Using Large Hard Drives With HDB-DOS

Robert Gault

HDB-DOS, and its predecessor, RGB-DOS have some interesting potentials that were never exploited by Roger Krupski and Ken-Ton. The original documentation for HDB-DOS seemed to be following in the same footsteps, so I wrote this tutorial to display the full potential of these OS.

In normal usage, with the DOS code in ROM (EPROM) changes are not possible. Whatever is burnt into the ROM is what you get. In particular, this means that the hard drive offset is fixed. RGB-DOS was issued before the CoCo 3 and it was not envisioned that the user would switch into the all RAM mode on a CoCo 1. The advent of the CoCo 3, which is always in RAM mode, changed everything but Ken-Ton did not take advantage of it.

If you consider the DOS unchangeable then you must use the same offset for all hard drives attached to the system. But if you can change the offset value, what benefits do you receive?

The examples that follow present impossible problems for our DOS (RGB-DOS/HDB-DOS) unless the offset values are variable instead of fixed.

Example 1) – Multiple hard drives of different sizes on the same system. Each drive is partitioned for both OS-9 and Disk BASIC. Each drive has allocated 256 disks for Disk BASIC. What should the offset values be for each drive? It should be easy to see that if the offsets were identical then the larger the drive the more wasted space.

Example 2) – Multiple hard drives on the same system where it is desired to use some drives only for OS-9 and others only for Disk BASIC. What do you do with the offset values?

Example 3) – One very large drive where it is desired to just use Disk BASIC. Drives now far exceed the 256 disk limit. Should the extra space be wasted?

DOS Offset

Address	Usage
$D938	MSB (most significant byte) of OS-9 offset
$D939 - $D93A	LSW (least significant word) of OS-9 offset

The OS-9 offset indicates the number of sectors at the beginning of the drive that are not used by Disk BASIC. In an all Disk BASIC system, these three bytes are $000000. However, it is simple to change these values with the POKE command and if this is done the starting point on the drive for Disk BASIC will change.

The OS will at turn-on automatically run AUTOEXEC.BAS on disk 0 if the program is present. AUTOEXEC.BAS can be used to POKE new offset values into memory and then call the DOS

command. DOS will first look for an OS-9 boot track and if not found, then look for and run AUTOEXEC.BAS. Since the offset value was changed, a new AUTOEXEC.BAS will be run that is different from the first.

For many years I used an 80 meg and an 11 meg drive on the same system. These drives had different offsets and different numbers of Disk BASIC disks. Each drive had it's own AUTOEXEC.BAS which poked in the correct offset value for the drive in use and new values when switching drives.

With the advent of gigabyte drives, there is a new use for variable offsets. The OS can address 256 separate disks on a single drive. Each disk has 35 tracks of 18 sectors for a total of 630 sectors. The sectors are assumed to be 256 bytes but if larger, only the first 256 bytes in a sector will be used. Using a 20 GB drive as an example formatted with 512 byte sectors, there will be about 39,062,500 sectors or about 242 partitions of 256 disks; 630x256x242 = 39,029,760 sectors. This size drive is now very common and even with the bloatware of a Microsoft OS installed, most of the drive will be free for an emulated Coco. If this is the smallest drive available for use with HDB-DOS and a TC^3 controller, 99.6% would be wasted using a fixed offset.

Here are the offsets that would be used to access multiple partitions of 256 disks on a large hard drive.

Partition	Offset	Drive #s in the partition
0	$000000	0-255
1	$027600	256-511
2	$04EC00	512-767

And so on…

partition# x $27600 = offset value (partition# +1) x 256 = number of available disks

The OS as it is currently written will address each partition with drive numbers 0-255. However, since this OS permits naming each disk, one could add the true drive number to the disk name. The BASIC program that follows, if installed as AUTOEXEC.BAS on each partition disk #0, will enable stepping through the partitions, POKE the correct offset values, and indicate what the real drive numbers are.

```
10 ONERRGOTO230
20 DRIVEON:FP$="OFF"
30 'DEMO OF MULTIPLE DRIVE USE
40 SIZE=2^32:' THIS IS A MAXIMUM FOR STOCK HDB-DOS
50 DR=256:SC=DR*35*18:' $27600 SECTORS
60 RGB:WIDTH80:HZ=25:VT=0:MS$="BOZO'S Hard Drive System":GOSUB240
70 M=&HD938:LSN=PEEK(M)*65536+PEEK(M+1)*256+PEEK(M+2)
80 HZ=25:VT=2:MS$="Available drives ":GOSUB240:PRINT INT(SIZE/256/630);
90 PRINTTAB(5)"Floppies ";FP$
100 GOSUB250
110 HZ=20:VT=4:MS$="This is block ":GOSUB240:PRINTBLOCK;", drives ";MINDR;"
TO";MAXDR
120 LOCATE0,5:PRINT"1) Select drive block"
130 PRINT"2) Floppies ON
140 PRINT"3) Floppies OFF
```

```
150 PRINT"ETC."
160 A$=INKEY$:A=VAL(A$):IFA>0 AND A<4 THEN170 ELSE160
170 ON A GOTO 180,210,220:GOTO170
180 LOCATE30,10:INPUT"DRIVE BLOCK #=";DB:LSN=DB*SC
190 LS(1)=INT(LSN/65536):LS(2)=INT((LSN-LS(1)*65536)/256):LS(3)=LSN-LS(1)*65536-
LS(2)*256
200 POKEM,LS(1):POKEM+1,LS(2):POKEM+2,LS(3):DRIVE0:RUN"AUTOEXEC
210 DRIVEOFF:FP$="OFF":GOTO60
220 DRIVEON:FP$="ON":GOTO60
230 END
240 LOCATEHZ,VT:PRINTMS$;:RETURN
250 BLOCK=INT(LSN/SC):MINDR=BLOCK*DR:MAXDR=MINDR+DR-1:RETURN
```