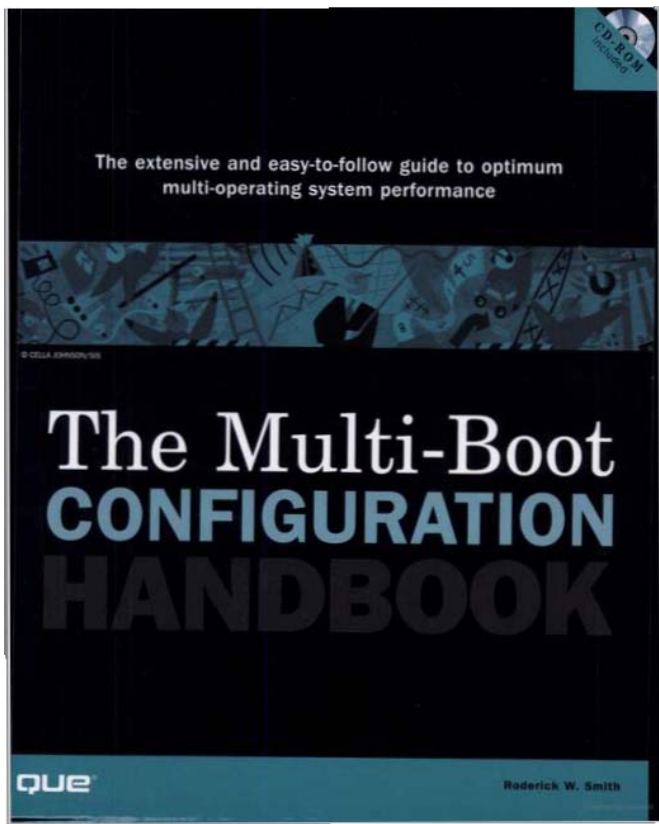




THE
Multi-Boot
CONFIGURATION
HANDBOOK

Roderick Smith

QUE[®]

A Division of Macmillan USA
201 West 103rd Street, Indianapolis, Indiana 46290

The Multi-Boot Configuration Handbook

Copyright © 2000 by Que® Corporation

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

International Standard Book Number: 0-7897-2283-6

Library of Congress Catalog Card Number: 99-068221

Printed in the United States of America

First Printing: April, 2000

02 01 00 4 3 2

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Que Corporation cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The author and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the CD or programs accompanying it.

ASSOCIATE PUBLISHER

Jim Minatel

SENIOR ACQUISITIONS EDITOR

Jenny L. Watson

DEVELOPMENT EDITORS

Todd Brakke

Rick Kugben

MANAGING EDITOR

Matt Parcell

PROJECT EDITOR

Tonya Simpson

COPY EDITOR

Michael Dietsch

INDEXER

Mary SeRine

PROOFREADER

Benjamin Berg

TECHNICAL EDITOR

Dan Maker, Keylabs

Tom McHugh

TEAM COORDINATOR

Vicki Harding

MEDIA DEVELOPER

Michael Hunter

INTERIOR DESIGNER

Anne Jones

COVER DESIGNER

Anne Jones

COPYWRITER

Eric Borgert

EDITORIAL ASSISTANT

Angela Boley

PRODUCTION

Tim Osborn

Mark Walchle

If you find that an OS has difficulties locating hardware or memory, you might want to try adjusting CMOS settings. The following are some options you might want to investigate:

- **Memory holes**—BIOSs sometimes place a “hole” in memory at 16MB or 64MB. Some OSs (especially OS/2) sometimes misinterpret this hole as the end of memory, so they don’t use all the RAM in the computer. Disabling this hole can recover this lost RAM.
- **PnP settings**—When using a PnP-aware OS, it’s generally best to let the OS configure PnP devices. Sometimes you might want to make the BIOS do the configuration, though. BIOSs provide varying options for how they assign system resources to PnP devices. You’ll have to experiment if you have problems with resource conflicts in a new OS. Be sure to test existing OSs after making changes, too!
- **APM settings**—Some OSs have problems dealing with APM features, so you might need to disable these in the BIOS.
- **Hard disk geometry**—You might need to adjust your hard disk geometry settings to get all your OSs to coexist on one drive. I discuss this feature in more detail later in this chapter.

What BIOS Do You Have?

Up to this point, I’ve referred to *the BIOS*, as if there were only one BIOS used, albeit with modifications, in all computers. In fact, there are several. Common BIOSs in use today include the following:

- **AMI BIOS**—American Megatrends, Inc. (AMI; <http://www.ami.com>) produces a popular BIOS that’s used on its own and many other companies’ motherboards.
- **Phoenix BIOS**—The Phoenix BIOS is another popular one. It closely resembles the AMI BIOS in its layout and options. For more information, see <http://www.phoenix.com>.
- **Award BIOS**—Phoenix purchased Award, so Award BIOSs are being shipped only on systems that were designed prior to the merger. Many existing Award BIOSs exist, however. The Award BIOS was somewhat different in its overall layout than the AMI and Phoenix BIOSs, but comparable in features. As I write this, the Award Web site still exists at <http://www.award.com>.
- **MR BIOS**—Microid Research (<http://www.mrbios.com>) sells after-market BIOS upgrades, mostly for older computer systems. These upgrades can be a useful way to add modern BIOS features to an older computer.

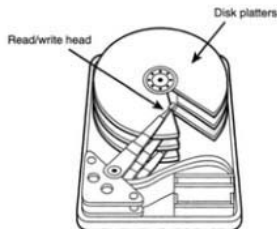
Each of these BIOSs can be considered more of a BIOS family than a single product. Each BIOS must be fine-tuned for each type of computer. For example, a BIOS for an Intel Pentium III motherboard must be different from an AMD Athlon motherboard’s BIOS. Even within a CPU family, BIOSs differ because of differences in the motherboard *chipsets*—the collection of circuitry that mediates between the CPU and the memory, add-in cards, and on-board

Understanding CHS Geometry Limits

The basic design of hard disks hasn't changed much since the early 1980s, although today's models are orders of magnitude higher in capacity and faster than those from two decades ago. The basic design of a hard disk is illustrated in Figure 3.3. Data are stored on magnetic coatings that cover one or more circular disk platters. To read or write data, the disk platters spin at high velocity (5400 to 10,000 rpm in typical drives today) and the read/write heads read the data off the disks. When a head remains stationary, it reads data from a ring-shaped portion of a platter. Because the platters are stacked on top one another, these ring-shaped areas themselves form a virtual cylinder.

FIGURE 3.3

Data on a hard disk can be addressed by cylinder (distance from the center), head (which platter and side), and sector (position within a cylinder).



Hard disks read and write data in discrete units of information, typically 512 bytes. Each such unit is known as a *sector*. On older drives, there were a fixed number of sectors per cylinder. Thus, it was natural to describe the layout of a hard disk in terms of three numbers: The cylinder (sometimes referred to as the *track*), the head, and the sector within a cylinder. This triplet of numbers is often referred to as the *CHS address*, as in cylinder 45, head 2, sector 11. On the simple hard drives of the early 1980s, using CHS addressing meant that the hard drives could use simple electronics; they did not need to decode some other addressing scheme into those three numbers. Both the PC's BIOS and assorted hard disk structures therefore express locations of data on a disk in terms of CHS addresses.

The CHS system is convenient for some very old drives, but most drives today, and even in the early 1990s, used a variable number of sectors per cylinder. This design allows more data to fit onto a disk because all other things being equal, more sectors can fit on outside cylinders than inside ones. Thus, IDE and EIDE hard drives actually lie to the computer about their geometries—they make up geometries that allow for the correct data capacity and then quietly translate between their true capacities and what they tell the BIOS. Still, you can pretend that what an IDE drive reports to the BIOS is correct because the limits I'm about to discuss reside mostly in the BIOS, or at least in the communication between the hard disk and the computer.

Unfortunately, the PC's designers, in a day when 10MB hard disks were huge, didn't plan for hard disks as large as we have today. These programmers allocated 10 bits for the cylinder number (mirroring the size of a register in an early PC hard disk controller), 8 bits for the head number, and 6 bits for the sector number. There were two further limitations, though: First, early IDE drives themselves had limits (16, 4, and 8 bits, respectively), which effectively limited the head field to 4 bits. Second, although the sector number is 6 bits, it is numbered starting from 1 rather than 0, so one sector number is effectively lost (it's not lost from the disk itself; there's simply one fewer sector available per cylinder in the numbering scheme). Thus, there was an effective limit of $1024 (2^{10})$ cylinders, $16 (2^4)$ heads, and $63 (2^6 - 1)$ sectors per cylinder. Multiply those values together, multiply the result by 512 bytes per sector, and convert to megabytes and you find that the limit on IDE hard drive size is 504MB.

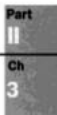
NOTE In general, a megabyte in computer circles is 2^{20} or 1,048,576 bytes. Hard drive manufacturers, however, tend to define a megabyte as 1,000,000 bytes, thus slightly inflating the apparent size of a disk for advertising purposes. (Similar differences exist for definitions of a kilobyte and a gigabyte.) I use the former definition exclusively throughout this book, but you should be aware of the difference. If you buy a hard disk that's advertised as being an 18.2GB unit, don't be surprised to see your disk partitioning software report it as 17.0GB. The disk manufacturer hasn't made a mistake, nor is this a difference between "formatted" and "unformatted" capacities, as some people seem to think; it's merely an artifact of differing definitions of what constitutes a kilobyte, a megabyte, or a gigabyte. Because of this difference, the 504MB limit is sometimes referred to as the 528MB limit. Some people mistakenly regularize the value as 512MB, which is inaccurate by any measure. ■

SCSI hard disks work somewhat differently. Rather than use a CHS value, SCSI has always used a single linear address space. To interface with a PC's BIOS, though, SCSI host adapters translate the SCSI linear addressing mode into CHS values. Details differ from one SCSI host adapter to another, but most use five bits for the number of heads, and some begin numbering sectors from 0, leading to a 1GB capacity limit. Most adapters today provide options to raise these limits further, as described shortly.

Getting Around the 1024-Cylinder Limit

By 1994, hard drives exceeding 504MB, or even 1GB, were becoming not only desirable, but common. Drives of higher capacity typically reported themselves as having the maximum number of heads and sectors and more than 1024 cylinders. Thus, this problem came to be referred to as the *1024-cylinder limit*, and it's still with us today.

The question in 1994 was one of circumventing this limit. Some of the methods I describe later in this chapter were useful in raising the limit, but by 1997 drives exceeding even these raised limits started to become common. Although many modern OSs aren't bothered by the 1024-cylinder limit, and can in fact handle much larger hard disks, portions of the modern BIOS boot code are still plagued by this limit, and this fact has important consequences for OS installation, as discussed in this and the next few chapters.



Ignorance Is Bliss One method of getting around the 1024-cylinder limit is to ignore it. This is a perfectly workable solution for OSs that don't rely on the BIOS for disk access. OS/2 and Linux, for example, even in 1994, could access all of a 1GB or larger disk; these OSs simply use more than 10 bits to represent the number of cylinders on a disk, thus allowing direct access to the disk's full capacity. There are several problems with this solution, however:

- To boot an OS, the BIOS must be able to read certain OS files, so those files must fall below the 1024-cylinder limit. Ensuring that this will be the case can be a nuisance.
- If you're running a dual-boot system, some OSs might be able to ignore the 1024-cylinder limit and some might not. Those that can't ignore it might become confused by a disk partitioned by the more capable OS or might become confused by the contents of a partition that spans the limits. For this reason, OS/2 won't format a partition that falls even partly past the 1024-cylinder limit for the file allocation table (FAT) filesystem used by DOS, although OS/2 will use its own high-performance filesystem (HPFS) on such partitions.
- Even two OSs that can each handle over-1024-cylinder drives might have subtle incompatibilities in how they encode partitioning or other data, so they might have problems coexisting on such a system.

Using CHS Geometry Translation One method of handling over-1024-cylinder drives was what came to be known as *CHS geometry translation* or various truncations of that term. Recall that the reason for using only four bits for the head number was that this was a limitation of IDE drives themselves. Internal to the BIOS, an eight-bit field could be used for the head number. CHS geometry translation typically works by taking excess cylinders and turning them into heads. For example, if a drive reports that it has 2000 cylinders and 16 heads, the BIOS can translate this to 1000 cylinders and 32 heads. The total drive capacity remains the same, the BIOS and most software can handle this system, and the total capacity limit is raised to 7.875GB, depending on how vigorously the software translates geometries.

CHS geometry translation was first done using special software BIOS extensions that were stored in the hard disks' boot sectors. Such software programs were frequently designed for DOS and didn't work well with OS/2, Linux, or other OSs. These OSs eventually learned to spot these programs and adjust their own handling of the drive to fit in. Therefore, if you have an old system that uses such software, you can probably continue to use it, even if you're planning to install a new OS. You might want to check the documentation for your new OS to be sure it's safe, though.

Eventually, CHS geometry translation came to be a standard feature in PC BIOSs, so no additional software was required. Similarly, many EIDE add-on controller cards include BIOS extensions that replace or augment the motherboard BIOS's hard disk routines to provide such support. As with software BIOS extensions, though, each OS must be able to play along with this solution. If an OS queries the hard disk directly about its capacity and uses the

values returned by the disk itself rather than by the BIOS, the OS will use incorrect values as far as other OSs are concerned.

CAUTION

It's vitally important that all OSs agree on a single CHS geometry for each drive. If two OSs don't agree, they'll write conflicting data in the partition table and can corrupt data structures all over the disk. The end result can be absolutely disastrous. Fortunately, problems like this are rare and are mostly related to old OSs, such as OS/2 2.1, that don't understand modern CHS translation schemes.

Note that CHS geometry translation is a solution applied to IDE and EIDE drives only. Because SCSI drives don't communicate to their host adapters using CHS values, there's little sense in applying such schemes to them. Instead, SCSI host adapters modified their methods of creating CHS values for attached hard drives, as described in the next section.

Using LBA Mode in EIDE and SCSI Drives *LBA stands for linear block addressing or logical block addressing*, depending on whom you ask. By either name, it refers to a system of hard disk access whereby a single sector number is used instead of a triplet of numbers. The sector number used in EIDE LBA mode is a 28-bit number (from IDE's 16-bit cylinder, 4-bit head number, and 8-bit sector number added together). Therefore, LBA mode supports hard drives of up to 128GB (2^{28} sectors $\times$ 512 bytes) in capacity.

SCSI hard drives have always used LBA mode internally and in talking to their host adapters; the SCSI host adapters have been the ones to translate LBA mode into CHS mode. Eventually, EIDE drives began to emulate SCSI drives in this respect. This method of addressing requires that the drive electronics support the change (as is true of most EIDE drives but not the older IDE drives), and it requires support in the BIOS. As with CHS geometry translation, LBA mode results in the capacity to handle hard disks of up to approximately 8GB in size when using old CHS addressing methods (which are still constrained by the BIOS's cylinder, head, and sector number limits), but software that can directly handle LBA mode can use disks of up to 128GB.

In general, LBA mode uses disk space a little more efficiently than does CHS geometry translation because less disk space is lost to rounding errors. Depending on the hardware in use, LBA mode can be slightly faster than CHS geometry translation, as well. Most modern OSs can use LBA mode directly (bypassing the BIOS), but some software, particularly low-level disk utilities, still uses CHS mode.