BIOS

Enhanced Disk Drive Specification

Version 1.1

May 9, 1995

Technical Editor:

Curtis E. Stevens
Phoenix Technologies
2575 M^{c}Cabe Way
Irvine, Ca. 92714
Phone: (714) 440-8000
Fax: (714) 440-8300
Curtis_Stevens@BanNet.PTLTD.COM

Phoenix Technologies Ltd.

Revision History		
Rev	**Date**	**Description**
1.0	January 25, 1994	Initial Release
1.1	January 25, 1995	Added the following: • Description of the 528 MB limitation • Description of compatibility issues caused by translation • Description of Int 13h Extensions as implemented by Phoenix • Description of the Translated Fixed Disk Parameter Table. • Support for ATAPI devices • Support for translation reporting

Companies Supporting this Specification

Phoenix Technologies
2575 McCabe Way
Irvine, Ca. 92714
Phone: (714) 440-8000
Fax: (714) 440-8300

Table of Contents

1. Overview

The Phoenix Technologies Enhanced Disk Drive Support Specification is a comprehensive BIOS solution to problems that result from advances in the disk-drive industry. The purpose of this specification is to provide a downward compatible method for expanding the current Fixed Disk Parameter Tables, as well as a new method of accessing these tables. This new specification not only provides for the unlimited expansion of systems beyond two drives, but it also provides information regarding IDE-compatible features enabled by the BIOS for OS and application software.

1.1 Scope

This paper assumes that the reader is familiar with the conventional Int 13h interface, the usage of the Fixed Disk Parameter table, and the basic operation of IDE devices. This document describes in detail extensions to the BIOS and extensions to the data maintained by the BIOS.

1.2 Introduction

The disk-drive industry has increased the capacity and functionality of the IDE-compatible disk drive, finally surpassing the capability of the BIOS to service these new capabilities. This specification solves the following BIOS specific problems:

- The BIOS must now support drives with a capacity greater than 528 MB. At the present time the conventional Int 13h interface has a limit of 1024 cylinders.
- The Int 13h interface allows more than two drives to be attached to a system but has no consistent method for storing the additional parameters.
- New, CHS-independent methods for accessing the drives have now been defined. These are drive-geometry independent and require a different method of data representation and operation.
- New methods of data transfer are now supported. Devices now support additional DMA modes as well as multi-sector data transfers and Fast PIO. None of this information is currently available to OS or application software.
- Systems require more than two disk drives, and with this requirement comes the requirement to assign the order in which the drives are to be accessed.

This document is divided into three sections:

1. **Parameter Tables**. Describes alterations to the Fixed Disk Parameter Table (FDPT) as well as the information in the Extended FDPT.
2. **BIOS Extensions**. Describes new Int 13h functions, which accomplish the following:
   - Device access without regard to geometry
   - Removable Media Support
   - Hardware configuration reporting
3. **Compatibility Issues**. Discusses some of the topics related to implementing the structures in this document.

1.3 Definition of Terms

1.3.1 Enhanced BIOS

All Enhanced BIOS's support drives that are greater than 528 MB. Enhanced BIOS's can optionally support the following:

- PIO Mode 3 or greater
- DMA Mode 1 or greater

1.3.2 Enhanced IDE Device

An Enhanced IDE Device is a hard disk or other device which interfaces to the system via Integrated Drive Electronics (IDE). These devices must be ATA-2/ATAPI compliant and have *both* of the following capabilities:

- PIO Mode 3 or greater
- DMA Mode 1 or greater

1.3.3 Enhanced IDE Channel

An Enhanced IDE Channel (or Chip or interface card) provides a communications port with an Enhanced IDE Device. These channels must be ATA-2/ATAPI compliant and have *one* of the following capabilities:

- PIO Mode 3 or greater
- DMA Mode 1 or greater

1.3.4 Conventional vs Enhanced

When a word, term, or phrase is modified by the word "conventional" it refers to the old style or current method of operation. Conversely, when a word, term, or phrase is modified by the word "enhanced" it means there is an old/"conventional" and a new/"enhanced" method of operation

2. Fixed Disk Parameter Tables (FDPT)

This section provides a comprehensive description of the data which the BIOS makes available to OS and application software. Register calling conventions limit the conventional Int 13h interface, resulting in the limitation of OS and application software as well. The Int 13h interface operates directly from the FDPT and therefore limits the size of the cylinder field in this table.

There is a compatible version of the FDPT, which has been embraced by some OS developers, that allows the BIOS to change the way the geometry is represented and then to translate the changed ("logical") geometry back to actual ("physical") geometry when a disk access is performed. This Enhanced FDPT is identified by a signature byte (A0h) which flags the system that some form of translation is taking place. Phoenix has chosen to use this format for representing drives with more than 1024 cylinders.

The FDPT is directly accessable only for drives 80h and 81h. Int 41h and Int 46h provide pointers directly to the FDPT for drive 80h and 81h respectivly. These pointers are maintained for backward compatibility with older DOS applications only. Geometry information for drive numbers 82h and above is available only through Int 13h Fn 8h and Fn 48h.

2.1 Fixed Disk Parameter Table (FDPT) Extensions

It has become necessary for the BIOS to store information representing the type of translation currently in operation, as well as any information relating to the current operation of the drive. One purpose of this specification is to define a standard format for this extension area and to document its fields and functions so that other OS/Application software can easily use this information. This new information, called the "Fixed Disk Parameter Table Extension" is 16 bytes long and is accessed via the extended Int 13h functions described in this document. Please see table 1 for a layout of the Fixed Disk Parameter Table with its Extension.

2.2 The 528-Megabyte Barrier

The BIOS provides Int 13h services for accessing ATA drives from DOS. In the past the Cylinder-Head-Sector (CHS) values supplied to the Int 13h interface were passed to the drive without modification. This method of access allows other “ill-behaved” applications to successfully access the drive, bypassing the BIOS Int 13h interface. IDE drives have now grown to the point that the Int 13h interface can no longer address the full media by passing CHS values directly to the drive. The following table illustrates the limitations caused by the differences between the Int 13h and ATA maximum geometries:

	BIOS	**IDE**	**Limit**
Max Sectors/Track	63	255	63
Max Heads	256	16	16
Max Cylinders	1024	65536	1024
Capacity	8.4GB	136.9GB	*528 MB*

This table illustrates how the conventional Int 13h interface with an 8.4 GB limit is restricted to 528 MB (63 * 16 * 1024 * 512). One solution to this problem is to address the drive using the Int 13h Extensions described in this document. Another solution is to create a false geometry that “fits” within Int 13h limitations, and also uses the full capacity of the drive. This capability is called geometric or drive translation. The translated geometry is applied in a manner that causes all sectors to maintain the same physical location on the media as when the drive is used in an untranslated environment. The Int 13h interface only has 10 bits for the cylinder, therefore *Int 13h Fn 08h* ***always*** returns the altered geometry information. This allows all DOS applications to function normally. Windows 3.11 and below functions normally when 32-bit disk access mode is disabled. A Windows driver which supports the geometry reported by *Int 13h Fn 08h* is required for 32-bit disk access mode.

Several BIOS manufacturers have implemented drive translation and now there are new problems. There is no standard for assigning the altered geometry when drive translation is employed. This has created a drive interchange problem. The root of the problem is in the boot code and Partition Table stored at CHS=0,0,1. The boot code ***assumes*** that the geometry stored in the Partition Table matches the geometry returned by Int 13h Fn 08h. When a drive is moved from the original system where the partitions were defined, to a new system with an incompatible BIOS, the boot code will use Int 13h with boundaries defined by the Partition Table in the boot sector. Boot failures are caused because the geometry in the Partition Table does not match the BIOS geometry in the new system.

Phoenix has implemented a simple bit-shift mapping scheme to create altered drive

geometries. This method has the advantage of working with all ATA drives, including those drives which do not support LBA. A second advantage is that operation is fast and the code is small. The disadvantage of this method is that it lacks the flexibility to translate ***all*** geometries reported by a drive with a capacity less than 8.4GB. However, drives which are ATA-2 (X3T10 Document 948D) compatible will always supply geometries that can be translated. Annex D places limits on geometries for drives with less than an 8GB capacity. Therefore, all drives can be interfaced using this method. The Phoenix method of translation manipulates the HEAD and CYLINDER part of the geometry, but not the SECTORS per track. The following table describes the Phoenix translation capability:

Actual Cylinders	Actual Heads	Altered Cylinder	Altered Heads	Max Size
1<C≤1024	1<H≤16	C=C	H=H	528 MB
1024<C≤2048	1<H≤16	C=C/2	H=H*2	1GB
2048<C≤4096	1<H≤16	C=C/4	H=H*4	2.1GB
4096<C≤8192	1<H≤16	C=C/8	H=H*8	4.2GB
8192<C≤16384	1<H≤16	C=C/16	H=H*16	8.4GB
16384<C≤32768	1<H≤8	C=C/32	H=H*32	8.4GB
32768<C≤65536	1<H≤4	C=C/64	H=H*64	8.4GB

Another popular method of translation exists which is very flexible and works well on drives with LBA. This method places no limits on the reported drive geometry. The disadvantage of this method is that it does not function well on drives which do not support LBA, and the code is a little larger than the Phoenix method. The following table describes the LBA assisted translation method:

Range	Sectors	Heads	Cylinders
1<X≤528MB	63	16	X/(63*16*512)
528MB<X≤1GB	63	32	X/(63*32*512)
1GB<X≤2.1GB	63	64	X/(63*64*512)
2.1GB<X≤4.2GB	63	128	X/(63*128*512)
4.2GB<X≤8.4GB	63	256	X/(63*255*512)

These two translation methods yield similar geometries in many cases. The difference between the two translations methods becomes apparent when a drive reports less than 63 sectors per track. The LBA assisted method *always* assigns a geometry with 63 sectors per track. The Phoenix method uses the sectors returned by the drive. This document provides a way to report the type of translation employed by the BIOS. Currently there is no solution for the interchange problem.

Table 1

Standard Fixed Disk Parameter Table (FDPT)

Byte	Type	Description
0-1	Word	Physical Number of Cylinders
2	Byte	Physical Number of Heads
3	Byte	Reserved
4	Byte	Reserved
5-6	Word	Precompensation (Obsolete)
7	Byte	Reserved
8	Byte	Drive Control Byte
9-10	Word	Reserved
11	Byte	Reserved
12-13	Word	Landing Zone (Obsolete)
14	Byte	Sectors per Track
15	Byte	Reserved

Translated Fixed Disk Parameter Table (FDPT)

Byte	Type	Description
0-1	Word	Logical Cylinders, limit 1024
2	Byte	Logical Heads, limit 256
3	Byte	A0h Signature, indicates translated table
4	Byte	Physical Sectors pre Track
5-6	Word	Precompensation (Obsolete)
7	Byte	Reserved
8	Byte	Drive Control Byte
9-10	Word	Physical Cylinders, limit 65536
11	Byte	Physical Heads , limit 16
12-13	Word	Landing Zone (Obsolete)
14	Byte	Logical Sectors per Track, limit 63
15	Byte	Checksum

Fixed Disk Parameter Table (FDPT) Extension

Byte	Type	Description
0-1	Word	I/O port base address
2-3	Word	Control port address
4	Byte	Head Register Upper Nibble Bit 0-3 Reserved, must be 0 Bit 4 Master/Slave Bit (0=Master) Bit 5 Reserved, must be 1 Bit 6 LBA Enable (1 = Enabled) Bit 7 Reserved, must be 1
5	Byte	Internal Use Only.
6	Byte	IRQ Information Bits 0-3 IRQ for this drive Bits 4-7 Reserved, must be 0
7	Byte	Sector count for multi-sector transfers
8	Byte	DMA Information Bits 0-3 DMA Channel Bits 4-7 DMA Type
9	Byte	PIO Information Bit 0-3 PIO Type Bits 4-7 Reserved, must be 0
10-11	Word	Hardware Specific Option Flags Bit 0 Fast PIO accessing enabled Bit 1 DMA accessing enabled Bit 2 Block PIO accessing enabled Bit 3 CHS translation enabled Bit 4 LBA translation enabled Bit 5 Removable Media Bit 6 ATAPI Device Bit 7 32-bit transfer mode Bit 8 ATAPI Device uses Interrupt DRQ Bits 9-10 CHS Translation Type Bits 11-15 Reserved, must be 0
12-13h	Word	Reserved, must be 0
14	Byte	11h, Revision level of this extension
15	Byte	Checksum, 2's complement of the sum of bytes 0-14

2.3 Fixed Disk Parameter Table (FDPT) Definitions

Table 1 describes two possible FDPTs. The Standard FDPT is used for devices with less than 1024 cylinders, the translated FDPT is used for devices with more than 1024 cylinders (devices which require translation). These tables are only available for drives 80h and 81h.

2.3.1 Physical Values

Physical values are the values returned by the ID Drive command in words 1, 3 and 6. In the Standard table these values are limited to 1024 Cylinders, 16 Heads and 63 Sectors. In the translated table the limits are 65536 Cylinders, 16 Heads, and 63 Sectors.

2.3.2 Logical Values

Logical values only appear in the translated table. These values represent the geometry which the BIOS is receiving from applications that use Int 13h to address the device. Int 13h will then convert the Logical Values it receives to Physical values when it accesses the device. Logical values are limited to 1024 Cylinders, 256 Heads, and 63 Sectors per Track.

2.3.3 Obsolete Fields

The Precompensation and Landing Zone fields are obsolete because they are handled internally by today's IDE devices. These two fields are documented for compatibility reasons only. We expect to use these fields for other purposes in the future.

2.4 Fixed Disk Parameter Table (FDPT) Extension

The FDPT Extension provides hardware configuration information to applications that bypass Int 13h for accessing an ATA device. An application can receive a pointer to the FDPT Extension by issuing Int 13h Fn 48h.

2.4.1 Bytes 0-1 - I/O Port Base

This word is the physical base address of the ATA C*ommand Block Registers*. Any application which provides a proprietary interface to the device can use this base address.

2.4.2 Bytes 2-3 - Control Port Base

This word is the physical address of the ATA *Control Block Register*. Any application which provides a proprietary interface to the device can use this address.

2.4.3 Byte 4 - Head Prefix

This is the upper nibble of the byte used for selecting the head. This byte is logically ORed with the head each time the disk is addressed. The Master/Slave bit and the LBA addressing bit are preset, which makes these functions transparent to any software using this extension.

2.4.4 Byte 5 - Internal Use Only

This field is for BIOS use only. Phoenix uses this byte to store the number of shifts required for converting a cylinder count that is greater than 1024 to one that is less than 1024.

2.4.5 Byte 6 - IRQ

Each ATA channel requires an interrupt. This byte enables applications to access the interrupts for additional ATA channels.

2.4.6 Byte 7 - Sector Count

If the BIOS has configured the drive for multi-sector transfers, this field contains the block size of the transfer, in sectors, used by the BIOS.

2.4.7 Byte 8 - DMA Channel/DMA Type

If the BIOS has configured the system to perform multi-word DMA data transfers in place of the normal PIO transfers, this field specifies the DMA mode in the upper nibble (currently 0, 1 or 2), as per the ATA-2 definition, and the DMA Channel in the lower nibble. Note that the DMA Type field does not follow the format of the data returned by the drive. The value of the DMA mode is not limited to 2. Phoenix anticipates that new specifications will define mode 3 and 4 DMA.

2.4.8 Byte 9 - PIO Type

If the BIOS has configured the system to perform Fast PIO data transfers in place of normal PIO data transfers, this field specifies the PIO mode as per the ATA-2 definition (currently 1, 2, 3 or 4).

2.4.9 Byte 10-11 - Hardware Specific Option Flags

This word has a bit for each of the currently defined drive enhancements. If this word is 0, the device is a "standard" drive with less than 1024 cylinders using normal PIO-type data transfers.

2.4.9.1 Bit 0 - Fast PIO

If the system is configured for Fast PIO, this bit is set to 1 and byte 9 (PIO Type) should be used to configure the system. If this bit is 0, the PIO-Type field should be ignored, regardless of any data it may contain.

2.4.9.2 Bit 1 - Fast DMA

If the system is configured for Fast DMA, this bit is set to 1 and byte 8 (DMA Channel/DMA Type) should be used to configure the system. If this bit is 0, the DMA Channel/DMA Type field should be ignored, regardless of any data it may contain.

2.4.9.3 Bit 2 - Block PIO

If the system is configured for multi-sector transfers, this bit is set to 1 and byte 7 (Sector Count) specifies the number of sectors used for each data transfer. If Block PIO is disabled, ignore the Sector Count field.

2.4.9.4 Bit 3 - CHS Translation

If the disk-drive has more than 1024 cylinders, this bit is set to 1. CHS translation translates from "Logical" to "Physical" drive geometry. When this bit is 1 Phoenix BIOS uses byte 5 (Internal Use) to perform the actual translation.

2.4.9.5 Bit 4 - LBA Translation

If the system is configured for LBA type addressing, this bit is set to 1. When LBA translation is on, the Extended Int 13h interface (Fns 41h-48h) pass LBA values directly to the device. The conventional Int 13h interface ignores this bit and *always* uses CHS. LBA-type addressing is available on drives with less than 1024 cylinders, and therefore it should not be assumed that bit 3 (CHS translation) will always be enabled whenever this bit is enabled.

2.4.9.6 Bit 5 - Removable Media

If the device supports removable media, this bit is set to 1.

2.4.9.7 Bit 6 - ATAPI Device

If this ATA device uses the packet interface (ATAPI), this bit is set to 1.

2.4.9.8 Bit 7 - 32-bit Transfer Mode

If the BIOS has configured the system to perform 32-bit wide data transfers, this bit is set to 1.

2.4.9.9 Bit 8 - ATAPI Device uses Interrupt DRQ

If bit 6 is set to zero, then this field is ignored and must be 0. If bit 6 is set to one, this bit indicates how the ATAPI devices signals it is ready to receive a packet command. When this bit is 1, the ATAPI device returns an interrupt when it is ready for a packet. When this bit 0, the ATAPI device sets DRQ when it is ready for a packet.

2.4.9.10 Bits 9-10 - Translation Type

If bit 3 is zero then this field is ignored and must be 0. If bit 3 is 1 then this field identifies the geometric translation as follows:

Bits 9-10	Description
00	Phoenix bit-shifting translation
01	LBA assisted translation
10	Reserved, invalid selection
11	Proprietary Translation

2.4.10 Byte 14 - Extension Revision

This is set to 11h and represents the revision level of this extension as a BCD number, where the first digit is the major revision number and the second digit is the minor revision number. Each time the FDPT Extension is updated, this revision number reflects the corresponding revision of the Enhanced Disk Drive Specification.

2.4.11 Byte 15 - Checksum

This is the two's complement of the sum of bytes 0 through 14. Adding bytes 0 through 15 should in all cases produce an 8 bit result of 0.

3. BIOS Extensions

If compatibility with current and future software is to be maintained, the BIOS must provide several extended Int 13h functions to deal with the changes in system architecture that allows for more than two drives to be supported as well as making the FDPT Extension available to application and OS software. The major purpose of these Int 13h Extensions is to remove the current requirement of using Int pointers to point at the FDPT information and to give the BIOS better control over how this data is manipulated. Phoenix has chosen to integrate this information with the Int 13h Extensions adopted by both IBM and Microsoft. The following sections fully describe the Int 13h Extensions. Modifications to the Int

13h Extensions which support changes required by this specification are printed in **BOLD**.

3.1 Calling Conventions

The extended Int 13h functions are numbered 41h-48h. These new functions are fundamentally different from the conventional Int 13h interface in the following ways:

- All addressing information is passed via a buffer, not registers.
- Register conventions have been changed to support the passing of data structures.
- Flags are used to identify optional capabilities.

3.1.1 Data Structure

The fundamental data structure for the Int 13h Extensions is the Disk Address Packet. Int 13h converts addressing information in the Disk Address Packet to physical parameters appropriate to the media. The following table describes the Disk Address Packet

.

Disk Address Packet		
Offset	**Type**	**Description**
0	Byte	Packet size in bytes. Must be 16 (10h) or greater. If the packet size is less than 16 the request is rejected with CF=1 and AH=01. Packet sizes greater than 16 are not rejected, the additional information will be ignored.
1	Byte	Reserved, must be 0
2	Byte	Number of blocks to transfer. This field has a maximum value of 127 (7Fh). If a value greater than 127 is supplied the request is rejected with CF=1 and AH=01. A block count of 0 means no data will be transferred.
3	Byte	Reserved, must be 0
4	Double Word	Address of transfer buffer. The is the buffer which Read/Write operations will use to transfer the data. This is a 32-bit address of the form Seg:Offset.
8	Quad Word	Starting absolute block number (LBA address), on the target device, of the data to be transferred. This is a 64 bit linear address. If the device supports LBA addressing this value should be passed unmodified. If the device does not support LBA addressing the following formula must hold true when the address is converted to a CHS value: **$LBA = (C_1 * H_0 + H_1) * S_0 + S_1 - 1$** Where: C_1 = Selected Cylinder Number H_0 = Number of Heads (Maximum Head Number + 1) H_1 = Selected Head Number S_0 = Maximum Sector Number S_1 = Selected Sector Number The geometry of the drive is supplied by words 1, 3 & 6 of the identify drive information for ATA compatible devices.

3.1.2 Extended Int 13h Conventions

The Int 13h Extensions normally use the following registers conventions:

AH - Function code/return status
DL - Drive number
DS:SI - disk address packet

The distinction between "removable" disks numbered 0-7Fh and "fixed" disks numbered 80h-FFh differ from conventional Int 13hh functions. Drives numbered 0-7Fh are not changed, they follow conventional Int 13h standards for floppy operation. Drives numbered 80h-FFh include traditional fixed disks, and now also include advanced removable media devices that support change-line as well as software locking and unlocking capabilities. New functions in this specification support these devices. Return codes defined for the standard Int 13h interface are still valid, and the following return codes have been added to support removable media:

- B0h - Volume Not Locked In Drive
- B1h - Volume Locked In Drive
- B2h - Volume Not Removable
- B3h - Volume In Use
- B4h - Lock Count Exceeded
- B5h - Valid Eject Request Failed

3.1.3 Int 13h Interface Subsets

It is permissible for a BIOS to support only certain subsets of the Int 13h Extensions defined in this specification. These subsets are well defined, and each must be supported in its entirety. The Interface subsets can be determined via the Check Extensions Present function. If the support bit map indicates a function is not supported and that function is subsequently invoked, then the function rejects the request with CF=1, AH=1. Currently there are three subsets defined.

3.1.3.1 Fixed Disk Access

These functions are necessary for a BIOS to support basic access to devices using the disk address packet structure as follows:

- Check Extensions Present (41h)
- Extended Read (42h)
- Extended Write (43h)
- Verify Sectors (44h)
- Extended Seek (47h)
- Get Drive Parameters (48h)

3.1.3.2 Drive Locking and Ejecting

These functions are necessary for a BIOS to support software control of drive locking and ejecting as follows:

- Check Extensions Present (41h)
- Lock/Unlock Drive (45h)
- Eject Drive (46h)
- Get Drive Parameters (48h)
- Get Extended Disk Change Status (49h)
- The Int 15h Removable Media Eject Intercept.

3.1.3.3 Enhanced Disk Drive (EDD) Support

These functions are necessary for a BIOS to provide Enhanced Disk Drive Support (EDD) as follows:

- **Check Extensions Present (41h)**
- **Get Parameters with EDD extensions (48h)**
- **Set Hardware Configuration (4Eh)**

3.2 Int 13h Extensions

3.2.1 Check Extensions Present

Entry:

AH - 41h
BX - 55aah
DL - Drive number

Exit:

carry clear
AH - Major version of extensions
AL - Internal Use Only
BX - AA55h
CX - Interface support bit map:

Bit	Description
0	1 - Device Access using the packet structure
1	1 - Drive Locking and Ejecting
2	1 - Enhanced Disk Drive Support (EDD)
3-15	Reserved, must be 0

carry set
AH - error code (01h, Invalid Command)

This function is used to check for the presence of the Int 13h extensions. If the carry flag is returned set, the extensions are not supported for the

requested drive. If the carry flag is returned cleared, BX is then checked for the value AA55h to confirm that the extensions are present. Next, the value of CX is checked to determine what subsets of this interface are supported for the requested drive. At least one subset must be supported. The major version is 21h. This indicates that the Int 13h Extensions are compliant with this specification (EDD 1.1).

3.2.2 Extended Read

Entry:
AH - 42h
DL - Drive number
DS:SI - Disk address packet

Exit:
carry clear
AH - 0
carry set
AH - error code

Transfer sectors from disk to memory. In the event of an error, the block count field of the disk address packet is filled in with the number of blocks read before the error occurred.

3.2.3 Extended Write

Entry:
AH - 43h
AL - 0 or 1, write with verify off
2, write with verify on
DL - Drive number
DS:SI - Disk address packet

Exit:
carry clear
AH - 0
carry set
AH - error code

Transfer sectors from memory to disk. If write with verify is not supported, this function rejects the request with AH=01, CF=1. Function 48h can be used to detect whether write with verify is supported. In the event of an error, the block count field of the disk address packet is filled in with the number of blocks written before the error occurred. AL must contain the values 0, 1, or 2. This function rejects all other values with AH=01, CF=1

3.2.4 Verify Sectors

Entry:
AH - 44h
DL - Drive number
DS:SI - Disk address packet

Exit:
carry clear
AH - 0
carry set
AH - error code

Verify sectors without transferring data between the device and system memory. When an error is reported the block count field of the disk address packet is filled in with the number of blocks verified before the error occurred.

3.2.5 Lock/Unlock Drive

Entry:
AH - 45h
AL - 0 - Lock volume in drive
1 - Unlock volume in drive
2 - Return lock/unlock status
3h-FFh - Invalid
DL - Drive number

Exit:
carry clear
AH - 0
AL - 1 if drive is locked, 0 if not
carry set
AH - error code.

Logically lock/unlock removable media in a specified device. All removable media devices numbered 80h and above require this function. If a fixed disk (non-removable device) supports the Drive Locking and Ejecting subset, this function always returns with success, AH=0, CF=0. There must be support for up to 255 locks per drive. A drive may not be physically unlocked until all locks to that drive have been matched with a corresponding unlock. Excess unlock calls return with carry set and AH = B0h, “Drive Not Locked”. If the number of locks supported value is exceeded on a lock request, this function rejects the request with carry set and AH = B4h, “Lock Count Exceeded”. Locking a drive without media present is a valid operation. On return from a lock or unlock request, AL contains the lock state of the drive as maintained by the BIOS. This provides for unlock requests when the lock count is greater than 0. In this case, the drive remains locked. Any physical locking and unlocking of the drive is implementation dependent, but system software operates on the assumption that a locked drive cannot be removed without an unlock request.

3.2.6 Eject Removable Media

Entry:
AH - 46h
AL - 0h, reserved
DL - Drive number

Exit:
carry clear

AH - 0
carry set
AH - error code

Eject media from the specified device. All devices with removable media numbered 80h and above require this function. If a fixed disk (non-removable device) supports the Drive Locking and Ejecting interface subset, this function always returns CF=1, AH = B2h, "Volume Not Removable". An attempt to eject media locked in a device must return with CF=1, AH = B1h, "Volume Locked In Drive". This function represents a request to remove media from the selected device. Actual ejection is implementation dependent, but system software that issues or observes this function should flush any buffers it is holding. If this function is issued for a drive without media the request is returned with CF=1, AH = 31h, "No Media In Drive". If this call is issued to an unlocked removable media device that has media present, the Int 13h code invokes Int 15h, AH=52h to determine if it may proceed with the ejection request. If the ejection request is rejected, the error code returned in AH is the same as the Int 15h error code. If the ejection request is accepted and issued, followed by a hard failure, this function returns with CF=1, AH = B5h "Valid Eject Request Failed".

3.2.7 Extended Seek

Entry:
AH - 47h
DL - Drive number
DS:SI - Disk address packet

Exit:
carry clear
AH - 0
carry set
AH - error code

Position the device at a specified sector

3.2.8 Get Drive Parameters

Entry:
AH - 48h
DL - Drive number
DS:SI - address of result buffer.

Exit:
carry clear
AH - 0
DS:SI - result buffer
carry set
AH - error code

Return physical device characteristics. This function is mandatory regardless of the interface subset which is supported.

<table>
<tr><th colspan="3">Result Buffer</th></tr>
<tr><th>Offset</th><th>Type</th><th>Description</th></tr>
<tr><td>0</td><td>Word</td><td>Buffer Size, must be 26 or greater. <u>The caller sets this value to the maximum buffer size</u>. If the length of this buffer is less than 30, this functions does not return the pointer to the Enhanced Disk Drive structure (EDD). If the Buffer Size is 30 or greater on entry, it is set to exactly 30 on exit. If the Buffer Size is between 26 and 29, it is set to exactly 26 on exit. If the Buffer Size is less than 26 on entry an error is returned.</td></tr>
<tr><td>2</td><td>Word</td><td>Information Flags

In the following table, a 1 bit indicates that the feature is available, a 0 bit indicates the feature is not available and will operate in a manner consistent with the conventional Int 13h interface.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>DMA boundary errors are handled transparently</td></tr>
<tr><td>1</td><td>The geometry supplied in bytes 8-12 is valid</td></tr>
<tr><td>2</td><td>Device is removable</td></tr>
<tr><td>3</td><td>Device supports write with verify</td></tr>
<tr><td>4</td><td>Device has change line support (bit 2 must be set)</td></tr>
<tr><td>5</td><td>Device is lockable (bit 2 must be set).</td></tr>
<tr><td>6</td><td>Device geometry is set to maximum, no media is present (bit 2 must be set). This bit is turned off when media is present in a removable media device.</td></tr>
<tr><td>7-15</td><td>Reserved, must be 0</td></tr>
</table></td></tr>
<tr><td>4</td><td>Double Word</td><td>Number of physical cylinders. This is 1 greater than the maximum cylinder number. Use Int 13h Fn 08h to find the logical number of cylinders.</td></tr>
<tr><td>8</td><td>Double Word</td><td>Number of physical heads. This is 1 greater than the maximum head number. Use Int 13h Fn 08h to find the logical number of heads.</td></tr>
<tr><td>12</td><td>Double Word</td><td>Number of physical sectors per track. This number is the same as the maximum sector number because sector addresses are 1 based. Use Int 13h Fn 08h to find the logical number of sectors per track.</td></tr>
<tr><td>16</td><td>Quad Word</td><td>Number of physical sectors. This is 1 greater than the maximum sector number.</td></tr>
<tr><td>24</td><td>Word</td><td>Number of bytes in a sector.</td></tr>
<tr><td>26</td><td>Double Word</td><td>Pointer to Enhanced Disk Drive (EDD) configuration parameters. This field is only present if Int 13h, Fn 41h, CX register bit 2 is enabled. This field points to a temporary buffer which the BIOS may re-use on subsequent Int 13h calls. A value of FFFFh:FFFFh in this field means that the pointer is invalid.</td></tr>
</table>

3.2.9 Get Extended Disk Change Status

Entry:

AH - 49h
DL - Drive number

Exit:

carry clear
AH - 00, change-line inactive
carry set
AH - 06, change-line active

Return disk change status. If this function returns with carry flag set, the media has not necessarily been changed; the change line can be activated by simply unlocking and locking the drive door without removing the media. This function corresponds to Int 13h Function 16h, but explicitly allows any drive number to be passed in. If a fixed disk (non-removable device) supports the Drive Locking and Ejecting interface subset, this function always returns with success, AH=0, CF=0.

3.2.10 Set Hardware Configuration

Entry:

AH - 4Eh
AL - Hardware Configuration Subfunction

Sub-function	Description
0	Enable Prefetch
1	Disable Prefetch
2	Set Maximum PIO transfer Mode. Set the maximum rate allowed by both the platform and the drive.
3	Set PIO Mode 0. Clear to the minimum PIO transfer rate.
4	Set Default PIO transfer Mode. Return the system to the PIO Mode enabled by SETUP.
5	Enable Int 13h DMA maximum Mode
6	Disable Int 13h DMA

DL - Drive Number

Exit:

carry clear
AH - 0
AL - 0 if command was "safe"
1 if other devices are affected
carry set
AH - error code

The purpose of this function is to allow non-device-specific software to configure motherboard devices for optimal operation. Conventional ATA channels may have 2 devices attached, but this function operates on a single-device basis. This mandates the need for a "safe" bit which is returned in AL. If the chip supports the requested subfunction on a "device" basis AL, is set to 0. If the chips only supports the setting on a "channel" basis, AL is set to 1. Once this function has been invoked, all subsequent Int 13h device-access functions use the new mode. This means that if "DMA Maximum" is enabled, Int 13h Fn 02h reads from the device using DMA transfers. The DMA/PIO selections are mutually exclusive. When "DMA Maximum" is enabled, "PIO Maximum" is disabled. If the ATA hardware/BIOS do not support mode changes this function returns with carry set and AH=1

3.2.11 Int 15h Removable Media Eject

Entry:

AH - 52h
DL - Drive number

Exit:

carry clear
AH - 0, ejection may proceed
carry set
AH - error code,
B1h or B3h, Ejection is rejected

Call this function in response to a software request (Int 13h, AH=46h, Eject drive) to eject media from a removable media device. Typically a user will press an eject button or use a software command to request that a particular volume be ejected. The code that handles this request will issue Int 15h, Fn 52h before actually honoring the request. By default the Int 15h handler returns with ejection accepted status. A disk cache program could hook this Int 15h call and return acceptance or rejection based on the state of its buffers for this disk. It could also be used by operating system software as a media changed signal.

4. Compatibility Issues

Adding new features to the BIOS, or changing the functionality of an existing BIOS, will always impact software performance. The purpose of this section is to discuss how to handle some of the anticipated issues.

4.1 Int 41h/46h

Int 41h and Int 46h are platform-specific issues. When a platform supports Int 41h/46h, the BIOS uses Int 41h to point to an FDPT for drive 80h, and Int 46h to point to and FDPT for drive 81h. These pointers are maintained for backward compatibility purposes only. DOS/Windows applications should use Int 13 to get geometry information. The BIOS maintains pointers to drives greater than 81h internally. If the platform does not support Int 41h/46h, the BIOS maintains all pointers internally. In either case, Int 13h Fns 8h and 48h always return the requested information.

4.2 Disk Drive Mapping

When a system has more than 1 drive, the person configuring the system needs a way to set the relationship between the BIOS references (80h, 81h, 82h, ...) and the physical drive. The FDPT Extension easily allows this capability because it stores the physical port addresses and control bits used for setting Master/Slave as well as LBA accessing. One side effect of this implementation is that any drive can be the boot drive, and slave drives may be at any location. The following are some examples of legal configurations:

Standard Configuration:

80h - Primary Master [Int 41h]
81h - Primary Slave [Int 46h]

Alternate Configurations:

80h - Primary Master [Int 41h]
81h - Secondary Master [Int 46h]
82h - Secondary Slave

80h - Secondary Master [Int 41h]
81h - Primary Master [Int 46h]
82h - CD (Secondary Slave)

80h - Secondary Slave [Int 41h]
81h - Secondary Master [Int 46h]
82h - Primary Master
83h - Primary Slave

4.3 Geometric Translations

Some applications get device geometry information simply by reading the tables which are accessed via the Int 41h/46h pointers, they fail to call Int 13h Fn 08h. These are “ill-behaved” applications. Ill-behaved applications fall into two categories: some of them read the Int 41h data and then use the conventional Int 13h interface for accessing the device. These are “compatible” ill-behaved applications. The remaining ill-behaved applications read the Int 41h/46h data and then access the drive in a proprietary manner. These are incompatible ill-behaved applications.

4.3.1 Compatible Ill-Behaved Applications

Compatible ill-behaved applications require that address 0, 2, and 14 (Cylinder, Head, and Sector) information in the FDPT be identical to the information returned in Int 13h Fn 08h. This class of application normally fails to call Int 13h Fn 08h to get device geometry, but uses Int 13h Fn 02h to read data.

4.3.2 Incompatible Ill-Behaved Applications

Incompatible ill-behaved applications require that address 0, 2, and 14 information have the geometry returned by ID drive data words 1, 3, and 6, a requirement that can violate restrictions placed on Standard FDPTs. Further, these incompatible ill-behaved applications may not check for the Translated FDPT signature (A0h at byte 3). Examples of incompatible ill-behaved applications are SCO Unix and early versions of Novell Netware.

4.3.3 Resolving the Compatibility Problem

The BIOS can only serve one class of these ill-behaved applications each time the system boots. This presents the BIOS and the USER with a compatibility problem. Phoenix has chosen to add a Setup field which allows the user to select which ill-behaved applications will function correctly. The menu item reads “Large Disk Access Mode”. This field defaults to “DOS”, which creates a Translated FDPT. Compatible ill-behaved applications will operate correctly when “DOS” is selected.

The remaining selection for Large Disk Access Mode is “OTHER”. Incompatible ill-behaved applications will function correctly with “OTHER,” which creates a Standard FDPT.

Because this format uses only physical geometries, “OTHER” creates problems for the compatible ill-behaved applications by generating an illegal Standard FDPT with more than 1024 cylinders. *The conventional Int 13h interface, however, continues to use a Translated FDPT, which is maintained internally by the BIOS, and is accessable only through Int 13h Fn 08h*. ***SETUP never changes the method of translation used by the BIOS.*** Well behaved DOS and Windows applications continue to function normally because they only use Int 13h Fn 08h, which returns translated geometry.

5. Index