X3T10 95-321
Rev 2

PC Boot Considerations for Devices >8GB

This is a draft of a document proposed in the System Issues Study Group meeting held on 7/12/95 in Colorado Springs. It is intended to be an informative document which will be issued, when approved, as an X3T10 Technical Report.

Please address comments to:
Duncan Penman
IIX Consulting
penman@netcom.com
(408) 730-2565

Overview

The PC has some historical firmware and software limitations on the size of disk that can be addressed. The major ones are:

528MB Affects IDE drives only. Caused by interaction of 16 heads maximum (limited by disk register size) and 1024 cylinders maximum (a limit at the OS-to-BIOS interface).

2GB Independent of drive type. A file system limit for any OS using a FAT based (DOS/Windows type) file system.

8GB Affects any drive accessed through the BIOS INT13H interface. This limit is a function of the number of bits available at the BIOS interface to address the drive.

In the past 2 years the PC community has painfully overcome the 528MB barrier. Two parallel solutions have emerged:

1. A standard method for address translation within the BIOS which supports addressing up to 8GB on a disk.
2. Direct disk support built into the OS, such that it does not use the BIOS interface to access disk storage. Using this approach, the 8GB barrier does not exist.

The latter solution has been implemented in all recent PC operating systems and appears to be the standard approach for the future.

Unfortunately, the OS must still be booted from a disk through the BIOS interface. Until the OS is in memory and initialized, there is no support for bypassing the BIOS. It is very desirable that the BIOS be able to address the full capacity of the boot disk, even if that extends beyond 8GB.

The rest of this document provides more background on this situation and recommends that the INT 13H extensions as documented in the Enhanced Disk Drive (EDD) 1.1 specification be adopted as the standard for the BIOS interface used to access drives with >8GB capacity.

Informative Background

A. Booting From an IDE Drive

When power is applied to a PC, it begins executing self test and initialization code, usually called POST (Power On Self Test), from nonvolatile memory that is resident on the motherboard. Once the motherboard resources, including add-on I/O adapters, have been initialized, POST makes an INT 19H call to the BIOS boot routine.

Current BIOSes allow booting from the first floppy (the A drive) or the first hard disk (the C drive). The user is generally allowed to configure which drive, A or C, is tried first as a boot device. There is also a mechanism implemented in some BIOSes for booting from a CD-ROM.

The BIOS boot routine reads the Boot Sector from the hard disk. This contains a small loader program that brings in the first part of the operating system. For details on this portion of the process, see Hale Landis' writeups on disk partitions and OS2 and Windows Boot Sector formats. These can be retrieved by ftp access from fission.dt.wdc.com under /pub/otherdocs/.... When finished, the boot routine passes control to the loader program.

The loader uses the INT 13H interface to read a portion of the operating system into memory. That portion of the OS then takes control and brings in the rest of the load image, typically still using the INT 13H interface in this phase. Figure 1 illustrates this. It is in these two steps that the addressability of the boot disk through the BIOS is a concern, even though the operating system may bypass the BIOS later and access the disk directly. The reason for the concern is that it is undesirable to have to place restrictions on where the files referenced at boot time are placed on the hard disk.

Throughout this process, the drive is unaware that this is a boot operation. It simply sees a series of read commands.

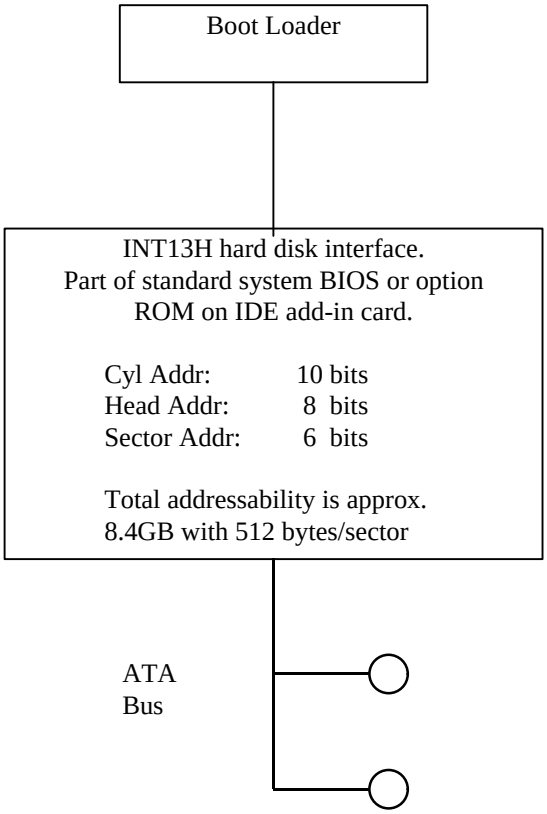

Figure 1: Basic IDE Boot Interface

B. Booting From a SCSI Drive

If the C drive has a SCSI interface, rather than IDE, the process is identical except for one of the steps in POST.

During POST, a reserved range of memory addresses in the first megabyte of storage is scanned to see if any option ROMs are present. When an option ROM is found, control is temporarily passed to its code in order to allow for product specific initialization. In order for a SCSI drive to be the boot device, its host adapter must have an associated option ROM. One of its initialization functions will be to replace the INT 13H entry address in the Interrupt Vector Table (IVT) with its own entry address. This is commonly called 'hooking INT 13'. Figure 2 illustrates this.

As a result of this initialization, any calls to the INT 13H interface will be first directed to the SCSI code in the option ROM (shadowed into RAM storage). If the disk being addressed is a SCSI drive, the INT 13H request parameters are translated into a SCSI CDB and issued to the target drive. If the request is for an IDE drive, then the SCSI option code simply filters the request and passes control to the address it previously

overlaid at the INT 13H location of the IVT. Using this mechanism, the boot process and any runtime software that uses the INT 13H interface can be indifferent to whether a drive is IDE or SCSI. It will treat it as if it were IDE in either case.

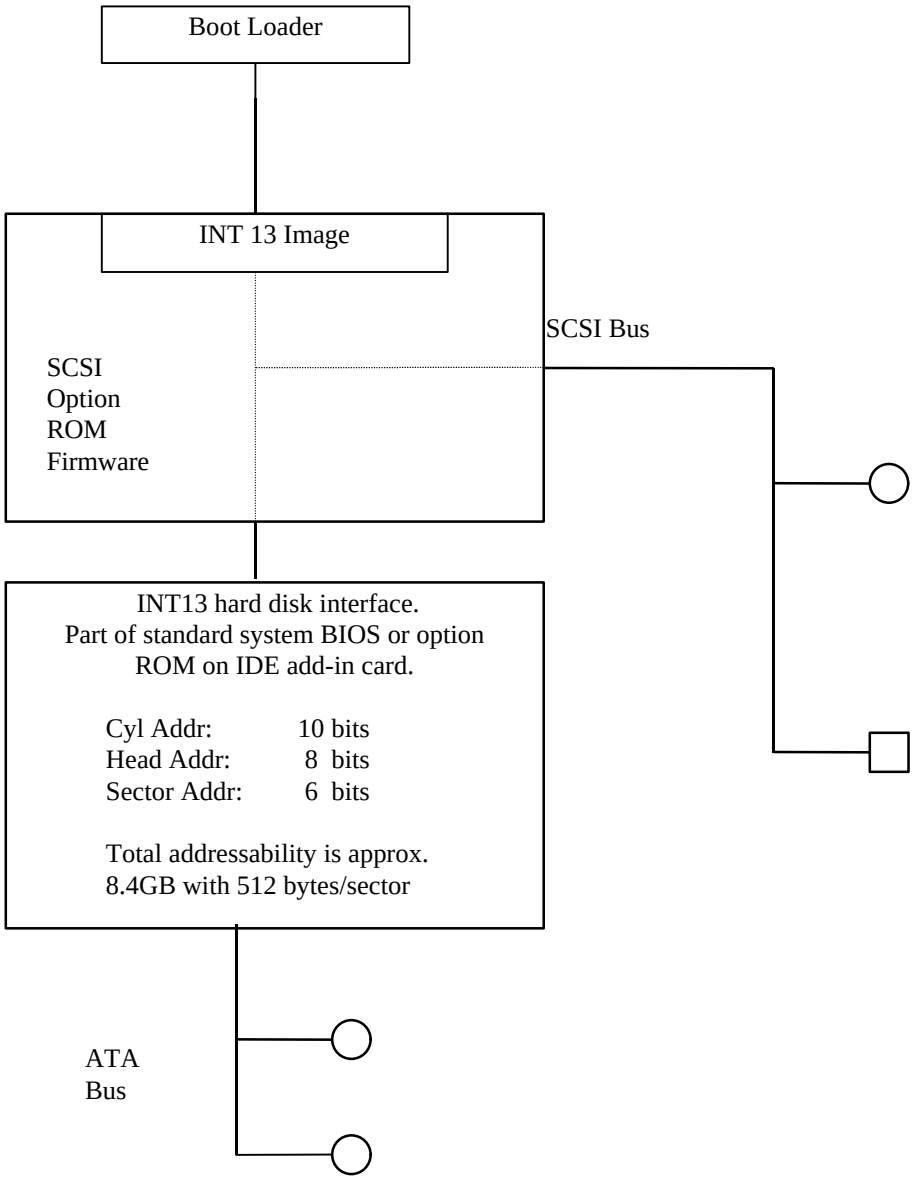

Figure 2: SCSI Boot Interface

C. The 8GB Barrier

The INT 13H interface has only enough bits to address approximately 8.4GB of storage on a disk. This is referred to as the 8GB barrier. There are already SCSI disks with more capacity than that, and we can expect to see IDE drives exceeding that limit in the future. In order to access the full capacity of of a disk larger than this through the BIOS, the INT 13H interface must be enhanced.

D. INT 13H Extensions

In 1993 IBM and Microsoft developed a set of INT 13H extended functions that, among other things, provided a way to address the full capacity of disks bigger than 8GB.
In 1994 Phoenix Technologies published the Enhanced Disk Drive (EDD) BIOS interface specification, which included a more complete description of this interface than the original specification from IBM and Microsoft.

The basic INT 13H interface uses registers to pass request parameters to the BIOS. These are in the form of cylinder, head, and sector values for the data location on the disk. In contrast, the extended interface places the request parameters in a data structure in memory. A pointer to this structure is passed in a pair of registers to the BIOS.

The data structure for an extended INT 13H request contains a 64 bit field for a logical block address (LBA). This is more than adequate for the next several years, as SCSI is currently limited to a maximum LBA size of 32 bits and IDE to 28 bits. See Figure 3.

Recommendation For Boot Support

The recommended approach for supporting boot devices larger than 8GB is to implement the INT 13H extensions as published in EDD 1.1. For operation after the boot processing is complete, the OS can continue to use the extended INT 13H interface or can use a device driver that bypasses the BIOS.

Effect On Existing Standards

No changes are required in SCSI to support drives larger than 8GB until we reach a point where we want to use an LBA larger than 32 bits. For a block size of 512 bytes, this means a drive with a capacity greater than 2 terabytes. There are no current plans to expand SCSI LBAs beyond 32 bits, so this document does not affect any SCSI-3 standards.

The ATA-4 standard, when it comes into existence, will need to describe IDENTIFY DEVICE data for drives larger than 8GB. This should probably be an annex similar to

the one in ATA-2 and ATA-3 describing IDENTIFY DEVICE data for drives less than 8GB.

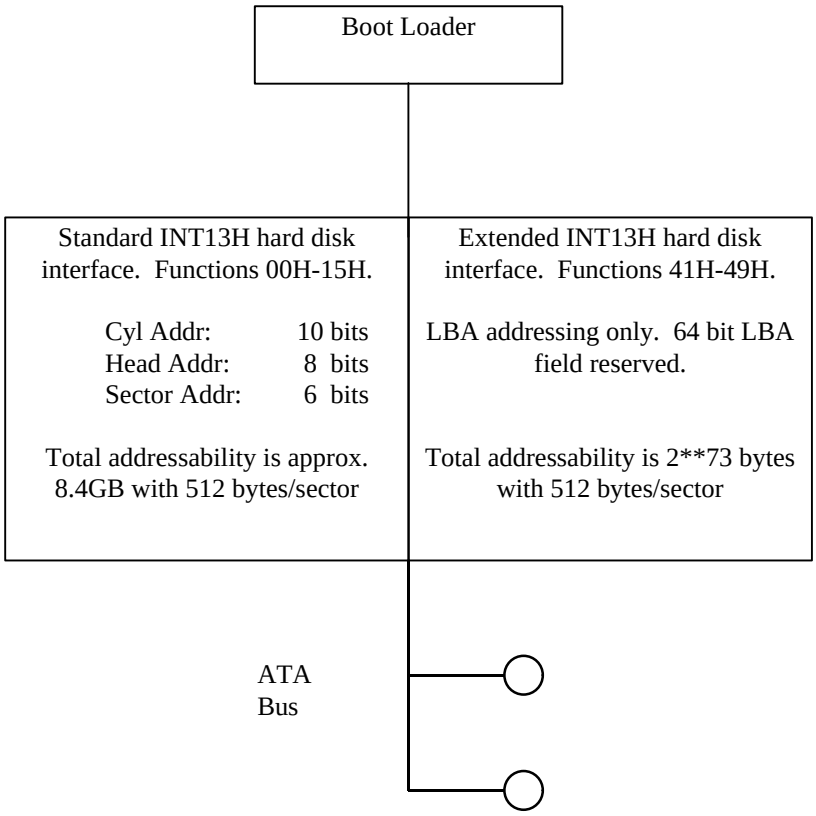

Figure 3: INT13H Extensions Boot Interface