

Some Notes on LBA

An explanation of LBA (Logical Block Addressing)

LBA is an acronym which stands for Logical Block Addressing.

LBA is a way of addressing hard drives by assigning numbers sequentially to each sector on the drive, starting from 0 for the first sector.

If used by operating systems and applications on the software side of the BIOS as well as by the drive on the hardware side of the BIOS, LBA can be a more efficient way to address hard drives than the CHS (Cylinder, Head, Sector) method of addressing. It may be awhile before this becomes widely used by hardware, BIOSes, and software. In the meantime it isn't necessarily a performance increase to use LBA and may even be less efficient. There is widespread confusion throughout the industry on the subject of LBA and BIOS support for large drives - many people mistakenly think these to be the same thing. Most drives that support LBA do not require that LBA is used and LBA addressing is not normally required in order for a BIOS to support large drives. What is required to support large drives is a translating BIOS.

There are two widely used translation algorithms used in PC BIOSes to support large drives. One kind is called Extended CHS (ECHS, also called CHS to CHS and sometimes called "bit-shift"). In some BIOSes, this translation type is referred to as "Large". This type of translation works by halving the cylinder count and doubling the head count until the cylinder count is 1024 or less. The other main type of translation used is called LBA Assist or Assisted LBA, often just called LBA.

LBA addressing has always been used on SCSI drives. IDE drives began to use LBA addressing as an option at about the same time that large (over 504MB) IDE drives began to appear. Most large IDE (or EIDE) drives support LBA mode. Normally, LBA mode is optional on drives that support it. To use LBA addressing it must be implemented in the BIOS as well as by the hard drive.

Phoenix BIOSes began to support LBA addressing with version 4.03, the same time that the Phoenix BIOS began to support large hard drives. These BIOSes normally used ECHS translation and did not use LBA-Assist translation.

In order for LBA mode to be used on the software side of the BIOS, the operating system or application must support the new "INT13 extensions" and the BIOS must also support INT13 extensions.

The Phoenix BIOS began to support INT13 extensions with version 4.04.

There is a significant difference in the way that Phoenix BIOSes handle LBA addressing between version 4.03 and version 4.05. In 4.03 code, if "LBA mode" is enabled in the BIOS on a drive that supports it, then LBA addressing will be used on all accesses to the drive. In 4.05 code, if LBA is enabled in the BIOS on a drive that supports it, it will only use LBA on drive accesses if the operating system or application making the call is using INT13 extended BIOS calls. In other words LBA will only be used if it is used all the way through. This makes sense because if a call is made to the drive in CHS form and the BIOS accesses the drive in LBA mode, then an extra translation step is required by the BIOS to convert the CHS address to LBA.

When we began to add support for hard drives over 8.4GB to our BIOS upgrades based on Phoenix 4.05 code, we modified them so that if "LBA mode" is enabled on in CMOS for a particular drive, the BIOS will always use LBA addressing to access the drive.

There is a utility that Western Digital makes available on their Web site, <http://www.wdc.com>, called WDTBLCHK.EXE, which is used to determine whether a BIOS can support large drives. It also returns a lot of other info on the drive and the BIOS. We had a customer report that, according to

Some Notes on LBA

An explanation of LBA (Logical Block Addressing)

WDTBLCHK, LBA was not being used on his drive even though it was enabled in the BIOS. This was a 4.05 BIOS which does not use LBA unless it is used from the software on through. So this is the result we would expect unless a call is made to the drive in LBA form. We have tested this to verify that the BIOS is working as it should. At this time there is not a lot of support for INT13 extensions in operating systems or applications. WIN95 does have support for INT13 extensions. Apparently WIN95 will only make LBA calls to a drive if WIN95 FDISK has been used to partition the drive and WIN95 FDISK has assigned a new partition type (type 0E or type 0F) to the drive. See our text file W95PARTN.TXT for a discussion of some problems associated with the new partition types used by WIN95. This file is on our web site at this URL - <http://www.firmware.com/support/bios/w95partn.htm>
In some BIOSes there may be a danger of data loss if LBA is enabled or disabled after the drive has already been partitioned. In the Phoenix BIOS there should be no harm in enabling or disabling LBA at any time.

Here is a DEBUG script that can be used to determine whether LBA was used during the last access to a hard drive.

Run this after performing read or write to hard drive:

```
C:>DEBUG  
  
- I 1F6 (replace 1F6 with 176 for drive on secondary port)  
  
  xx    (returns 2-digit hex value)  
  
- Q    (quits to DOS)  
  
Ax or Bx = LBA not used in last operation  
Ex or Fx = LBA was used in last operation
```

Note - this DEBUG script may not act as expected if there is more than one drive in the system. If DEBUG is loaded from a hard drive, this script can test only that drive. It may be preferable to copy DEBUG to a floppy disk and run this test from there.