drive number that begins hard drive access. The DRIVE ON and DRIVE OFF command sets this variable.
$014F	DRVSEL	1 byte for Logical Unit Number (LUN) selection. The POKE command can be used to change this byte. Note that the IDE version of HDB-DOS does not use LUN.
$0150	MAXDRV	1 byte containing the highest drive number available to the system. The contents of this byte are determined by the size of the attached hard drive and should not be changed. Making the number larger will not provide more storage but will simply cause an ?IO ERROR.

And multiply the product by the number of bytes per sector, 512:

161,280 * 512 = 82,575,360, or 82 megabytes

In this case, half of the space, approximately 41 megabytes, will be wasted. Given the size of today's hard drives, this is a minimal amount of loss. Although some may not like the waste, we believe that the problems associated with fully supporting 512 byte sector devices far outweigh any potential benefits. Remember: compatibility with existing Disk BASIC applications has always been an advantage of HDB-DOS.

5. Hard Drive Setup

Once you have selected your hard drive, you will find that setup is simple and straight forward. As always, the instructions should be followed carefully if the best results are expected.

(1) Turn OFF power to the Color Computer, Multi-Pak Interface and any other equipment connected to the computer.

(2) Locate your hard drive data cable and connect it to your hard disk interface and the hard drive unit(s). Be sure to install the cable properly (Pin 1 to Pin 1, etc.).

(3) Plug your hard disk interface into the Multi-Pak Interface or "Y" cable. If using the Multi-Pak Interface, set the selector switch to the slot that contains the HDB-DOS EPROM. If your HDB-DOS EPROM is installed in your Floppy Disk controller set your slot selector switch to the Floppy Disk Controller's slot. If the EPROM was preinstalled in your hard disk interface, set your slot selector switch to the interface's slot.

(4) Plug any other accessories you may have into the Multi-Pak Interface or "Y" cable, including the Floppy Disk Controller.

$D93B (PORTHI) – Port Address, HI BYTE.

Reflects the address of the SCSI controller in the CoCo's memory map.

$D93C (PORTLO) – Port Address, LO BYTE.

Reflects the address of the SCSI controller in the CoCo's memory map.

$D93D (TIMOUT) – Timeout Value.

For SCSI this is used as a command timeout. For IDE, this value is used at the startup of HDB-DOS to indicate how long to wait for an IDE device to spin-up and become available (in 1/10ths of a second).

$D93E (DEFID) – Default SCSI/IDE ID.

This is the SCSI/IDE ID of the main HDB-DOS drive. In the case of SCSI, some drives, such as ZIP drives, can only be set to SCSI IDs 5 or 6, and this is where that ID would be placed. IDE allows only the values 0 (master) or 1 (slave).

RAM Variables

HDB-DOS utilizes certain portions of low RAM for its own use. Here are the offsets and their designations.

Address	Name	Usage
$00F3	VCMD	Up to 10 bytes used to store the command data which is sent to the SCSI/IDE device during the COMMAND phase. This VCMD+9 buffer is also used to read SCSI sense (error) information.

$D932 (SETUP) - Setup a CDB in RAM starting at VCMD

Passed: A = Controller Command Op-code
B = LUN / Sector Address, MS Byte
X = Sector Address, LS Word
Y = Block Count / Options

Returned: 6 Byte Command, setup at VCMD ($00F3)
LUN set as per state of DRVSEL ($014F)

Registers Modified: NONE

$D934 (BEEP) - Used to generate a "BEEP" tone

Passed: NOTHING

Returned: BEEP and AUDIO OFF

Registers Modified: NONE

$D936 (DSKCON) - DSKCON Re-entry point

Used to return to DSKCON if the original entry point has been patched such as by a RAM disk program.

$D938 (OS9HI) - OS-9 sector offset, HI BYTE.

Set to $00 for an all RS-DOS system.

$D939 (OS9LO) - OS-9 sector offset, LO WORD.

Set to $0000 for an all RS-DOS system.

(5) Plug the hard drive unit's LINE CORD into a GROUNDED WALL OUTLET. For safety's sake, do NOT defeat the grounding feature of the line cord!

(6) Visually inspect the setup for loose or forgotten connections. A few seconds double checking the setup NOW will save hours of grief LATER!

(7) Switch on all components. The computer should start up as normal, but there will be no cursor UNTIL THE HARD DISK REACHES OPERATING SPEED. It is usually better to switch on ALL components at once using an "Outlet Strip" or other multi outlet device. Otherwise, switch the HARD DISK UNIT on FIRST and wait until the Hard Drive light goes off before switching ON the computer.

(8) When the Hard Drive is READY, a "BEEP" will be heard from the Computer if the volume is up sufficiently. At this time, the "OK" prompt appears and the usual blinking cursor should be seen. The system is now READY TO RUN!

If the computer does not act normally, SHUT IT OFF and re-check all wiring. REMEMBER that the computer will NOT display a cursor UNTIL the hard drive has reached full speed and passed the self test. If the hard drive does NOT become ready after 30 seconds, the system will automatically switch to the floppy disk drives and the hard disk will NOT be available. This is, of course, not normally expected and indicative of a problem such as a bad drive, no power on the drive, loose wires etc...

Once the system is up and running, proceed to use it as you would a floppy disk. A good place to start is by copying a few of your favorite disks to the hard drive and see how they run.

Programs which will ONLY run with drives 0 thru 3 can be easily transferred with the following steps:

1. Type `DRIVE OFF` <ENTER> to ENABLE the floppy disk drives.
2. Insert your floppy disk into DRIVE 0
3. Type `BACKUP 0 TO 4` <ENTER> to copy the disk to hard disk #4
4. Type `DRIVE ON` <ENTER> to ENABLE hard disks #0 thru 3
5. Type `BACKUP 4 TO 0` <ENTER> to copy the data to hard disk #0
6. Type `DSKINI 4` <ENTER> to erase hard disk #4
7. The message `ERASE HARD DISK 4 ... ARE YOU SURE? (Y/N)` appears.
8. Respond by hitting the "Y" key for YES. Hard disk #4 will be erased.

Your floppy disk is now copied onto hard disk 0!

Now, have some FUN! Select a program that you would like running AS SOON AS the computer is turned on. Let us assume that the name of this program is "GAME/BAS". Type in the following program:

```
10 CLS ' just clear the screen
20 RUN"GAME/BAS" ' load in & run
```

Now, type:
`SAVE"AUTOEXEC"` <ENTER>

Your "GAME/BAS" program will come up as soon as the computer is turned on without the need to touch the keys at all!

9. Technical Information

There are several ROM routines and memory locations which may be of some use to both Assembly Language and BASIC Programmers. The following is a list of the routines which are accessed using EXTENDED INDIRECT ADDRESSING (i.e. `JSR [$D9301]`).

NOTE: This information is presented for the benefit of EXPERIENCED PROGRAMMERS. If you are NOT SURE of what you are doing, DON'T DO IT! A seemingly small error could corrupt some data on the Hard Disk, or ERASE IT ENTIRELY! Cloud-9 CAN NOT and WILL NOT be responsible for the use or misuse of this information. Peek and Poke at your OWN RISK!

$D930 (DISKIO) - Universal Hard Disk I/O Routine

Passed: Command Packet starting at VCMD ($00F3)
Read/Write/Data address in DCBPT ($00EE)

Returned: Command Executed
Completion status in DCSTAT ($00F0)
If NO ERROR, DCSTAT ($00F0) = 0
If ERROR, DCSTAT = Hard Disk ERROR CODE

Regs Modified: NONE

As with the LOADM command, an optional LOAD OFFSET may be specified in addition to the FILENAME. To LOADM and EXEC a Binary Program with an Optional Offset, the following syntax is used:

EXAMPLE:
`RUNM"FILENAME",&H1000` OR `RUNM"FILENAME",4096` <ENTER>

Both of the above examples will load the Program 4096 bytes higher ($1000 bytes higher) than usual. The "EXEC" address is also moved accordingly. If the program is loaded with the Optional Load Offset it will, of course, run ONLY IF it was written in P.I.C. (Position Independent Code). Programs NOT written in P.I.C. will likely crash or run unpredictably if a load offset is used.

Finally, the RUNM command should be used with binary (Machine Language) programs which require an "EXEC" after loading. Programs which "Auto-start" MAY crash or run unpredictably, since the "EXEC" function ends up being done twice. If the RUNM command fails to properly load a program, try using "LOADM" and see if this solves the problem.

If your favorite program were BINARY (machine language), such as "GAME/BIN", line 20 would look like this: (note the new command, RUNM)

```
20 RUNM"GAME/BIN" ' loadm & exec at once
```

NOTE: HDB-DOS gives your hard drive the ability to hold the equivalent of MANY floppy disks. Even so, things can go wrong. The WISE USER will keep BACKUP COPIES of important work on FLOPPY DISK! It is VERY EASY to forget that floppy disks even EXIST once you get used to the hard drive. So, as always, remember to KEEP BACKUPS of important work!

6. Special Features

Auto Execution Feature

HDB-DOS has a unique feature which allows ANY program to be automatically loaded and running WITHOUT the need to even touch the keyboard! This feature is very handy for loading in utilities or menu programs, performing custom "Pokes", setting up special screen modes or any other operations that you wish to have done EVERY TIME the system is started. The "AUTOEXEC" feature can also make any Bulletin Board System (BBS) completely self-restarting, thereby minimizing downtime.

Whenever the Color Computer is first turned ON or a full "Cold Reset" is performed, the system will do the following things:

1. The system will be checked for the presence of a hard drive and the software will adjust itself to the characteristics of the type of hard drive installed, if any.

2. The hard drive will be tested to see if it is READY.

3. The system will BEEP if the hard drive is READY. The system will NOT BEEP and give a HARD DRIVE NOT FOUND message if the hard drive is NOT READY. The system will then attempt to find "AUTOEXEC/BAS" from Floppy Disk, Drive ZERO.

COMMAND: RUN, RUNM

SYNTAX and USAGE:
RUN"*filename*"
RUNM"*filename*"
RUNM"*filename*",*offset*

Just as the RUN command will LOAD and RUN a BASIC program automatically, so to will the RUNM command LOADM and EXEC a BINARY program automatically.

The DEFAULT EXTENSION for the RUN command is "BAS" meaning that it will LOAD and RUN any BASIC PROGRAM. The extension may be included in the filename if the program was saved with an extension other than "BAS".

EXAMPLE: RUN"PROGRAM" <ENTER> will load and run a BASIC program saved as "PROGRAM/BAS"

If the program were saved with a different EXTENSION other than "BAS", then that extension MUST be specified in the RUN command.

EXAMPLE: The program is BASIC, but saved with the name "PROGRAM/PRO" then the BASIC command then MUST BE RUN"PROGRAM/PRO" <ENTER>

The same is true with the RUNM command. However, the DEFAULT EXTENSION for BINARY PROGRAMS is "BIN". Therefore, the RUNM"PROGRAM" command will look for a program saved with the filename "PROGRAM/BIN". If the Binary Program were saved with an extension OTHER THAN "BIN", then the Extension MUST be specified in the RUNM command.

That command would place a BLUE graphics square before and after the Diskname string.

To set the DISKNAME apart from the Directory contents, it is suggested (but not necessary) to use stars (*) or any other characters before and after the DISKNAME, such as: `RENAME DRIVE 51,"*** BINARY GAMES DISK #2 ***"` <ENTER>

Finally, the DISKNAME may be removed by simply "RENAME"ing a Drive with a "Null" string: `RENAME DRIVE 51,""` <ENTER> ... removes the DISKNAME from Drive 51. The Directory then displays only the Filename(s), the Drive Number and the Free Granules remaining on the "drive" WITHOUT the DISKNAME.

Technical Info: The DISKNAME string is stored on Track 17, Sector 17 of each "drive" on the hard disk or floppy disk. You may also read/write this string with BASIC's DSKI$ and DSKO$ commands. If the DISKNAME string is written using the DSKO$ command, be sure to terminate it with a CHR$(0). When the RENAME DRIVE command is used, the balance of the DISKNAME sector (256 - string length) is filled with CHR$(255).

4. The system will look for a BASIC program on hard disk ZERO called “AUTOEXEC/BAS". If "AUTOEXEC" IS found, it will be loaded and run by the system automatically. If it is NOT found, the system will return to the BASIC OK prompt.

The program "AUTOEXEC/BAS" may be saved in "Crunched" (Standard) format or in ASCII (the "comma-A") format. "AUTOEXEC/BAS" MUST be a BASIC program. It may NOT be a Machine Language Program. However, if you wish to automatically run a Machine Language Program upon startup, you may use the RUNM command in a BASIC program, then save that program with the name "AUTOEXEC" on drive ZERO. BASIC will automatically assign the "BAS" extension if none is specified. Note that the program MUST be named "AUTOEXEC/BAS". If a different filename OR extension is used, the system will NOT find and auto-execute it.

Sometimes it is desirable to BYPASS "AUTOEXEC" and NOT run the usual program upon startup. The "AUTOEXEC" feature may be bypassed, if desired, by pressing and HOLDING either or both SHIFT keys during startup until the "OK" prompt appears.

FLEXIKEY - Last Line Recall & Edit

FlexiKey was written by Colin J. Stearman, © 1984, and is included in HDB-DOS with the written permission of the author.

The basic function of FlexiKey is to allow the RIGHT ARROW Key of the Color Computer to perform the OPPOSITE function of the LEFT ARROW. That is, the LEFT ARROW will DELETE ONE character at a time and a SHIFT-LEFT ARROW will delete an ENTIRE line at a time as usual.

FlexiKey now allows the RIGHT ARROW to RECALL ONE character at a time and the SHIFT-RIGHT ARROW to recall an ENTIRE LINE. Once the line has been recalled, pressing the <ENTER> key will re-execute that command line and it may again be recalled if desired.

The recalled command line may also be EDITED, if desired, before re-executing. FlexiKey allows characters to be INSERTED into a recalled line using the SHIFT-UP arrow to begin insertion. After inserting the desired characters, the SHIFT-RIGHT ARROW will recall the remainder of the command line. Characters may also be DELETED using the DOWN ARROW KEY.

In order to continue to edit a command line without actually executing it, the SHIFT-DOWN arrow may be used to place the cursor at the beginning of the command line WITHOUT EXECUTING IT. Then, further insertions or deletions may be performed.

FLEXIKEY will recall up to 250 characters for editing or re-execution and will also allow a longer "Previous" line to be recalled, explained as follows:

(This was typed first:)
`FOR X = 0 TO 255: PRINT CHR$(X); :NEXT X` <ENTER>

The SHIFT-RIGHT arrow will RECALL the ENTIRE LINE for RE-EXECUTION.

Next, you typed this: `PRINT"HELLO` <ENTER>

The SHIFT-RIGHT arrow will recall only the command `PRINT"HELLO` . However, if you type just PART of the OLD

COMMAND: RENAME

SYNTAX and USAGE:

`RENAME "`*`old filename`*`" TO "`*`new filename`*`"`
`RENAME DRIVE `*`n`*`,"`*`string`*`"`

In addition to the usual function of the RENAME command, which is to allow you to CHANGE the name of an existing filename, the command also allows you to write a "Heading" or "Disk Label" on each Directory. This is also sometimes referred to as a "Volume Label". For example, to write the words "BINARY GAMES DISK #2" on the Directory of Drive number 51, the command RENAME DRIVE would be used as follows:

`RENAME DRIVE 51,"BINARY GAMES DISK #2"` <ENTER>

That command writes the STRING "BINARY GAMES DISK #2" on the Directory of Drive number 51. Both the DRIVE NUMBER and STRING may be BASIC VARIABLES, as below:

`X=51 : A$="BINARY GAMES DISK #2" : RENAME DRIVE X,A$`

The STRING may include ANY characters, including graphics characters and may be up to 255 characters long. As an example of the flexibility of this command,"
following may be done:

`RENAME DRIVE X,CHR$(175)+"DISK NAME HERE"+CHR$(175)` <ENTER>

... will be presented. To begin erasing that "disk", the "Y" (for "YES") key must be pressed. This is done for two reasons: (1) It allows you to be SURE that you are erasing a HARD DISK "disk" and not a FLOPPY DISK and (2) It allows you the option to cancel the DSKINI and NOT erase the "disk". This is important if, for example, you wanted to FORMAT a FLOPPY DISK in Drive Zero, but FORGOT to type `DRIVE OFF` <ENTER> to ENABLE the Floppy Disk Drives. The prompt would immediately alert you that you were really about to erase the HARD DRIVE'S "disk" Zero and give you the chance to cancel the request. If you are formatting a FLOPPY DISK, the warning is NOT given and the DSKINI proceeds as usual.

The process of erasing a Hard Disk "disk" is subject to the VERIFY command. If `VERIFY ON` is used, the erased Hard "disk" will also be verified during erasure) insuring you that there are no data errors on the disk.

Finally, the DSKINI command has been fixed so that it no longer erases the BASIC program in memory. Therefore, the DSKINI command may be PART OF a BASIC program! It is now possible to format disks (floppy OR hard) totally under BASIC program control. Since the DSKINI command requires some free memory to operate, a very large program with many variables may cause the DSKINI command to return an `?OM ERROR` (out of memory error). If this happens, shorten the BASIC program or reserve less variable storage space (CLEAR and PCLEAR).

COMMAND PAST the end of the FIRST COMMAND, the longer command will be recalled. You type this:

```
FOR X = 0 TO        (then press SHIFT-RIGHT)
FOR X = 0 TO 255: PRINT CHR$(X); :NEXT X ....
```

will be recalled!

Finally, some of the ARROW keys which FlexiKey uses previously created printable characters such as Square brackets, etc... These characters may be generated by pressing SHIFT-CLEAR first to temporarily "Turn off" FlexiKey. Then the "Special" characters may be generated. FLEXIKEY automatically turns itself back "ON" after each special key is pressed.

FlexiKey does NOT function during BASIC EDIT procedures, using the EDIT command. EDIT still performs as usual. In fact, FLEXIKEY will recall the EDIT command line, NOT the work performed during EDIT. Example: You type: `EDIT 1230` <ENTER>

Line 1230 comes up for editing, and you edit it, finally pressing <ENTER>

Pressing SHIFT-RIGHT will recall: `EDIT 1230`

This is especially handy when editing many lines of a BASIC program, as it is easier to use FlexiKey then to type the entire command: `EDIT XXXX`

7. Commands Reference

COPY *"filename.ext:drv*" TO *n*
Allows copying of a file without the need to specify destination "filename.ext:drv The destination filename may be included if it is desired that the copied file have a DIFFERENT filename. Files which already exist may be over-written if desired.

DIR *n*
Displays DIRECTORY of drive number. Directory is now displayed with DISKNAME, CONTENTS, DRIVE NUMBER and FREE GRANULES.

DOS *n*
The same as the usual DOS command, but an optional drive number "n" may be specified. If a valid OS-9 boot is not found, the DOS will attempt to find and run AUTOEXEC from drive number *n*.

DRIVE ON (default setting)
Drive numbers 0 to 3 access the hard disk.

DRIVE OFF or DRIVE OFF *n*
Drive numbers 0 to 3 access the FLOPPIES. If the optional Drive Number (n) is used, then only Drives 0 to *n* will access the Floppy Disk Drives.

DRIVE RETORE or DRIVE RESTORE *n*
Causes the Hard Disk Drive to re-zero and re-calibrate itself. If the optional DRIVE NUMBER (n) is specified, then DRIVE

COMMAND: DSKINI

SYNTAX and USAGE:

DSKINI *n*

The DSKINI command is used to format and erase disks. However, the operation of "DSKINI" is somewhat different when applied to the hard drive. The command DSKINI *n* <ENTER> will ERASE the specified "Drive" on the hard drive. The hard disk is NOT "formatted" with this command, rather just erased. This is in contrast to the the floppy disks which are FORMATTED by this command. The Floppy Disk FORMATTING process does several things:

- It divides each floppy disk into 35 TRACKS
- It divides each floppy disk TRACK into 18 SECTORS
- It LABELS each SECTOR with an ADDRESS MARKER so that the system "knows" which Track and Sector it is on.
- It fills each SECTOR with CHR$(255) characters (erases each sector).

The hard drive comes from the factory "Pre-Formatted". That is, the process of defining tracks and sectors has already been done. The DSKINI command when applied to a hard drive "disk" number simply ERASES that "disk" so that it may be used again. All of the other "disks" on the hard drive are NOT erased.

Finally, when the DSKINI command is applied to the hard drive, the prompt:

ERASE HARD DRIVE *n*
ARE YOU SURE? (Y/N)

This command will cause the hard drive to re-calibrate itself and restore the read/write heads to Track zero. The number 0 – *n* refers to the ID of the drive attached to the controller (See the DRIVE SET command). The number 0 - *n* is optional, and if omitted, the default hard disk drive will be restored. This command also internally does a "DRIVE #" command, so typing `DRIVE RESTORE 1` will recalibrate SCSI device 1 (IDE slave), but will also leave the system setup on drive number 1. To go back to drive zero, the command `DRIVE #0` or `DRIVE RESTORE 0` may be used. Typing `DRIVE RESTORE` will CLOSE all open files.

a will be restored. Legal values of *n* are 0 thru 7 for SCSI and 0 thru 1 for IDE.

`DRIVE #`*n*

Allows setting the device to be used. Legal values for *n* are 0 thru 7 for SCSI and 0 thru 1 for IDE.

`DRIVE STOP` or `DRIVE STOP` *n*

Parks the Hard Drive in the shipping zone.Use this command BEFORE powering off. If the optional DRIVE NUMBER (n) is specified, then DRIVE a will be parked. Legal values of *n* are 0 thru 7 for SCSI and 0 thru 1 for IDE.

`RENAME DRIVE` *n*`,"`*string*`"`

Writes a DISKNAME on the specified drive. May also be used with variables such as: `RENAME DRIVE X,A$` ... where X is the Drive Number and A$ is the Disk Name.

`RUNM"`*filename.ext:drv*`",`*offset*

LOADM's and EXEC's a BINARY file. If the optional offset is used, the program will be loaded offset bytes higher than usual. As always, offset can "wrap around" zero thereby allowing negative offset loading.

8. Detailed Command Summary

COMMAND: BACKUP

SYNTAX and USAGE:

`BACKUP source TO destination`

The usage and syntax of the BACKUP command has not been changed. However, the machine language code has been fixed so that the BACKUP command NO LONGER erases the BASIC program in memory! Therefore, it is now possible to include the BACKUP command WITHIN a BASIC program successfully. This feature, in combination with the repaired DSKINI command, allows TOTALLY AUTOMATED disk formatting and copying, all under the control of a BASIC program!

This feature is perfect for mass producing disks, club and business use, or just simplifying your own Hard Disk backup program.

Since the BACKUP command now only uses FREE memory, the copy buffer size used may be smaller than it was before if a large Basic program is in memory, or if a great deal of variable storage space is allocated. This means that IF (and only if) you are making a SINGLE DRIVE backup, more disk swaps per backup would be required, since less is copied on each "pass". It is suggested (but NOT necessary) that a PCLEAR 1 statement be used prior to making a single drive backup. This will minimize the number of disk swaps

floppy disk drive ZERO, and all the rest (1 thru MAX) hard drive. The command `DRIVE OFF 0` <ENTER> would accomplish this. Typing `DRIVE OFF` will CLOSE all open files.

NOTE: Drive numbers FOUR to MAXIMUM will ALWAYS access the Hard Drive, regardless of the access specified by using DRIVE ON or DRIVE OFF.

`DRIVE STOP` <ENTER> or `DRIVE STOP 0` *thru* `n` <ENTER>

This command will PARK the hard drive in the "Safe Landing Zone" and may be used every time the hard drive is to be turned off. If no DRIVE NUMBER is specified after the DRIVE STOP command, the system will Park the Drive LAST SPECIFIED with the `DRIVE #` command. Attempting to access the Hard Drive AFTER it has been parked may result in an error since the read/write heads are now in a non-data zone. The drive will AUTOMATICALLY re-calibrate the next time power is switched on. Typing `DRIVE STOP` will CLOSE all open files.

`DRIVE #0 thru n` <ENTER>

This command allows setting the SCSI or IDE bus channel. Normally, the system runs on SCSI channel zero or IDE master (DRIVE #0 command), but the channel may be changed, if desired, to access other devices on the bus. Attempting to select a non-active channel will result in an `?IO ERROR`. Attempting to select an illegal channel outside the range will result in an `?FC ERROR`. To restore the system to normal, use `DRIVE #0` <ENTER> Typing `DRIVE #n` will CLOSE all open files.

`DRIVE RESTORE 0 thru n` <ENTER>

COMMAND: DRIVE

SYNTAX and USAGE:

```
DRIVE n
DRIVE ON
DRIVE OFF
DRIVE OFF 0 thru 3
DRIVE STOP
DRIVE STOP 0 thru n
DRIVE #0 thru n
DRIVE RESTORE
DRIVE RESTORE 0 thru n
```

`DRIVE n` <ENTER>

Changes the drive DEFAULT to the drive number you specify. If you do not use the DRIVE command, the system will default to and use drive number ZERO.

`DRIVE ON` <ENER>

Allows you to access Drive numbers ZERO through THREE from the HARD DRIVE. This is the default setting. That is, the system will automatically access the HARD DRIVE upon power-up or cold reset. Typing `DRIVE ON` will CLOSE all open files.

`DRIVE OFF` <ENTER> or `DRIVE OFF 0 thru 3` <ENTER>

Allows you to access Drive numbers ZERO through THREE from the FLOPPY DISK DRIVE(S). If a NUMBER is included with this command, then ONLY Floppy Drive(s) ZERO to NUMBER will be enabled. Example: You wish to use ONLY

necessary. On multiple drive or hard drive backups, this step is not needed since the system will simply switch back and forth as many times as it needs to. If, for example, a PCLEAR 8 statement were used prior to backup, very little memory would be available to the BACKUP function and it might require 50 or more disk swaps to copy a single disk! With multiple disk drives and/or a hard drive, the drive switching is automatic, and of no concern to the user.

COMMAND: COPY

SYNTAX and USAGE:

```
COPY          "filename/ext:drive"          TO
                "filename/ext:drive"
COPY "filename/ext:drive" TO n
```

The COPY command has been improved. You may use the COPY command as usual, that is, to COPY a SOURCE Filename to a DESTINATION filename. Usually, the Source and Destination Filenames are the same. Therefore, the COPY command required you to type the SAME Filename TWICE! This is inefficient and time consuming.

With HDB-DOS, it is now possible to copy a file to another "Drive" by simply specifying the SOURCE FILENAME and the DESTINATION DRIVE NUMBER.

The entire Destination Filename MAY be used, if desired, but it is no longer NECESSARY. The Destination Filename is usually specified ONLY IF it is desired that the copied file have a DIFFERENT filename than the original.

HDB-DOS will also allow you to COPY over an existing file. You no longer are faced with an `?AE ERROR` (File Already Exists) error. If the Destination Filename is already in use, HDB-DOS will prompt you with the message:

```
FILE ALREADY EXISTS OVERWRITE IT? (Y/N)
```

If you wish to replace the Old file with the New (If you wish to OVERWRITE the file), simply type "Y" for "YES" and the old file will be overwritten. Striking any other key will END the COPY function and NOT overwrite or copy anything.

NOTE: A special OS-9 type boot track is NOT required to use this feature. Since no boot track will be found, the DOS command will next try to run AUTOEXEC on that disk, thereby accomplishing the desired action.

The following may help in understanding the actions of the new DOS command:

Type: `DOS 35` <ENTER>
System looks for OS-9 boot on drive 35
Was OS-9 boot found?

If yes, go run OS-9 from drive 35 (done)
Else, see if AUTOEXEC.BAS is on drive 35

Was AUTOEXEC.BAS found on drive 35?

If yes, go run AUTOEXEC.BAS on drive 35 (done)
Else, go to Basic's OK prompt. (done)

The end result of all this? `DOS 35` accomplishes the SAME thing as if you typed this command: `RUN"AUTOEXEC.BAS:35"` Which is easier?

COMMAND: DOS

SYNTAX and USAGE:

`DOS n` (*n* is optional)

The DOS command is normally used to start up an OS-9 floppy boot disk and run the OS-9 Operating System. This command still performs the same function as before, but with several improvements.

The new DOS command now allows the use of an optional drive number argument. For example, if you wished to start up an OS-9 disk which was backed up to hard "drive" number 35, the command `DOS 35` <ENTER> would be used. Note that the command DOS used alone ALWAYS accesses drive zero, regardless of the default drive setting, thereby maintaining full compatibility with the original DOS command.

An additional feature of the new DOS command is that IF a valid OS-9 boot disk OR OS-9 type utility is NOT found on the requested disk, the system NEXT tries to run AUTOEXEC.BAS, if present, on the requested disk. This feature allows you to have MANY AUTOEXEC files on different disks, all starting up their OWN application or program. Then, to run AUTOEXEC on ANY disk, just type `DOS n`, where "n" is the drive number that contains the program(s) you wish to run.

This prompt is ONLY given when COPY is used in the "DIRECT" (keyboard) mode. If the COPY command is used WITHIN a BASIC program, the already-existing file will be automatically replaced (overwritten) WITHOUT any warning. Also, any attempt to COPY a program to ITSELF on the SAME DRIVE is pointless and will result in an `?AE ERROR` (File Already Exists) error.

This is an example of the usage of the improved COPY command:

`COPY"GAME/BIN:51" TO 92` <ENTER>

That command will COPY the program called "GAME/BIN" on "Drive" 51 to the filename "GAME/BIN" on "Drive" 92. The following example will do the same exactly the same thing:

`COPY"GAME/BIN:51" TO "GAME/BIN:92"` <ENTER>

The Filename(s) and Destination "Drive" numbers may also be BASIC variables if desired. The following command example explains this:

`A$="GAME/B1N": FOR X=0 TO 10 : COPY A$ TO X : NEXT X` <ENTER>

This command would make a copy of the program "GAME/BIN" on each "Drive" from Drive Zero to Drive Ten.

COMMAND: DIR

SYNTAX and USAGE:

```
DIR
```

Displays a directory of the drive number you specify. If you omit the drive number, the system will use Drive Zero (unless you use the DRIVE command to change this default.) The Directory will be displayed with the DISKNAME (see the RENAME DRIVE command), the FILENAME(S) SAVED, the DRIVE NUMBER and the FREE GRANULES on that drive. Below is a typical listing:

```
DIR 51 <ENTER>

BINARY GAMES #2
COPTER.BAS    3  CLOWNS.BIN    4
DEVIL.BIN     4  DISKUTIL.BIN  4
ELECTRON.BIN  6  GALACTIC.BIN  4
GALAXY3.BIN   4  INVASION.BIN  5
PROTECT.BIN   10 PROTECT2.BIN  8
ROBOTRON.BIN  3  SHOOTING.BIN  4
SPACEINV.BIN  2  TEMPEST.BIN   3
WHYBIRD.BIN   4
DRIVE=51 FREE=1
```

The Directory is displayed in a 2 column format on the Color Computer 1, 2 and 3 in the 32 column mode and 40 column mode of the Color Computer 3. The 80 column mode of the Color Computer 3 will display the Directory with 5 columns:

```
DIR 51 <ENTER>

BINARY GAMES #2
COPTER.BAS   3 CLOWNS.BIN   4 DEVIL.BIN     4 DISKUTIL.BIN  4 ELECTRON.BIN  6
GALACTIC.BIN 4 GALAXY3.BIN  4 INVASION-BIN  5 PROTECT.BIN  10 PROTECT2.BIN  7
ROBOTRON.BIN 3 SHOOTING.BIN 4 SPACEINV.BIN  2 TEMPEST.BIN   3 WHYBIRD.BIN   4
DRIVE=51 FREE=1
```


The digit(s) following each filename represent the number of GRANULES each file consumes. The File type and Storage Format characters that were normally present have been eliminated from the display. The Filetype and Format information bytes are STILL PRESENT in the Directory, they simply are not DISPLAYED. That information may still be accessed through BASIC or MACHINE LANGUAGE as usual.