CHS Translation Algorithms

by **Hale Landis** (landis@sugs.tware.com)

Definitions
---

* 528MB - The maximun drive capacity that is supported by 1024 cylinders, 16 heads and 63 sectors (1024x16x63x512). This is the limit for CHS addressing in the original IBM PC/XT and IBM PC/AT INT 13H BIOS.

* 8GB - The maximum drive capacity that can be supported by 1024 cylinders, 256 heads and 63 sectors (1024x256x63x512). This is the limit for the BIOS INT 13H AH=0xH calls.

* ATA - AT Attachment -- The real name of what is widely known as IDE.

* CE Cylinder - Customer Engineering cylinder. This is the last cylinder in P-CHS mode. IBM has always reserved this cylinder for use of disk diagnostic programs. Many BIOS do not account for it correctly. It is of questionable value these days and probably should be considered obsolete. However, since there is no industry wide agreement, beware. There is no CE Cylinder reserved in the L-CHS address. Also beware of diagnostic programs that don't realize they are operating in L-CHS mode and think that the last L-CHS cylinder is the CE Cylinder.

* CHS - Cylinder/Head/Sector. This is the traditional way to address sectors on a disk. There are at least two types of CHS addressing: the CHS that is used at the INT 13H interface and the CHS that is used at the ATA device interface. In the MFM/RLL/ESDI and early ATA days the CHS used at the INT 13H interface was the same as the CHS used at the device interface.

  Today we have CHS translating BIOS types that can use one CHS at the INT 13H interface and a different CHS at the device interface. These two types of CHS will be called the logical CHS or L-CHS and the physical CHS or P-CHS in this document.

L-CHS is the CHS used at the INT 13H interface and P-CHS is the CHS used at the device interface.

The L-CHS used at the INT 13 interface allows up to 256 heads, up to 1024 cylinders and up to 63 sectors. This allows support of up to 8GB drives. This scheme started with either ESDI or SCSI adapters many years ago.

The P-CHS used at the device interface allows up to 16 heads up to 65535 cylinders, and up to 63 sectors. This allows access to 2^28 sectors (136GB) on an ATA device. When a P-CHS is used at the INT 13H interface it is limited to 1024 cylinders, 16 heads and 63 sectors. This is where the old 528MB limit originated.

ATA devices may also support LBA at the device interface. LBA allows access to approximately 2^28 sectors (137GB) on an ATA device.

A SCSI host adapter can convert a L-CHS directly to an LBA used in the SCSI read/write commands. On a PC today, SCSI is also limited to 8GB when CHS addressing is used at the INT 13H interface.

* EDPT - Enhanced fixed Disk Parameter Table -- This table returns additional information for BIOS drive numbers 80H and 81H. The EDPT for BIOS drive 80H is pointed to by INT 41H. The EDPT for BIOS drive 81H is pointed to by INT 46H. The EDPT is a fixed disk parameter table with an AxH signature byte. This table format returns two sets of CHS information. One set is the L-CHS and is probably the same as returned by INT 13H AH=08H. The other set is the P-CHS used at the drive interface. This type of table allows drives with >1024 cylinders or drives >528MB to be supported. The translated CHS will have <=1024 cylinders and (probably) >16 heads. The CHS used at the drive interface will have >1024 cylinders and <=16 heads. It is unclear how the IBM defined CE cylinder is accounted for in such a table. Compaq probably gets the credit for the original definition of this type of table.

* FDPT - Fixed Disk Parameter Table - This table returns additional information for BIOS drive numbers 80H and 81H. The FDPT for BIOS drive 80H is pointed to by INT 41H. The FDPT for BIOS drive 81H is pointed to by INT 46H. A FDPT does not have a AxH signature byte. This table format returns a single set of CHS information. The L-CHS information returned by this table is probably the same as the P-CHS and is also probably the same as the L-CHS returned by INT 13H AH=08H. However, not all BIOS properly account for the IBM defined CE cylinder and this can cause a one or two cylinder difference between the number of cylinders returned in the AH=08H data and the FDPT data. IBM gets the credit for the original definition of this type of table.

* LBA - Logical Block Address. Another way of addressing sectors that uses a simple numbering scheme starting with zero as the address of the first sector on a device. The ATA standard requires that cylinder 0, head 0, sector 1 address the same sector as addressed by LBA 0. LBA addressing can be used at the ATA interface if the ATA device supports it. LBA addressing is also used at the INT 13H interface by the AH=4xH read/write calls.

* L-CHS -- Logical CHS. The CHS used at the INT 13H interface by the AH=0xH calls. See CHS above.

* MBR - Master Boot Record (also known as a partition table) - The sector located at cylinder 0 head 0 sector 1 (or LBA 0). This sector is created by an "FDISK" utility program. The MBR may be the only partition table sector or the MBR can be the first of multiple partition table sectors that form a linked list. A partition table entry can describe the starting and ending sector addresses of a partition (also known as a logical volume or a logical drive) in both L-CHS and LBA form. Partition table entries use the L-CHS returned by INT 13H AH=08H. Older FDISK programs may not compute valid LBA values.

* OS - Operating System.

* P-CHS -- Physical CHS. The CHS used at the ATA device interface. This CHS is also used at the INT 13H interface by older BIOS's that do not support >1024 cylinders or >528MB. See CHS above.

Background and Assumptions

First, please note that this is written with the OS implementor in mind and that I am talking about the possible BIOS types as seen by an OS during its hardware configuration search.

It is very important that you not be confused by all the misinformation going around these days. All OS's that want to be co-resident with another OS (and that is all of the PC based OS's that I know of) MUST use INT 13H to determine the capacity of a hard disk. And that capacity information MUST be determined in L-CHS mode. Why is this? Because: 1) FDISK and the partition tables are really L-CHS based, and 2) MS/PC DOS uses INT 13H AH=02H and AH=03H to read and write the disk and these BIOS calls are L-CHS based. The boot processing done by the BIOS is all L-CHS based. During the boot processing, all of the disk read accesses are done in L-CHS mode via INT 13H and this includes loading the first of the OS's kernel code or boot manager's code.

Second, because there can be multiple BIOS types in any one

system, each drive may be under the control of a different type of BIOS. For example, drive 80H (the first hard drive) could be controlled by the original system BIOS, drive 81H (the second drive) could be controlled by a option ROM BIOS and drive 82H (the third drive) could be controlled by a software driver. Also, be aware that each drive could be a different type, for example, drive 80H could be an MFM drive, drive 81H could be an ATA drive, drive 82H could be a SCSI drive.

Third, not all OS's understand or use BIOS drive numbers greater than 81H. Even if there is INT 13H support for drives 82H or greater, the OS may not use that support.

Fourth, the BIOS INT 13H configuration calls are:

* AH=08H, Get Drive Parameters -- This call is restricted to drives up to 528MB without CHS translation and to drives up to 8GB with CHS translation. For older BIOS with no support for >1024 cylinders or >528MB, this call returns the same CHS as is used at the ATA interface (the P-CHS). For newer BIOS's that do support >1024 cylinders or >528MB, this call returns a translated CHS (the L-CHS). The CHS returned by this call is used by FDISK to build partition records.

* AH=41H, Get BIOS Extensions Support -- This call is used to determine if the IBM/Microsoft Extensions or if the Phoenix Enhanced INT 13H calls are supported for the BIOS drive number.

* AH=48H, Extended Get Drive Parameters -- This call is used to determine the CHS geometries, LBA information and other data about the BIOS drive number.

* the FDPT or EDPT -- While not actually a call, but instead a data area, the FDPT or EDPT can return additional information about a drive.

* other tables -- The IBM/Microsoft extensions provide a pointer to a drive parameter table via INT 13H AH=48H. The Phoenix enhancement provides a pointer to a drive parameter table extension via INT 13H AH=48H. These tables are NOT the same as the FDPT or EDPT.

Note: The INT 13H AH=4xH calls duplicate the older AH=0xH calls but use a different parameter passing structure. This new structure allows support of drives with up to 2^64 sectors (really BIG drives). While at the INT 13H interface the AH=4xH calls are LBA based, these calls do NOT require that the drive support LBA addressing.

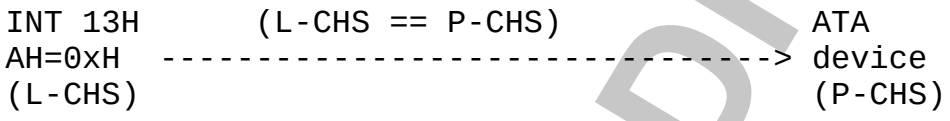
CHS Translation Algorithms

NOTE: Before you send me email about this, read this entire
  section. Thanks!

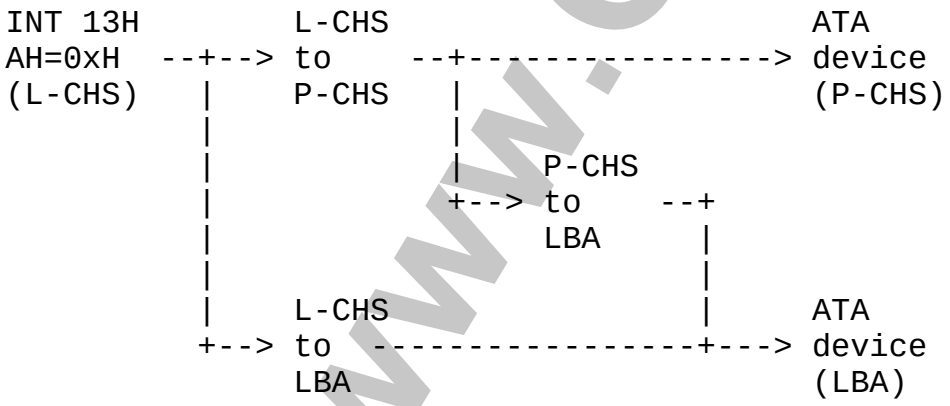
As you read this, don't forget that all of the boot processing done by the system BIOS via INT 19H and INT 13H use only the INT 13H AH=0xH calls and that all of this processing is done in CHS mode.

First, lets review all the different ways a BIOS can be called to perform read/write operations and the conversions that a BIOS must support.

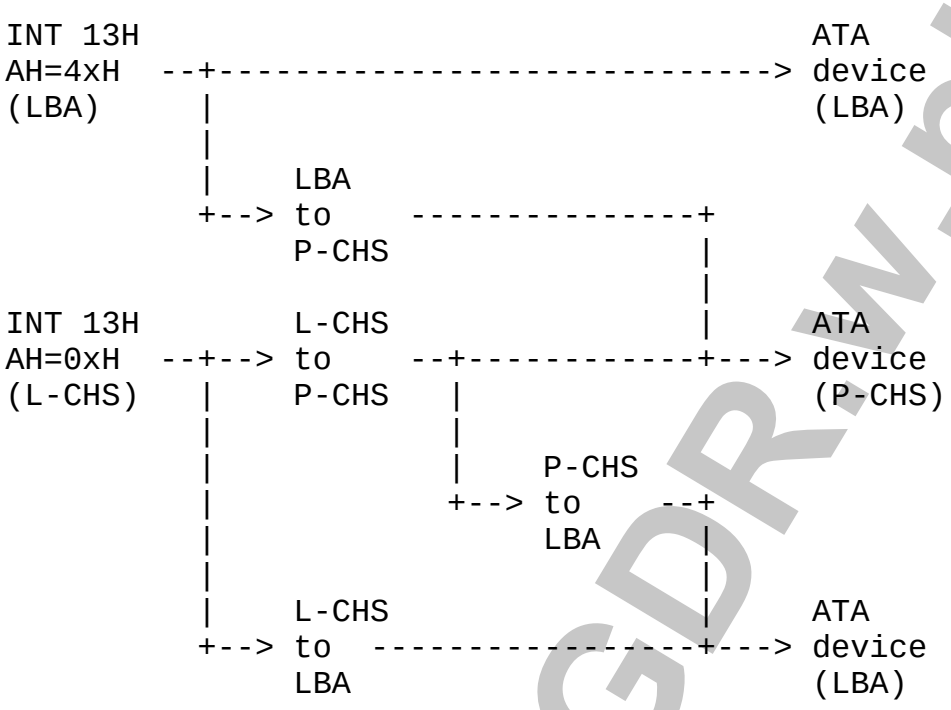
! * An old BIOS (like BIOS type 1 below) does no CHS translation
   and does not use LBA. It only supports the AH=0xH calls:



* A newer BIOS may support CHS translation and it may support
   LBA at the ATA interface:



* A really new BIOS may also support the AH=4xH in addtion to the older AH\0xH calls. This BIOS must support all possible combinations of CHS and LBA at both the INT 13H and ATA interfaces:



You would think there is only one L-CHS to P-CHS translation algorithm, only one L-CHS to LBA translation algorithm and only one P-CHS to LBA translation algorithm. But this is not so. Why? Because there is no document that standardizes such an algorithm. You can not rely on all BIOS's and OS's to do these translations the same way.

The following explains what is widely accepted as the "correct" algorithms.

An ATA disk must implement both CHS and LBA addressing and must at any given time support only one P-CHS at the device interface. And, the drive must maintain a strick relationship between the sector addressing in CHS mode and LBA mode. Quoting the ATA-2 document:

```
LBA = ( (cylinder * heads_per_cylinder + heads )
        * sectors_per_track ) + sector - 1
```

where heads_per_cylinder and sectors_per_track are the current translation mode values.

This algorithm can also be used by a BIOS or an OS to convert a L-CHS to an LBA as we'll see below.

This algorithm can be reversed such that an LBA can be converted to a CHS:

```
    cylinder = LBA / (heads_per_cylinder * sectors_per_track)
        temp = LBA % (heads_per_cylinder * sectors_per_track)
        head = temp / sectors_per_track
      sector = temp % sectors_per_track + 1
```

While most OS's compute disk addresses in an LBA scheme, an OS like DOS must convert that LBA to a CHS in order to call INT 13H.

Technically an INT 13H should follow this process when converting an L-CHS to a P-CHS:

1) convert the L-CHS to an LBA,
2) convert the LBA to a P-CHS,

If an LBA is required at the ATA interface, then this third step is needed:

3) convert the P-CHS to an LBA.

All of these conversions are done by normal arithmetic.

However, while this is the technically correct way to do things, certain short cuts can be taken. It is possible to convert an L-CHS directly to a P-CHS using bit a bit shifting algorithm. This combines steps 1 and 2. And, if the ATA device being used supports LBA, steps 2 and 3 are not needed. The LBA value produced in step 1 is the same as the LBA value produced in step 3.

AN EXAMPLE

Lets look at an example. Lets say that the L-CHS is 1000 cylinders 10 heads and 50 sectors, the P-CHS is 2000 cylinders, 5 heads and 50 sectors. Lets say we want to access the sector at L-CHS 2,4,3.

* step 1 converts the L-CHS to an LBA,

```
lba = 1202 = ( ( 2 * 10 + 4 ) * 50 ) + 3 - 1
```

* step 2 converts the LBA to the P-CHS,

```
cylinder =   4 = ( 1202 / ( 5 * 50 )
    temp = 202 = ( 1202 % ( 5 * 50 ) )
    head =   4 = ( 202 / 50 )
  sector =   3 = ( 202 % 50 ) + 1
```

* step 3 converts the P-CHS to an LBA,

```
lba = 1202 = ( ( 4 * 5 + 4 ) * 50 ) + 3 - 1
```

Most BIOS (or OS) software is not going to do all of this to convert an address. Most will use some other algorithm. There are many such algorithms.

BIT SHIFTING INSTEAD

If the L-CHS is produced from the P-CHS by 1) dividing the P-CHS cylinders by N, and 2) multiplying the P-CHS heads by N, where N is 2, 4, 8, ..., then this bit shifting algorithm can be used and N becomes a bit shift value. This is the most common way to make the P-CHS geometry of a >528MB drive fit the INT 13H L-CHS rules. Plus this algorithm maintains the same sector ordering as the more complex algorithm above. Note the following:

```
     Lcylinder = L-CHS cylinder being accessed
         Lhead = L-CHS head being accessed
       Lsector = L-CHS sector being accessed

     Pcylinder = the P-CHS cylinder being accessed
         Phead = the P-CHS head being accessed
       Psector = P-CHS sector being accessed

             NPH = is the number of heads in the P-CHS
               N = 2, 4, 8, ..., the bit shift value
```

The algorithm, which can be implemented using bit shifting instead of multiply and divide operations is:

```
     Pcylinder = ( Lcylinder * N ) + ( Lhead / NPH );
         Phead = ( Lhead % NPH );
       Psector = Lsector;
```

A BIT SHIFTING EXAMPLE

Lets apply this to our example above (L-CHS = 1000,10,50 and P-CHS = 2000, 5, 50) and access the same sector at at L-CHS 2,4,3.

```
     Pcylinder = 4 = ( 2 * 2 ) + ( 4 / 5 )
         Phead = 4 = ( 4 % 5 )
       Psector = 3 = 3
```

As you can see, this produces the same P-CHS as the more complex method above.

SO WHAT IS THE PROBLEM?

The basic problem is that there is no requirement that a CHS translating BIOS followed these rules. There are many other algorithms that can be implemented to perform a similar function. Today, there are at least two popular implementions: the Phoenix

implementation (described above) and the non-Phoenix implementations.

SO WHY IS THIS A PROBLEM IF IT IS HIDDEN INSIDE THE BIOS?

Because a protected mode OS that does not want to use INT 13H must implement the same CHS translation algorithm. If it doesn't, your data gets scrambled.

WHY USE CHS AT ALL?

In the perfect world of tomorrow, maybe only LBA will be used. But today we are faced with the following problems:

* Some drives >528MB don't implement LBA.

* Some drives are optimized for CHS and may have lower performance when given commands in LBA mode. Don't forget that LBA is something new for the ATA disk designers who have worked very hard for many years to optimize CHS address handling. And not all drive designs require the use of LBA internally.

* The L-CHS to LBA conversion is more complex and slower than the bit shifting L-CHS to P-CHS conversion.

* DOS, FDISK and the MBR are still CHS based -- they use the CHS returned by INT 13H AH=08H. Any OS that can be installed on the same disk with DOS must understand CHS addressing.

* The BIOS boot processing and loading of the first OS kernel code is done in CHS mode -- the CHS returned by INT 13H AH=08H is used.

* Microsoft has said that their OS's will not use any disk capacity that can not also be accessed by INT 13H AH=0xH.

These are difficult problems to overcome in today's industry environment. The result: chaos.

DANGER TO YOUR DATA!

See the description of BIOS Type 7 below to understand why a WD EIDE BIOS is so dangerous to your data.

The BIOS Types

I assume the following:

a) All BIOS INT 13H support has been installed by the time the OS starts its boot processing. I'm don't plan to cover what could happen to INT 13H once the OS starts loading its own

device drivers.

b) Drives supported by INT 13H are numbered sequentially starting with drive number 80H (80H-FFH are hard drives, 00-7FH are floppy drives).

And remember, any time a P-CHS exists it may or may not account for the CE Cylinder properly.

I have identified the following types of BIOS INT 13H support as seen by an OS during its boot time hardware configuration determination:

BIOS Type 1

Origin: Original IBM PC/XT.

BIOS call support: INT 13H AH=0xH and FDPT for BIOS drives 80H and 81H. There is no CHS translation. INT 13H AH=08H returns the P-CHS. The FDPT should contain the same P-CHS.

Description: Supports up to 528MB from a table of drive descriptions in BIOS ROM. No support for >1024 cylinders or drives >528MB or LBA.

Support issues: For >1024 cylinders or >528MB support, either an option ROM with an INT 13H replacement (see BIOS types 4-7) -or- a software driver (see BIOS type 8) must be added to the system.

BIOS Type 2

Origin: Unknown, but first appeared on systems having BIOS drive type table entries defining >1024 cylinders. Rumored to have originated at the request of Novell or SCO.

BIOS call support: INT 13H AH=0xH and FDPT for BIOS drives 80H and 81H. INT 13H AH=08H should return a L-CHS with the cylinder value limited to 1024. Beware, many BIOS perform a logical AND on the cylinder value. A correct BIOS will limit the cylinder value as follows:

```
cylinder = cylinder > 1024 ? 1024 : cylinder;
```

An incorrect BIOS will limit the cylinder value as follows (this implementation turns a 540MB drive into a 12MB drive!):

```
cylinder = cylinder & 0x03ff;
```

The FDPT will return a P-CHS that has the full cylinder value.

Description: For BIOS drive numbers 80H and 81H, this BIOS type supports >1024 cylinders or >528MB without using a translated CHS in the FDPT. INT 13H AH=08H truncates cylinders to 1024 (beware of buggy implementations). The FDPT can show >1024 cylinders thereby allowing an OS to support drives >528MB. May convert the L-CHS or P-CHS directly to an LBA if the ATA device supports LBA.

Support issues: Actual support of >1024 cylinders is OS specific -- some OS's may be able to place OS specific partitions spanning or beyond cylinder 1024. Usually all OS boot code must be within first 1024 cylinders. The FDISK program of an OS that supports such partitions uses an OS specific partition table entry format to identify these paritions. There does not appear to be a standard (de facto or otherwise) for this unusual partition table entry. Apparently one method is to place -1 into the CHS fields and use the LBA fields to describe the location of the partition. This DOES NOT require the drive to support LBA addressing. Using an LBA in the partition table entry is just a trick to get around the CHS limits in the partition table entry. It is unclear if such a partition table entry will be ignored by an OS that does not understand what it is. For an OS that does not support such partitions, either an option ROM with an INT 13H replacement (see BIOS types 4-7) -or- a software driver (see BIOS type 8) must be added to the system.

Note: OS/2 can place HPFS partitions and Linux can place Linux partitions beyond or spanning cylinder 1024. (Anyone know of other systems that can do the same?)

BIOS Type 3

Origin: Unknown, but first appeared on systems having BIOS drive type table entires defining >1024 cylinders. Rumored to have originated at the request of Novell or SCO.

BIOS call support: INT 13H AH=0xH and FDPT for BIOS drives 80H and 81H. INT 13H AH=08H can return an L-CHS with more than 1024 cylinders.

Description: This BIOS is like type 2 above but it allows up to 4096 cylinders (12 cylinder bits). It does this in the INT 13H AH=0xH calls by placing two most significant cylinder bits (bits 11 and 10) into the upper two bits of the head number (bits 7 and 6).

Support issues: Identification of such a BIOS is difficult. As long as the drive(s) supported by this type of BIOS have <1024 cylinders this BIOS looks like a type 2 BIOS because INT 13H AH=08H should return zero data in bits 7 and 6 of the head information. If INT 13H AH=08H returns non zero data in bits 7 and 6 of the head information, perhaps it can be assumed

that this is a type 3 BIOS. For more normal support of >1024 cylinders or >528MB, either an option ROM with an INT 13H replacement (see BIOS types 4-7) -or- a software driver (see BIOS type 8) must be added to the system.

Note: Apparently this BIOS type is no longer produced by any BIOS vendor.

BIOS Type 4

Origin: Compaq. Probably first appeared in systems with ESDI drives having >1024 cylinders.

BIOS call support: INT 13H AH=0xH and EDPT for BIOS drives 80H and 81H. If the drive has <1024 cylinders, INT 13H AH=08H returns the P-CHS and a FDPT is built. If the drive has >1024 cylinders, INT 13H AH=08H returns an L-CHS and an EDPT is built.

Description: Looks like a type 2 BIOS when an FDPT is built. Uses CHS translation when an EDPT is used. May convert the L-CHS directly to an LBA if the ATA device supports LBA.

Support issues: This BIOS type may support up to four drives with a EDPT (or FDPT) for BIOS drive numbers 82H and 83H located in memory following the EDPT (or FDPT) for drive 80H. Different CHS translation algorithms may be used by the BIOS and an OS.

BIOS Type 5

Origin: The IBM/Microsoft BIOS Extensions document. For many years this document was marked "confidential" so it did not get wide spread distribution.

BIOS call support: INT 13H AH=0xH, AH=4xH and EDPT for BIOS drives 80H and 81H. INT 13H AH=08H returns an L-CHS. INT 13H AH=41H and AH=48H should be used to get P-CHS configuration. The FDPT/EDPT should not be used. In some implementations the FDPT/EDPT may not exist.

Description: A BIOS that supports very large drives (>1024 cylinders, >528MB, actually >8GB), and supports the INT 13H AH=4xH read/write functions. The AH=4xH calls use LBA addressing and support drives with up to 2^64 sectors. These calls do NOT require that a drive support LBA at the drive interface. INT 13H AH=48H describes the L-CHS used at the INT 13 interface and the P-CHS or LBA used at the drive interface. This BIOS supports the INT 13 AH=0xH calls the same as a BIOS type 4.

Support issues: While the INT 13H AH=4xH calls are well defined, they are not implemented in many systems shipping

today. Currently undefined is how such a BIOS should respond to INT 13H AH=08H calls for a drive that is >8GB. Different CHS translation algorithms may be used by the BIOS and an OS.

Note: Support of LBA at the drive interface may be automatic or may be under user control via a BIOS setup option. Use of LBA at the drive interface does not change the operation of the INT 13 interface.

BIOS Type 6

Origin: The Phoenix Enhanced Disk Drive Specification.

BIOS call support: INT 13H AH=0xH, AH=4xH and EDPT for BIOS drives 80H and 81H. INT 13H AH=08H returns an L-CHS. INT 13H AH=41H and AH=48H should be used to get P-CHS configuration. INT 13H AH=48H returns the address of the Phoenix defined "FDPT Extension" table.

Description: A BIOS that supports very large drives (>1024 cylinders, >528MB, actually >8GB), and supports the INT 13H AH=4xH read/write functions. The AH=4xH calls use LBA addressing and support drives with up to 2^64 sectors. These calls do NOT require that a drive support LBA at the drive interface. INT 13H AH=48H describes the L-CHS used at the INT 13 interface and the P-CHS or LBA used at the drive interface. This BIOS supports the INT 13 AH=0xH calls the same as a BIOS type 4. The INT 13H AH=48H call returns additional information such as host adapter addresses, DMA support, LBA support, etc, in the Phoenix defined "FDPT Extension" table.

Phoenix says this this BIOS need not support the INT 13H AH=4xH read/write calls but this BIOS is really an extension/enhancement of the original IBM/MS BIOS so most implementations will probably support the full set of INT 13H AH=4xH calls.

Support issues: Currently undefined is how such a BIOS should respond to INT 13H AH=08H calls for a drive that is >8GB. Different CHS translation algorithms may be used by the BIOS and an OS.

Note: Support of LBA at the drive interface may be automatic or may be under user control via a BIOS setup option. Use of LBA at the drive interface does not change the operation of the INT 13 interface.

BIOS Type 7

Origin: Described in the Western Digital Enhanced IDE Implementation Guide.

BIOS call support: INT 13H AH=0xH and FDPT or EDPT for BIOS

drives 80H and 81H. An EDPT with a L-CHS of 16 heads and 63 sectors is built when "LBA mode" is enabled. An FDPT is built when "LBA mode" is disabled.

Description: Supports >1024 cylinders or >528MB using a EDPT with a translated CHS *** BUT ONLY IF *** the user requests "LBA mode" in the BIOS setup *** AND *** the drive supports LBA. As long as "LBA mode" is enabled, CHS translation is enabled using a L-CHS with <=1024 cylinders, 16, 32, 64, ..., heads and 63 sectors. Disk read/write commands are issued in LBA mode at the ATA interface but other commands are issued in P-CHS mode. Because the L-CHS is determined by table lookup based on total drive capacity and not by a multiply/divide of the P-CHS cylinder and head values, it may not be possible to use the simple (and faster) bit shifting L-CHS to P-CHS algorithms.

When "LBA mode" is disabled, this BIOS looks like a BIOS type 2 with an FDPT. The L-CHS used is taken either from the BIOS drive type table or from the device's Identify Device data. This L-CHS can be very different from the L-CHS returned when "LBA mode" is enabled.

This BIOS may support FDPT/EDPT for up to four drives in the same manner as described in BIOS type 4.

The basic problem with this BIOS is that the CHS returned by INT 13H AH=08H changes because of a change in the "LBA mode" setting in the BIOS setup. This should not happen. This use or non-use of LBA at the ATA interface should have no effect on the CHS returned by INT 13H AH=08H. This is the only BIOS type know to have this problem.

Support issues: If the user changes the "LBA mode" setting in BIOS setup, INT 13H AH=08H and the FDPT/EDPT change which may cause *** DATA CORRUPTION ***. The user should be warned to not change the "LBA mode" setting in BIOS setup once the drive has been partitioned and software installed. Different CHS translation algorithms may be used by the BIOS and an OS.

BIOS Type 8

Origin: Unknown. Perhaps Ontrack's Disk Manager was the first of these software drivers. Another example of such a driver is Micro House's EZ Drive.

BIOS call support: Unknown (anyone care to help out here?). Mostly likely only INT 13H AH=0xH are support. Probably a FDPT or EDPT exists for drives 80H and 81H.

! Description: A software driver that "hides" in the MBR such that it is loaded into system memory before any OS boot

processing starts. These drivers can have up to three parts: a part that hides in the MBR, a part that hides in the remaining sectors of cylinder 0, head 0, and an OS device driver. The part in the MBR loads the second part of the driver from cylinder 0 head 0. The second part provides a replacement for INT 13H that enables CHS translation for at least the boot drive. Usually the boot drive is defined in CMOS setup as a type 1 or 2 (5MB or 10MB drive). Once the second part of the driver is loaded, this definition is changed to describe the true capacity of the drive and INT 13H is replaced by the driver's version of INT 13H that does CHS translation. In some cases the third part, an OS specific device driver, must be loaded to enable CHS translation for devices other than the boot device.

! I don't know the details of how these drivers respond to INT 13H AH=08H or how they set up drive parameter tables (anyone care to help out here?). Some of these drivers convert the L-CHS to an LBA, then they add a small number to the LBA and finally they convert the LBA to a P-CHS. This in effect skips over some sectors at the front of the disk.

Support issues: Several identified -- Some OS installation programs will remove or overlay these drivers; some of these drivers do not perform CHS translation using the same algorithms used by the other BIOS types; special OS device drivers may be required in order to use these software drivers For example, under MS Windows the standard FastDisk driver (the 32-bit disk access driver) must be replaced by a driver that understands the Ontrack, Micro House, etc, version of INT 13H. Different CHS translation algorithms may be used by the driver and an OS.

! The hard disk vendors have been shipping these drivers with their drives over 528MB during the last year and they have been ignoring the statements of Microsoft and IBM that these drivers would not be supported in future OS's. Now it appears that both Microsoft and IBM are in a panic trying to figure out how to support some of these drivers in WinNT, Win95 and OS/2. It is unclear what the outcome of this will be at this time.

! NOTE: THIS IS NOT A PRODUCT ENDORSEMENT! An alternate solution for an older ISA system is one of the BIOS replacement cards. This cards have a BIOS option ROM. AMI makes such a card called the "Disk Extender". This card replaces the motherboard's INT 13H BIOS with a INT 13H BIOS that does some form of CHS translation. Another solution for older VL-Bus systems is an ATA-2 (EIDE) type host adapter card that provides a option ROM with an INT 13H replacement.

BIOS Type 9

Origin: SCSI host adapters.

BIOS call support: Probably INT 13H AH=0xH and FDPT for BIOS drives 80H and 81H, perhaps INT 13H AH=4xH.

Description: Most SCSI host adapters contain an option ROM that enables INT 13 support for the attached SCSI hard drives. It is possible to have more than one SCSI host adapter, each with its own option ROM. The CHS used at the INT 13H interface is converted to the LBA that is used in the SCSI commands. INT 13H AH=08H returns a CHS. This CHS will have <=1024 cylinders, <=256 heads and <=63 sectors. The FDPT probably will exist for SCSI drives with BIOS drive numbers of 80H and 81H and probably indicates the same CHS as that returned by INT 13H AH=08H. Even though the CHS used at the INT 13H interface looks like a translated CHS, there is no need to use a EDPT since there is no CHS-to-CHS translation used. Other BIOS calls (most likely host adapter specific) must be used to determine other information about the host adapter or the drives.

The INT 13H AH=4xH calls can be used to get beyond 8GB but since there is little support for these calls in today's OS's, there are probably few SCSI host adapters that support these newer INT 13H calls.

Support issues: Some SCSI host adapters will not install their option ROM if there are two INT 13H devices previously installed by another INT 13H BIOS (for example, two MFM/RLL/ESDI/ATA drives). Other SCSI adapters will install their option ROM and use BIOS drive numbers greater than 81H. Some older OS's don't understand or use BIOS drive numbers greater than 81H. SCSI adapters are currently faced with the >8GB drive problem.

BIOS Type 10

Origin: A european system vendor.

BIOS call support: INT 13H AH=0xH and FDPT for BIOS drives 80H and 81H.

Description: This BIOS supports drives >528MB but it does not support CHS translation. It supports only ATA drives with LBA capability. INT 13H AH=08H returns an L-CHS. The L-CHS is converted directly to an LBA. The BIOS sets the ATA drive to a P-CHS of 16 heads and 63 sectors using the Initialize Drive Parameters command but it does not use this P-CHS at the ATA interface.

```
!  Support issues:  OS/2 will probably work with this BIOS as
   long as the drive's power on default P-CHS mode uses 16 heads
   and 63 sectors.  Because there is no EDPT, OS/2 uses the ATA
   Identify Device power on default P-CHS, described in
   Identify Device words 1, 3 and 6 as the current P-CHS for the
   drive.  However, this may not represent the correct P-CHS.  A
   newer drive will have the its current P-CHS information in
   Identify Device words 53-58 but for some reason OS/2 does not
   use this information.
```

NOTE : The text has been written by **Hale Landis** (landis@sugs.tware.com). Some formatting has been done by GDR!