


Phoenix

Phoenix
Technical Reference Series

ABIOS for IBM® PS/2® Computers
and Compatibles

ABIOS for IBM® PS/2® Computers and Compatibles

The Complete Guide to
ROM-Based System Software for OS/2®


Addison
Wesley

51805

Phoenix Technologies Ltd.

Phoenix Technical Reference Series

*ABIOS for
IBM® PS/2®
Computers and
Compatibles*

*The Complete
Guide to
ROM-based
System
Software
for OS/2*

Phoenix Technologies Ltd.



Addison-Wesley Publishing Company, Inc.
Reading, Massachusetts Menlo Park, California New York
Don Mills, Ontario Wokingham, England Amsterdam Bonn
Sydney Singapore Tokyo Madrid San Juan

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book and Addison-Wesley was aware of a trademark claim, the designations have been printed in initial capital letters.

Limitation of Liability

While every reasonable precaution has been taken in the preparation of this book, the author and the publishers assume no responsibility for errors or omissions, or for the uses made of the material contained herein or the decisions based on such use. **No warranties are made, express or implied, with regard to the contents of this work, its merchantability, or fitness for a particular purpose.** Neither the author nor the publishers shall be liable for direct, special, incidental, or consequential damages arising out of the use or inability to use the contents of this book.

Library of Congress Cataloging-in-Publication Data

ABIOS for IBM PS/2 computers and compatibles : the complete guide to ROM-based system software for OS/2 / Phoenix Technologies, Ltd.

p. cm. -- (The Phoenix technical reference series)

Includes index.

ISBN 0-201-51805-8

1. IBM Personal System/2 (Computer system) 2. Systems software.
3. Read-only storage. 4. Computer input-output equipment.
I. Phoenix Technologies, Ltd. II. Title: BIOS. III. Series.
QA76.8.I25963A25 1989 005.4'469--dc 89-6449

Copyright © 1989, 1988, 1987 by Phoenix Technologies Ltd.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. Printed in the United States of America. Published simultaneously in Canada.

Cover design by Hannus Design Associates

Text design by Phoenix Technologies Ltd.

Set in 10-point Modern by Phoenix Technologies Ltd.

ABCDEFGHIJ-AL-89

First printing, June, 1989

Trademarks

This manual acknowledges the following trademarks:

Ashton-Tate and Framework are registered trademarks of Ashton-Tate Corporation.

AST is a registered trademark of AST Research, Inc.

AT, IBM, Personal Systems/2, PS/2 and PC/AT are registered trademarks of the International Business Machines Corporation. PC-DOS, PC/XT, VGA, CGA, MCA, EGA, MDA, OS/2 and Micro Channel are trademarks of the International Business Machines Corporation.

Intel is a registered trademark of Intel Corporation.

1-2-3, Lotus, and Symphony are registered trademarks of Lotus Development Corp.

Motorola is a registered trademark of Motorola Semiconductor Products, Inc.

MS, MS-DOS, and Microsoft are registered trademarks of the Microsoft Corporation.

NEC and Multisync are registered trademarks of Nippon Electric Corporation.

Quadram is a registered trademark of Quadram Corporation.

To Eric Enge, Stan Lyness, Paula Bishop, and the rest of the Phoenix Technologies Ltd. BIOS Engineering Department. Their tireless efforts have defined compatibility for the PC industry.

Table of Contents

About This Book

About This Book	xvii
-----------------------	------

Acronyms and Abbreviations

Acronyms and Abbreviations	xxi
----------------------------------	-----

Chapter 1 — The BIOS

Overview	1
BIOS Device Support	4
The BIOS Processing Model	5
BIOS Data Structures	10
BIOS Initialization Facts	12
Request Block Initialization	15
Transfer Conventions	17
BIOS and Program Access	19
Accessing BIOS via BIOSCommonEntry	20
Accessing BIOS via BIOSCall	23
Return Code Handling	24
BIOS Extensions	26
Where to Find More Information	28

Chapter 2 — Hardware Environment

Overview	29
80286/80386/80386SX Microprocessors	31
Math Coprocessors	32
Micro Channel	33
I/O Devices: Introduction	35
I/O Devices: Diskette and Disk Hardware	35
I/O Devices: Video Hardware	36
I/O Devices: Keyboard Hardware	38
I/O Devices: Parallel Port Hardware	39

continued

Chapter 2 — Hardware Environment, Continued

I/O Devices: Serial Port Hardware	40
System Time-Related Devices: Introduction	42
System Time-Related Devices: 82284 Clock Generator	42
System Time-Related Devices: 8254A PIT Chip	43
System Time-Related Devices: MC146818A RTC Chip	45
CMOS RAM Service	46
DMA Controller	47
Programmable Option Select (POS)	55
Intel 8259A Programmable Interrupt Controllers	56
System Control Port Definitions	58
Power-On Password	60
NMI Mask	60
Hardware I/O Port List	61

Chapter 3 — BIOS Data Structures

Overview	81
Common Data Area	84
Function Transfer Table	87
Device Block	89
Related Information	94

Chapter 4 — BIOS Initialization

Overview	95
Step 1: Build the System Parameters Table	98
Step 2: Build BIOS Initialization Table	100
Step 3: Build the Common Data Area	103
Step 4: Initialize Device Blocks and Function Transfer Tables	106
Step 5: Build Protected Mode Tables	110
Initializing Logical ID 2: BIOS Internal Calls	113
How BIOS Supports Multiple Instances of a Device	115
Related Information	117

continued

Chapter 5 — Request Block Structure

Overview	119
Request Block Parameters	121
Request Block Structure	122
Related Information	125

Chapter 6 — Calling BIOS

Overview	127
BIOS Processing Model	128
Request Block Initialization	132
A Generalized Look at Control Transfer	134
BIOS Transfer Convention	136
Operating System Transfer Convention	137
Return Code Handling	139
Hardware Interrupt Handlers	142
Default Interrupt Handler	147
Time-Out Handlers	149
BIOS and Program Access	151
Accessing BIOS via BIOSCommonEntry	152
Accessing BIOS via BIOSCall	155

Chapter 7 — BIOS Extensions

Overview	157
Recommendations for Extending BIOS	162
Requirement 1: Create Proper Extension Header	168
Requirement 2: Build Initialization Table Entry Routine	171
Requirement 3: Routine to Build Device Blocks and FTT	173
Requirement 4: BIOS Service Code	176
Initialization: BIOS ROM Extensions	176
Initialization: BIOS RAM Extensions	178
Examples of How to Modify an Existing Service	180
Example 1: Non-Intrusive Interception	181
Example 2: Redirection of a Nonstaged Function	184
Example 3: Redirection of a Staged Function	186

continued

Chapter 8 — BIOS Diskette Service

Overview	189
Hardware Environment	193
Error Handling	195
Function: 00h — Default Interrupt Handler	196
Function: 01h — Return Logical ID Parameters	197
Function: 03h — Read Device Parameters	199
Function: 04h — Set Device Parameters	202
Function: 05h — Reset/Initialize Diskette Subsystem	203
Function: 07h — Disable Diskette	206
Function: 08h — Read Diskette	208
Function: 09h — Write to Diskette	211
Function: 0Ah — Format Diskette	214
Function: 0Bh — Verify Diskette Sectors	218
Function: 0Ch — Read Media Parameters	221
Function: 0Dh — Set Media Type for Format	223
Function: 0Eh — Read Change Line Signal Status	226
Function: 0Fh — Turn Diskette Motor Off	228
Function: 10h — Interrupt Status	229

Chapter 9 — BIOS Fixed Disk Service

Overview	231
Hardware Environment	235
Fixed Disk Service Parameters Table	236
Error Handling	238
Function: 00h — Default Interrupt Handler	239
Function: 01h — Return Logical ID Parameters	240
Function: 03h — Read Device Parameters	242
Function: 05h — Reset/Initialize Fixed Disk	244
Function: 08h — Read Fixed Disk	249
Function: 09h — Write to Fixed Disk	251
Function: 0Ah — Write and Verify Fixed Disk	253

Chapter 10 — BIOS Keyboard Service

Overview	259
Hardware Environment	261

continued

Chapter 10 — BIOS Keyboard Service, Continued

101-Key Keyboard Layout	262
Scan Codes	263
System Scan Codes	264
Error Handling	268
Function: 00h — Default Interrupt Handler	269
Function: 01h — Return Logical ID Parameters	270
Function 03h — Read Keyboard ID Bytes	272
Function: 05h — Reset/Initialize Keyboard	274
Function: 06h — Enable Keyboard	276
Function: 07h — Disable Keyboard	278
Function: 08h — Continuous Keyboard Read	280
Function: 0Bh — Read Keyboard LED Status	282
Function: 0Ch — Set Keyboard LED Status	284
Function: 0Dh — Set Typematic Rate and Delay	286
Function 0Eh — Read Keyboard Scan Code Mode	289
Function 10h — Write Command(s) to Keyboard Controller	295
Function 11h — Write Command(s) and Data to Keyboard	300

Chapter 11 — BIOS Video Service

Overview	305
Hardware Environment	307
Video Modes	309
Mode/Monitor Support	311
ROM-Resident Fonts	313
Error Handling	315
Function: 00h — Default Interrupt Handler	316
Function: 01h — Return Logical ID Parameters	317
Function: 03h — Read Device Parameters	319
Function: 05h — Set Video Mode	324
Function: 0Bh — Return ROM Fonts Information	331
Function: 0Ch — Save Video Environment	333
Function: 0Dh — Restore Video Environment	335
Function: 0Eh — Select Character Generator Block	337
Function: 0Fh — Load Text Mode Font	339
Function: 10h — Enhanced Load Text Mode Font	342
Function: 11h — Read Palette Register	345
Function: 12h — Write Palette Register	347

continued

Chapter 11 — BIOS Video Service, Continued

Function: 13h — Read DAC Color Register	349
Function: 14h — Write DAC Color Register	351
Function: 15h — Read Block of Color Registers	354
Function: 16h — Write Block of DAC Color Registers	356

Chapter 12 — BIOS Serial Communications Service

Overview	359
Hardware Environment	361
Error Handling	363
Function: 00h — Default Interrupt Handler	364
Function: 01h — Return Logical ID Parameters	365
Function: 03h — Read Device Parameters	367
Function: 05h — Reset/Initialize Serial Port	370
Function: 0Bh — Set Modem Control	374
Function: 0Ch — Set Line Control	375
Function: 0Dh — Set Baud Rate	378
Function: 0Eh — Transmit	379
Function: 0Fh — Receive	389
Function: 10h — Transmit and Receive	399
Function: 11h — Modem Status	404
Function: 12h — Cancel	407
Function: 13h — Return Line Status	409
Function: 14h — Return Modem Status	410
Function: 15h — Enable FIFO Control	412

Chapter 13 — BIOS Parallel Port Service

Overview	415
Hardware Environment	417
Error Handling	418
Function: 00h — Default Interrupt Handler	419
Function: 01h — Return Logical ID Parameters	420
Function: 03h — Read Device Parameters	421
Function: 04h — Set Device Parameters	423
Function: 05h — Reset/Initialize Parallel Port	425
Function: 09h — Print Block	427

continued

Chapter 13 — BIOS Parallel Port Service, Continued

Function: 0Bh — Cancel Print Block	430
Function: 0Ch — Return Printer Status	432

Chapter 14 — BIOS System Timer Service

Overview	435
Hardware Environment	436
Error Handling	437
Function: 00h — Default Interrupt Handler	438
Function: 01h — Return Logical ID Parameters	439

Chapter 15 — BIOS Real Time Clock Service

Overview	441
Hardware Environment	443
Real Time Clock Data	444
Error Handling	445
Function: 00h — Default Interrupt Handler	446
Function: 01h — Return Logical ID Parameters	447
Function: 03h — Read Device Parameters	449
Function: 04h — Set Device Parameters	452
Function: 0Bh — Set Alarm Interrupt	454
Function: 0Ch — Cancel Alarm Interrupt	457
Function: 0Dh — Set Periodic Interrupt	458
Function: 0Eh — Cancel Periodic Interrupt	461
Function: 0Fh — Set Update-Ended Interrupt	462
Function: 10h — Cancel Update-Ended Interrupt	464
Function: 11h — Read Time and Date	465
Function: 12h — Write Time and Date	466

Chapter 16 — BIOS System Services

Overview	469
Error Handling	470
Function: 01h — Return Logical ID Parameters	471
Function: 03h — Read System Configuration	473

continued

Chapter 16 — BIOS System Services, Continued

Function: 0Bh — Switch to Real Mode	475
Function: 0Ch — Switch to Protected Mode	478
Function: 0Dh — Enable Address Line 20	481
Function: 0Eh — Disable Address Line 20	482
Function: 0Fh — Enable Speaker	483

Chapter 17 — BIOS Nonmaskable Interrupt (NMI) Service

Overview	485
Error Handling	487
Function: 01h — Return Logical ID Parameters	488
Function: 06h — Enable NMI	490
Function: 07h — Disable NMI	491
Function: 08h — NMI Continuous Read	492

Chapter 18 — BIOS Pointing Device Service

Overview	495
Hardware Environment	497
Error Handling	497
Function: 00h — Default Interrupt Handler	498
Function: 01h — Return Logical ID Parameters	499
Function: 03h — Read Device Parameters	500
Function: 05h — Reset/Initialize Pointing Device	503
Function: 06h — Enable Pointing Device	505
Function: 07h — Disable Pointing Device	507
Function: 08h — Pointing Device Continuous Read	509
Function: 0Bh — Set Sample Rate	512
Function: 0Ch — Set Resolution	514
Function: 0Dh — Set Scaling Factor	516
Function: 0Eh — Read Pointing Device Identification Code	519

Chapter 19 — BIOS CMOS RAM Service

Overview	523
Hardware Environment	524

continued

Chapter 19 — BIOS CMOS RAM Service, Continued

CMOS RAM Data	525
Extended CMOS RAM Data	528
Error Handling	532
Function: 01h — Return Logical ID Parameters	533
Function: 03h — Read Device Parameters	534
Function: 08h — Read CMOS RAM	536
Function: 09h — Write to CMOS RAM	538
Function: 0Bh — Recompute Checksum	540

Chapter 20 — BIOS Direct Memory Access (DMA) Service

Overview	543
Hardware Environment	545
Error Handling	552
Function: 01h — Return Logical ID Parameters	553
Function: 03h — Read Device Parameters	555
Function: 0Bh — Allocate Arbitration Level	556
Function: 0Ch — Deallocate Arbitration Level	558
Function: 0Dh — Disable Arbitration Level	559
Function: 0Eh — DMA Transfer Status	560
Function: 0Fh — Abort DMA Operation	561
Function: 10h — DMA Transfer from Memory to I/O	563
Function: 11h — Read from I/O and Write to Memory	565
Function: 12h — Load DMA Controller Parameters	567

Chapter 21 — BIOS Programmable Option Select Service

Overview	569
Hardware Environment	571
Error Handling	573
Function: 01h — Return Logical ID Parameters	574
Function: 0Bh — Read Stored POS Data from CMOS RAM	575
Function: 0Ch — Write Stored POS Data from CMOS RAM	577
Function: 0Dh — Read POS Data from an Adapter	579
Function: 0Eh — Write Dynamic POS Data from an Adapter	581

continued

Chapter 22 — BIOS Keyboard Security Service

Overview	583
Hardware Environment	585
System Scan Codes	586
Error Handling	589
Function: 01h — Return Logical ID Parameters	590
Function: 03h — Read Device Parameters	592
Function: 06h — Enable Keyboard Security	593
Function: 0Bh — Write Password	595
Function: 0Ch — Write Invocation Byte	597
Function: 0Dh — Write Match Byte	599
Function: 0Eh — Write Filter Byte 1	601
Function: 0Fh — Write Filter Byte 2	603

Chapter 23 — BIOS Error Log Service

Overview	605
Extended CMOS RAM	607
Error Handling	609
Function: 01h — Return Logical ID Parameters	610
Function: 08h — Read Error Log	612
Function: 09h — Write to Error Log	614

Appendix A — BIOS Return Codes	617
--------------------------------------	-----

Glossary	627
----------------	-----

Additional Resources	639
----------------------------	-----

Index	641
-------------	-----

About This Book

What this book is about

ABIOS for IBM PS/2 Computers and Compatibles is a detailed technical reference that describes the ABIOS, the portion of the PS/2 ROM BIOS designed to support multitasking operating systems such as OS/2. The information provided in this book is applicable to all Micro Channel Architecture-based IBM PS/2 and compatible computers.

Who should read this book

This book can be used by anyone interested in learning more about PS/2 or compatible computers.

Programmers who wish to make direct calls to the ABIOS will find complete instructions for accessing the ABIOS via any version of OS/2 (e.g. IBM OS/2) that supports **DevHlp** services **ABIOSCommonEntry** and **ABIOSCall**.

Programmers writing device drivers for new peripheral devices will find the general information on ABIOS services helpful. They will also find, in the chapter on ABIOS extensions, a complete description of how ABIOS services can be added, replaced, or modified.

Implementers of operating systems and other multitasking system software and systems programmers will find all the information they need on ABIOS/operating system interfaces.

What we assume you know

This book assumes a basic knowledge of 80x86 assembly language programming concepts, PC architecture, and operating system concepts. If you are new to these subjects, use this book along with some of the excellent introductory books listed at the end of this book.

How to find information

Chapters 1–7 of *ABIOS for IBM PS/2 Computers and Compatibles* provide a general introduction to ABIOS concepts and describe how to use ABIOS services. Chapters 8–23 describe the individual ABIOS services and serve as a technical reference. The Appendix provides a comprehensive list of potential error messages and Return Codes. Most readers will want to read chapters 1–7 first to get a basic grasp of the ABIOS features before turning to the individual service descriptions.

Each service-specific chapter (8–23) is organized in a similar fashion. There is a description of the service and its hardware environment, a discussion on how errors are handled, and complete descriptions of each service function. The function descriptions include a description of what the function does, a graphic outline of the required Request Block structure and a list of Return Codes.

Other volumes in this series

ABIOS for IBM PS/2 Computers and Compatibles is one of several volumes about BIOS software in the Phoenix Technical Reference Series published by Addison-Wesley. Other volumes are:

CBIOS for IBM PS/2 Computers and Compatibles — a complete technical reference describing the portion of a PS/2 BIOS designed to support single-tasking operating systems such as MS-DOS.

System BIOS for IBM PC/XT/AT Computers and Compatibles — a complete technical reference for the BIOS in all standard architecture computers.

The BIOS volumes of the Phoenix Technical Reference Series provide the most comprehensive source of information about IBM and compatible system BIOSs available today. Each volume lists complete I/O port addresses, CMOS RAM, and BIOS data definitions. Every function is described in detail, and complete lists of error messages are provided.

The volumes of this series are a natural companion for anyone who owns and programs an IBM PC, XT, AT, or PS/2 model, or any compatible system.

No writing project as large as this can be completed without significant contributions from many individuals. We would like to acknowledge the Phoenix employees whose time and effort helped make this book possible.

First and foremost, we gratefully acknowledge the vision and technical skill of Neil Colvin, founder, CEO, and Chief Scientist of Phoenix Technologies Ltd. We also gratefully acknowledge the expert guidance of Lance Hansche, President of Phoenix Technologies Ltd. Without Neil and Lance's leadership, Phoenix would not be in the position of technological preeminence it now enjoys.

In the marketing area, Phoenix Vice President Rich Levandov's foresight and support made this book possible. And Product Marketing Manager Henry Suwinsky's tactful and dauntless guidance shepherded this project through all phases of its existence.

On the technical side, we acknowledge the Phoenix engineers who developed the Phoenix PS/2 BIOS. Eric Enge, Director of PC Product Engineering, Paula Bishop, and Stan Lyness have been endlessly patient with us and tireless in their efforts to ensure quality PS/2-compatible BIOS products and documentation. We also must acknowledge the efforts expended by other Phoenix engineers, including Bruce Cairns, Paul Chicoine, Greg Honsa, Suzanne Laferriere, Malcolm Pordes, Debbie Schultz, and Trevor Western.

In the production of this volume, many people contributed significantly. Chief among these were Kathy Schiff, Manager of Technical Communications, who provided editorial guidance and direction. Writers Marianne Adams and Dr. George Elliott Tucker contributed critical and essential writing, editing, and production assistance. And last but not least, a special thanks goes to Sandie Zierak, Chief Production Coordinator, for her invaluable contributions in the fields of graphics, document design, and document production.

The Authors

Mike Boston
Paul Narushoff

Phoenix Technologies Ltd.
Norwood, MA
March, 1989

Acronyms and Abbreviations

The following abbreviations and acronyms are used in this manual:

ASCII	American Standard Code for Information Interchange
ASIC	Application-Specific Integrated Circuit
b	Binary
BCD	Binary coded decimal
BIOS	Basic input/output system
bps	Bits per second
CDA	Common data area
CGA	Color graphics adapter
CRC	Cyclic redundancy check
CMOS	Complementary metal oxide semiconductor
DB	Device Block
DMA	Direct memory access
DSR	Device service routine
ECC	Error checking and correction
EGA	Enhanced graphics adapter
EOI	End of interrupt
ESDI	Enhanced small device interface
FTT	Function transfer table
h	Hexadecimal
INT	Interrupt
I/O	Input/Output
IRQ	Interrupt request line
ISR	Interrupt service routine
K	Kilobytes
Kbs	Kilobits per second
LID	Logical ID
LSB	Least significant byte
LSI	Large scale integration
M, MB	Megabytes
MDA	Monochrome Display Adapter
MFM	Modified frequency modulation
MHz	Megahertz

continued

Acronyms and Abbreviations, Continued

MSB	Most significant byte
NMI	Nonmaskable interrupt
OS/2	Operating System/2
PGA	Professional graphics adapter
POS	Programmable Option Select
POST	Power-on self test
PTL	Phoenix Technologies Ltd.
RAM	Random access memory
RB	Request block
RLL	Run length limited
ROM	Read-only memory
RTC	Real time clock
VGA	Video graphics array
VLSI	Very large scale integration

Chapter 1

The BIOS

Overview

What is the PS/2 BIOS?

The ROM BIOS contained in Micro Channel Architecture-based IBM PS/2 and compatible computers performs the same function that all basic input/output systems do: it isolates the operating system from direct manipulation of hardware registers, timings, and attachments.

When compared with the ROM BIOS contained in IBM PC XT/AT and compatible computers, however, the PS/2 ROM BIOS has one critical difference — it is designed to support two kinds of operating systems. As such, the PS/2 ROM BIOS is divided into two discrete parts: the BIOS and the CBIOS.

The CBIOS

The CBIOS part of the PS/2 ROM BIOS provides IBM PS/2 and compatible computers backward compatibility with single-tasking, Intel 80x86 real address mode operating systems such as PC-DOS or MS-DOS. As a result, the CBIOS consists of a superset of the services and functions available in the IBM PC/XT/AT ROM BIOS, and it interfaces with the operating system in the same well understood way.

continued

Overview, Continued

The ABIOS

The ABIOS part of the PS/2 ROM BIOS provides IBM PS/2 and compatible computers with forward compatibility with multitasking, bimodal (real mode, protected mode, or both) operating systems, such as IBM OS/2. The ABIOS supports the same hardware devices as the CBIOS, but its interface and data structures are specifically constructed to facilitate the multitasking, bimodal nature of its design.

Program accessibility

The program accessibility of any ROM BIOS depends on the architecture of the operating system interfaced with the BIOS.

■ ABIOS Accessibility

Starting with IBM OS/2 Version 1.1, programs have full access to the ABIOS through the operating system via the two IBM OS/2 **DevHlp** services: **ABIOSCommonEntry** and **ABIOSCall**. ABIOS accessibility for versions of OS/2 other than IBM's varies from vendor to vendor. Programmers who are not using IBM OS/2 should refer to their OS/2 documentation to determine if their version of OS/2 supports program access to the ABIOS.

■ CBIOS Accessibility

Under the totally open architecture of PC-DOS or MS-DOS, application programs interface directly with the CBIOS. In fact, many MS-DOS application programs attempt to improve performance by bypassing MS-DOS system services in favor of the more direct CBIOS services.

Scope of this document

This book describes the ABIOS component of the PS/2 BIOS. The information is 100 percent applicable to both the IBM and the Phoenix Technologies Ltd. versions of ABIOS.

continued

Overview, Continued

In this chapter

This chapter outlines the major concepts and design features of the BIOS. The following topics are discussed:

- BIOS Device Support
 - The BIOS Processing Model
 - BIOS Data Structures
 - BIOS Initialization Facts
 - Request Block Initialization
 - Transfer Conventions
 - BIOS and End User Access
 - Accessing BIOS via BIOSCommonEntry
 - Accessing BIOS via BIOSCall
 - Return Code Handling
 - BIOS Extensions
 - Where to Find More Information
-

For more information on the CBIOS

For a complete treatment of the CBIOS, see *CBIOS for IBM PS/2 Computers and Compatibles* in this series.

ABIOS Device Support

Introduction

The ABIOS occupies 64K of the 128K PS/2 BIOS. The ABIOS supports 16 kinds of physical devices. There is one ABIOS device service for each physical device.

ABIOS supported devices

The table below lists the physical devices supported by the ABIOS and the ABIOS device IDs assigned to them.

Device ID	Device Type/Service	Device ID	Device Type/Service
00h	ABIOS Internal Calls	0Bh	Pointing Device
01h	Diskette	0Ch	Reserved
02h	Fixed Disk	0Dh	Reserved
03h	Video	0Eh	CMOS RAM
04h	Keyboard	0Fh	Direct Memory Access
05h	Parallel Port	10h	Programmable Option Select
06h	Serial Port	11h	Error Log
07h	System Timer	12h-15h	Reserved
08h	Real Time Clock Timer	16h	Keyboard Security
09h	System Services	17h-FFFFh	Reserved
0Ah	Nonmaskable Interrupt		

Standard ABIOS functions

Functions numbered 00h-09h are standard functions across all ABIOS services. Functions 10h-xxh are tailored to the device being serviced.

Function	Description
00h	Default Interrupt Handler
01h	Return Logical ID Parameters
02h	Reserved
03h	Read Device Parameters
04h	Set Device Parameters
05h	Reset/Initialize
06h	Enable
07h	Disable
08h	Read
09h	Write

The BIOS Processing Model

Introduction

In a traditional PC-based ROM BIOS (such as the CBIOS), all functions are processed on the single-staged, call/process/return model. Once a function is invoked, the CPU is prevented from turning to other work until the function completes and returns. If the function called must interface with slower external hardware, the BIOS initiates a Wait and suspends CPU processing until the hardware interrupt occurs. The resulting amount of idle CPU time can be considerable.

Single-staged processing/single-tasking operating systems

Although this single-staged call/return method is somewhat inefficient, the fact that the BIOS is interfaced with a single-tasking operating system minimizes the method's impact on system throughput. A BIOS that processes functions exclusively on the single-staged model is therefore best suited for single-tasking operating systems.

Multistaged processing/multitasking operating systems

For a multitasking operating system to be interfaced with a BIOS, issues surrounding system throughput have to be handled more carefully than they are with a single-tasking operating system. The CPU must be free to process other tasks while a BIOS function is waiting for a hardware interrupt to occur. In order to do this, the BIOS must process functions in a way that minimizes BIOS control of CPU time. This is accomplished by processing function calls in multiple stages.

Where the A BIOS fits in


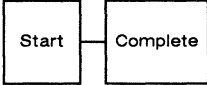
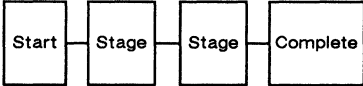
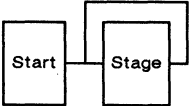
The A BIOS is written specifically for multitasking operating systems. As such, the A BIOS interface and the methods A BIOS employs to process function calls are specifically designed to minimize A BIOS control of processor time.

continued

The BIOS Processing Model, Continued

Processing models

The BIOS and CBIOS methods of processing function calls are contrasted below:

BIOS	CBIOS
<div style="text-align: center;">  </div> <p style="text-align: center;">SINGLE-STAGED FUNCTION</p> <p>When processing single-staged functions, the BIOS performs input/output immediately and returns to the operating system. BIOS requests that can be completed with a minimum of processor time are executed in this way.</p>	<div style="text-align: center;">  </div> <p style="text-align: center;">SINGLE-STAGED FUNCTION</p> <p>The CBIOS processes all function calls on the single-staged call/process/return model. Once a CBIOS function is invoked, the processor is prevented from turning to other work until the function completes and returns.</p>
<div style="text-align: center;">  </div> <p style="text-align: center;">DISCRETE MULTISTAGED FUNCTION</p> <p>BIOS functions that require a greater amount of processor time are processed in multiple stages.</p> <p>In discrete multistaged functions, the caller initiates input/output and returns to the operating system, moving from stage to stage until the function call is complete. Discrete multistaged functions are driven from stage to stage by an interrupt from the device being serviced or by the expiration of a function-requested time period.</p>	<p style="text-align: center;">DISCRETE MULTISTAGED FUNCTION</p> <p style="text-align: center;">NOT SUPPORTED</p>
<div style="text-align: center;">  </div> <p style="text-align: center;">CONTINUOUS MULTISTAGED FUNCTION</p> <p>When processing continuous multistaged functions, BIOS initiates input/output and returns to the operating system. Continuous multistaged functions can be considered "standing requests" in that they never reach a completion point. Like discrete multistaged functions, they are driven from stage to stage by a hardware interrupt at the expiration of a function-requested time period.</p>	<p style="text-align: center;">CONTINUOUS MULTISTAGED FUNCTION</p> <p style="text-align: center;">NOT SUPPORTED</p>

continued

The BIOS Processing Model, Continued

What drives multistaged functions

Multistaged functions are driven from stage to stage by either:

- hardware interrupt or
- elapse of a function requested period of time.

Some multistaged functions are driven purely by hardware interrupt. Others are driven purely by time period. Others still are driven by some combination of both hardware interrupt and time period.

Interrupt driven stages

Each interrupt-driven BIOS service is associated with one hardware interrupt level. The BIOS assumes that all hardware interrupt handlers are under the control of the operating system.

When a hardware interrupt occurs, the operating system must call the function associated with the interrupt so that the interrupting condition can be serviced. The BIOS resets the interrupting condition at the hardware level. The operating system's hardware interrupt handler must perform end-of-interrupt processing at the interrupt controller level.

A service can have more than one active function request. When this happens, the hardware interrupt handler calls each function until the BIOS replies that the hardware interrupt has been serviced.

Time period driven stages

Some BIOS functions are driven from stage to stage by the elapse of a function-requested period of time. The BIOS assumes that time-period stage handlers are under the control of the operating system. When the time period requested by the function expires, the operating system's time period handler must call the given BIOS function.

continued

The BIOS Processing Model, Continued

Hardware interrupt stages and hardware time-out

All hardware interrupt driven stages of a function indicate a maximum time (in seconds) to wait for the hardware interrupt. The BIOS assumes that all hardware time-out handlers are under the control of the operating system. Should the hardware time-out period associated with a given interrupt driven function elapse, the operating system must call the BIOS to terminate the function and reset the hardware.

Hardware time-out vs. time period stages

The terminology surrounding time-period driven stages and hardware time-out handling is similar. However, it is important not to confuse the processing associated with the hardware time-out handling and time-period stage handling.

Hardware time-out handling is associated exclusively with those stages of a multistaged function that are driven by hardware interrupt and is designed to handle function termination cleanly. Execution of a time-out handling routine is symptomatic of a hardware error.

Time-period handling is associated with those stages of a multistaged function that are driven by time periods. Execution of a time-period handling routine indicates the elapse of a function-requested time delay and should not be associated with a hardware error.

continued

The BIOS Processing Model, Continued

How handlers call BIOS functions: BIOS Entry Routines

Although we have mentioned that the various handlers under the control of the operating system must call BIOS functions, we have not mentioned how this is done: Each BIOS Service is associated with a set of function entry routines. There are three kinds of entry routines:

- **Start Routine**

The start routine associated with a service is called when a function is first started.

- **Interrupt Routine**

The Interrupt Routine associated with a service is called when the function interrupts or when a time period driven function requires servicing.

- **Time-out Routine**

The Time-out Routine associated with a service is called when a interrupt-driven function suffers a hardware time-out.

To reduce caller overhead, the BIOS also contains a set of Common Entry Routines: Common Start, Interrupt, and Time-out Routines. The Common Entry Routines do some initial processing then transfer control to the entry routine tied to the specific service.

ABIOS Data Structures

Introduction

The ABIOS makes use of four kinds of data structures. Three of the structures — the Function Transfer Table, the Device Block, and the Common Data Area are roughly analogous to the data structures found in the CBIOS.

The fourth structure is a function-specific, parameter block, called the Request Block. The Request Block has no data structure analog in the CBIOS, but it provides the ABIOS with the same parameter passing capability as is provided to the CBIOS by the processor's register set.

Each ABIOS data structure is defined further below.

ABIOS Data Structures	
Function Transfer Table	Function Transfer Table Each ABIOS Service is associated with one Function Transfer Table. Each Function Transfer Table contains a list of pointers to the Start, Interrupt, and Time-out Routine associated with its ABIOS service, as well as a list of vectors to the start of each function contained in that ABIOS Service.
Device Block	Device Block An ABIOS Service is usually associated with one Device Block. The Device Block contains the interrupt level, the arbitration level, and other information about the hardware device associated with a ABIOS device service. Some ABIOS services are associated with more than one physical instance of a device. When this is the case, that service is associated with more than one Device Block.
Common Data Area	Common Data Area The Common Data Area contains a list of pointers to the Function Transfer Table/Device Block pair associated with each ABIOS Service. Each pair of pointers in the table is identified by a unique Logical ID. Logical IDs are assigned dynamically when the ABIOS is initialized and are used by the caller as an index into the Common Data Area.
Request Block	Request Block Each ABIOS function is interfaced with the caller via a function-specific structure called the Request Block. All input and output parameters are passed between the caller and the ABIOS through the Request Block. Offset 0Ch of all Request Blocks is reserved for a function status indicator called the Return Code.

continued

ABIOS Data Structures, Continued

ABIOS initialization

The Common Data Area, all Function Transfer Tables, and all Device Blocks are initialized (with segment:offset pointers) as part of ABIOS real mode initialization. In bimodal operating environments, the operating system insures bimodal access to the ABIOS by initializing (in selector:offset format) a parallel Common Data Area and a parallel set of Function Transfer Tables.

Once they are initialized, the Common Data Area, the Function Transfer Tables, and the Device Blocks stay resident in system RAM for as long as the ABIOS is in use.

Request Block initialization

Before a given ABIOS function can be started, the caller must initialize its Request Block. The Request Block associated with a function call stays resident in system RAM for the life of the call. Request Blocks associated with completed function calls can be reused, or the memory they occupy can be reallocated for some other purpose.

Table: CBIOS/ABIOS analogs

End users familiar with CBIOS structures and conventions may find the table of analogies below useful.

Action	CBIOS	ABIOS
Pass Parameters	CPU Registers (bidirectional)	Request Block (bidirectional)
Identify BIOS service	Interrupt number	Logical ID
Locate BIOS service	Interrupt vector table	Logical ID index into Common Data Area
Locate BIOS function	Jump table internal to CBIOS code	Vector in Function Transfer Table

ABIOS Initialization Facts

Introduction

Initializing ABIOS is a matter of initializing ABIOS data structures that stay resident in system RAM for as long as the ABIOS is in use: The Common Data Area, the Function Transfer Tables, and the Device Blocks. ABIOS is initialized by the operating system in a five-step process involving both the CBIOS and the ABIOS.

Key facts to remember

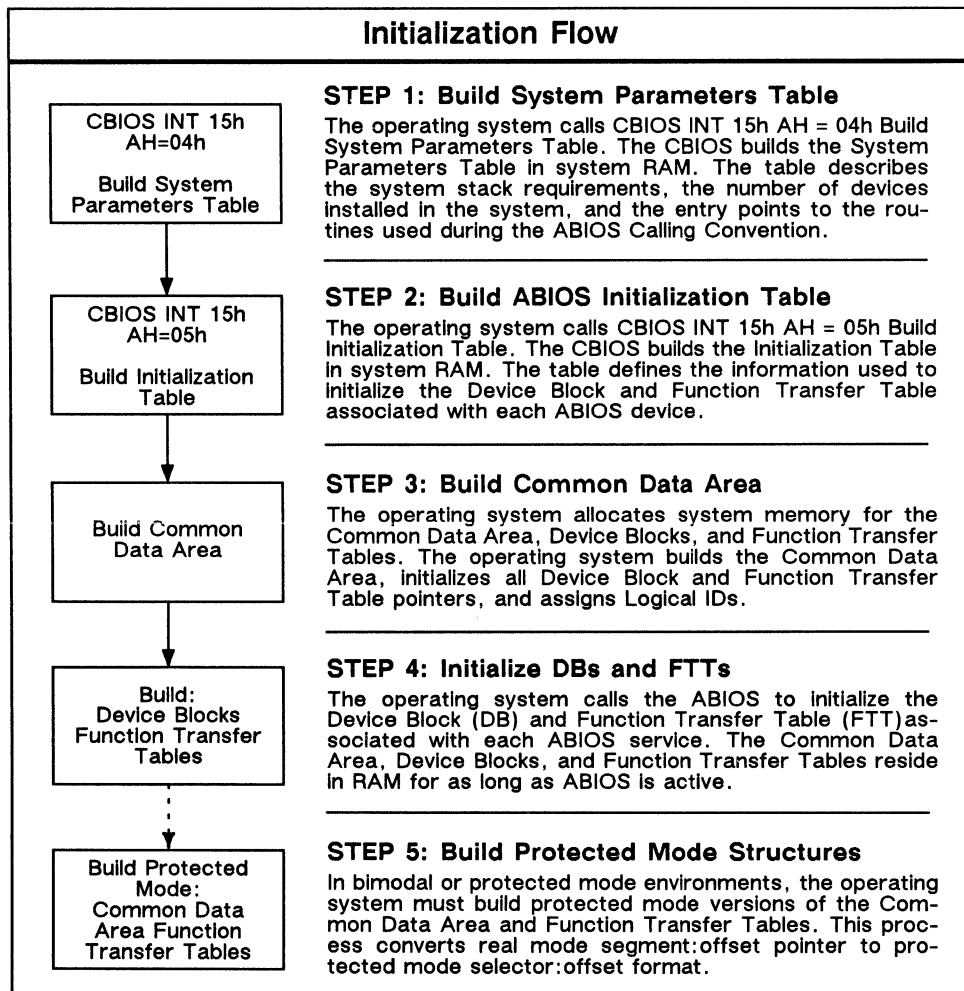
- Before ABIOS can be initialized, CBIOS must be initialized and the operating system must be booted.
 - ABIOS can only be initialized in the microprocessor's real mode.
 - Initializing ABIOS means initializing the ABIOS data structures.
 - In bimodal environments, the operating system must initialize parallel sets of Common Data Areas and Function Transfer Tables.
-

continued

ABIOS Initialization Facts, Continued

ABIOS initialization flow

ABIOS initialization flow is illustrated below.

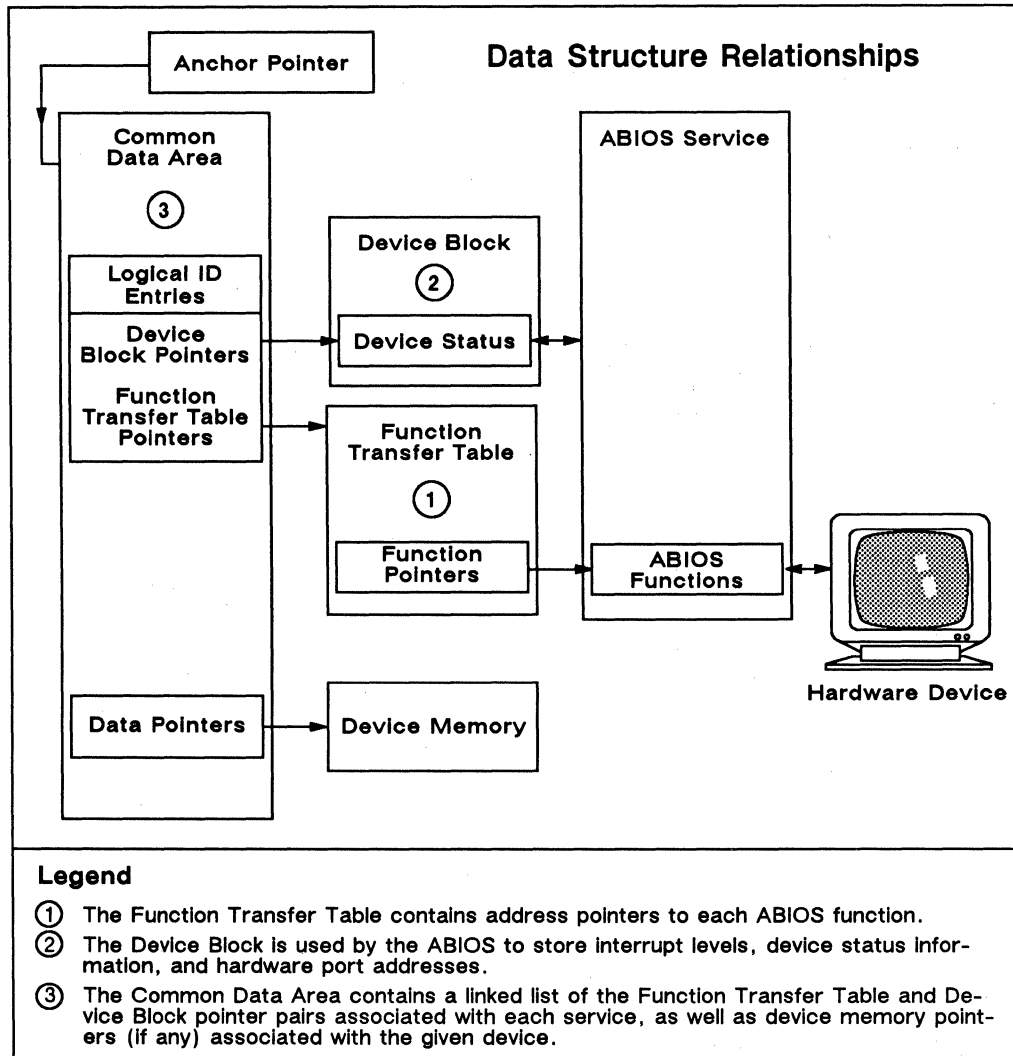


continued

BIOS Initialization Facts, Continued

Data structure relationships

The graphic below shows how the BIOS Data Structures relate to each other once they have been initialized.



Request Block Initialization

Introduction

Each BIOS function is interfaced with its caller via a function-specific structure called the Request Block. Before starting an BIOS function the caller must initialize its Request Block.

Physical device vs. logical device

A hardware device is a device that physically exists in a system configuration. Physical devices are identified by a Device ID; for instance, all hard disks are associated with device ID 02h. Logical IDs, on the other hand, are used to reference individual device services. They are assigned dynamically at initialization, and their assignments will change when system configuration changes. The first device to be initialized will be assigned a Logical ID of 1; the second will be assigned Logical ID 2, and so on.

Logical IDs are mandatory input

Logical IDs are assigned to each BIOS service during BIOS initialization when the operating system builds the Common Data Area. A service's Logical ID is used from then on as an index into the Common Data Area location where the service's Function Transfer Table and Device Block pointers are stored. A service's Logical ID is a mandatory input into each function Request Block calling on that service.

Request Block lifespan

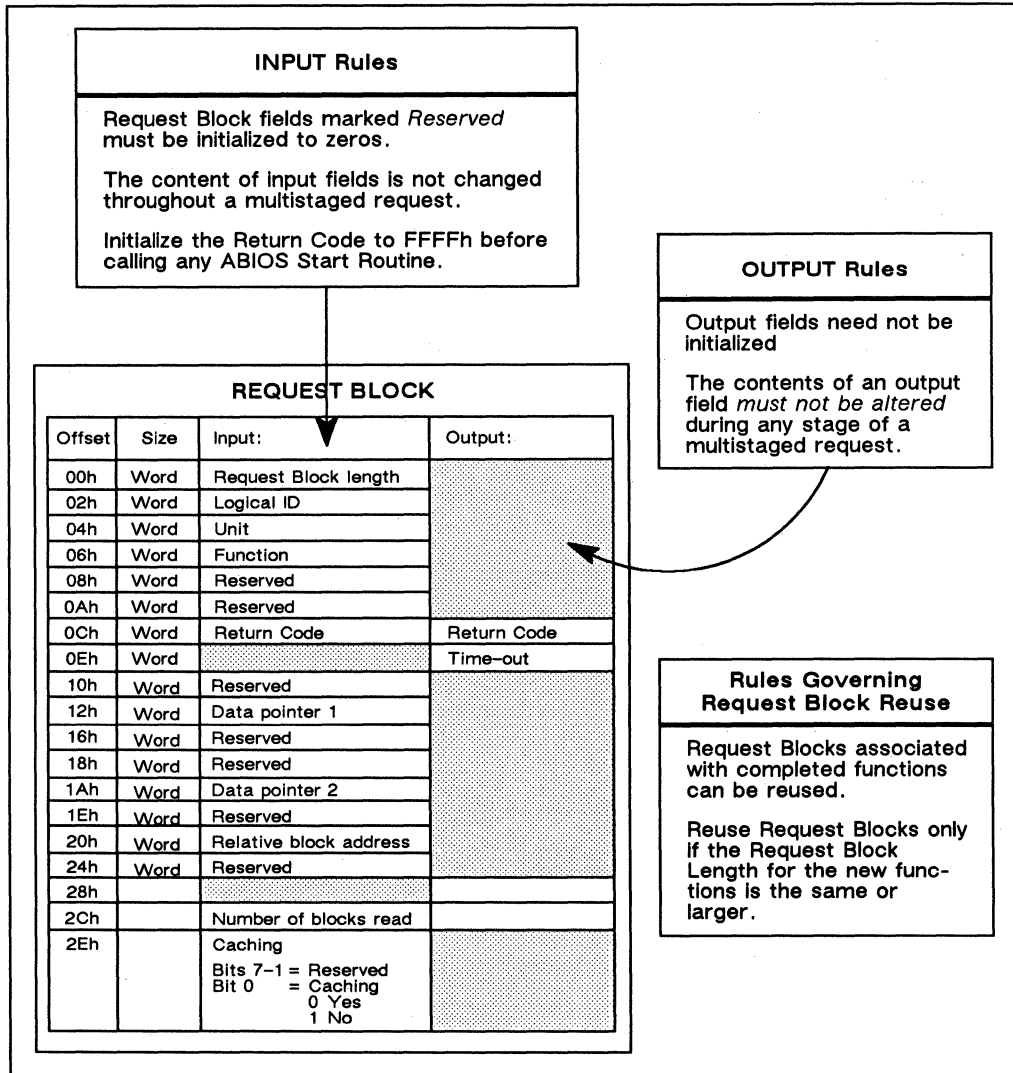
Once initialized, Request Blocks stay resident in system RAM for as long as the function call is active. That is to say, those Request Blocks associated with multistaged functions stay resident in system RAM for the life of the function. Request Blocks associated with completed functions may be reused, or the memory they occupy can be deallocated.

continued

Request Block Initialization, Continued

Rules governing Request Block use

The general rules associated with Request Block use are illustrated below.



Transfer Conventions

Introduction

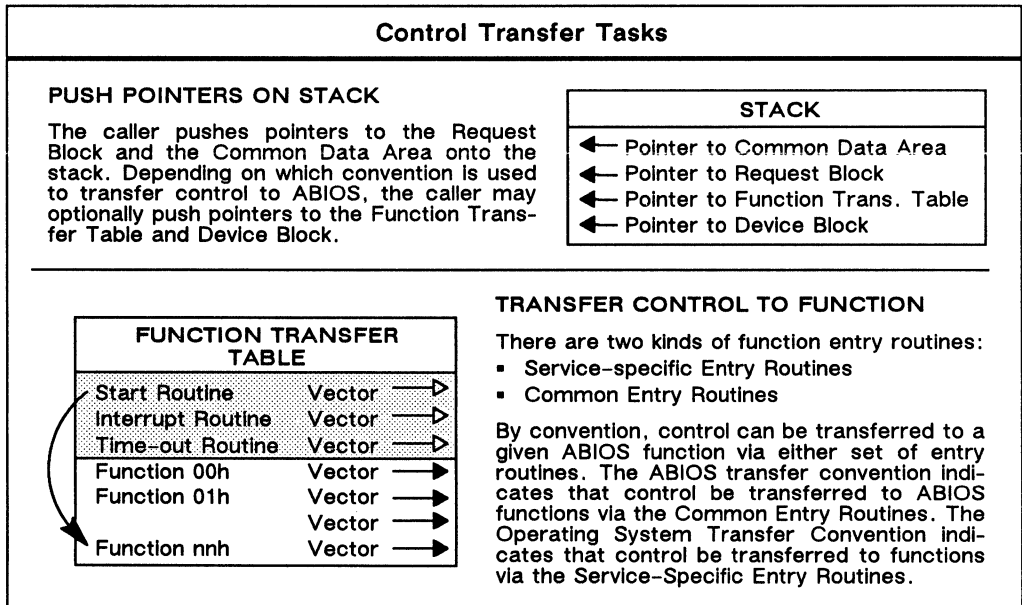
Once its Request Block has been initialized, the caller is free to transfer control to the BIOS function. The process of transferring control to an BIOS function is divided into two basic tasks:

1. PUSH required entry pointers onto stack frame, and
2. CALL the function entry routine.

The difference between the two methods of calling BIOS lies in the process used to locate pointers and load them onto the stack.

Two tasks

The graphic below details each control transfer task.



continued

Transfer Conventions, Continued

ABIOS transfer convention

In the BIOS transfer convention, the operating system transfers control to the BIOS function requested via the BIOS Common Start Routine. From the caller's point of view, the BIOS Transfer Convention is the simpler of the two transfer conventions in that it reduces programming overhead when BIOS functions are called in a bimodal environment.

When using this method, the operating system:

- builds the Request Block,
 - pushes pointers to the Common Data Area and Request Block onto the stack,
 - pushes place holders for the Function Transfer Table and Device Block onto the stack, and
 - calls the BIOS Common Entry Routine requested. When control is transferred to the Common Entry Routine, the Common Entry Routine loads pointers to the Function Transfer Table and the Device Block onto the stack. The operating system then transfers control to the BIOS service's Service-Specific Entry Routine.
-

Operating system transfer convention

In the operating system transfer convention, the operating system transfers control to the function requested via the BIOS service-specific routines.

This transfer convention is more direct and may result in slight performance improvements. It is most effective when used to handle interrupts from programmed I/O devices (such as the keyboard) that require repeated access to one function.

When using this method, the operating system:

- builds the Request Block,
 - pushes pointers to the Common Data Area, Request Block, Function Transfer Table, and Device Block onto the stack, and
 - calls the service-specific entry routine associated with the service/function requested.
-

ABIOS and Program Access

Introduction

For now, OS/2 is the primary operating system using BIOS. OS/2 itself provides end user programs with powerful functionality which makes bypassing the operating system in favor of the BIOS or hardware largely unnecessary. However, the need may arise for the greater hardware control provided by direct access to BIOS Services.

The Anchor Pointer, OS/2, and program access

When BIOS is initialized under OS/2, the segment address of the Common Data Areas is stored by the operating system in a nonpublic variable called the Anchor Pointer. Control cannot be transferred to an BIOS function without first accessing the information contained in the Common Data Area.

Therefore, access to the BIOS is impossible from a program running under OS/2 unless it is supported by the operating system. Although it does not make the Anchor Pointer public, IBM OS/2 (versions 1.1 and beyond), support program access of the BIOS via calls to two **DevHlp** services: **ABIOS-CommonEntry** and **ABIOSCall**.

Programmers who are not using IBM OS/2 should refer to their OS/2 documentation to determine if their version of OS/2 supports direct access to the BIOS.

Accessing BIOS via BIOSCommonEntry

Description

The IBM OS/2 service **BIOSCommonEntry** is used to call an BIOS functions via the BIOS Transfer Convention, that is to say via the BIOS Common Entry Routines.

BIOSCommonEntry initializes the stack frame with pointers in the format required by the current processor mode. It then calls the Common Entry Routine specified in DH. On return, **BIOSCommonEntry** cleans up the stack before returning to the caller.

Caller must locate Logical ID

Before invoking **BIOSCommonEntry**, the caller must first initialize the Request Block associated with the BIOS function to be called. Since a service's Logical ID is a mandatory input into each function Request Block, the caller is responsible for determining the Logical ID assigned to the service being called.

Locating Logical ID via function 01h

Because the Anchor Pointer to the Common Data Area is a nonpublic variable, the only way for the caller to determine a service's Logical ID is to invoke function 01h, Return Logical ID Parameters, for each entry in the Common Data Area.

To do this, the caller must use **BIOSCommonEntry** to invoke function 01h, "Return Logical ID Parameters" for Logical IDs 03h to nnh. The Request Block associated with function 01h of each BIOS service is fixed at 20h bytes. When called, function 01h returns to offset 12h the hardware Device ID associated with the service. From this value, the caller can determine which device service is linked to a given Logical ID.

continued

Accessing BIOS via BIOSCommonEntry, Continued

BIOS supported devices

The BIOS supports 16 kinds of physical devices. There is one BIOS device service for each device. The table below lists the physical device ID and the BIOS device services tied to those devices.

Device ID	Device Type/Service	Device ID	Device Type/Service
00h	BIOS Internal Calls	0Bh	Pointing Device
01h	Diskette	0Ch	Reserved
02h	Fixed Disk	0Dh	Reserved
03h	Video	0Eh	CMOS RAM
04h	Keyboard	0Fh	Direct Memory Access
05h	Parallel Port	10h	Programmable Option Select
06h	Serial Port	11h	Error Log
07h	System Timer	12h-15h	Reserved
08h	Real Time Clock Timer	16h	Keyboard Security
09h	System Services	17h-FFFFh	Reserved
0Ah	Nonmaskable Interrupt		

BIOSCommonEntry Input/Output

Input:

```
MOV SI, Request_Block_Offset ; Offset in DS of Request Block
MOV DH, Which_Com_Routine    ; Indicate in DH which Common
                               ; Routine to call, where:
                               ; 00h = Common Start Routine
                               ; 01h = Common Interrupt Routine
                               ; 02h = Common Time-out Routine

MOV DL, DevHlp_BIOSCommonEntry
CALL [Device_Help]
```

Output:

```
CF      = 0  If call was successful
        = 1  If error occurred
AX      = Error Code
        BIOS not present.
        Unknown BIOS command.
```

continued

To avoid suspension in the background

BIOS functions can sometimes be suspended if the operating environment is shifted from OS/2 mode to the DOS compatibility box. This can occur when functions executed in the DOS compatibility box put the service's operating environment in a state that is unknown to the function called in OS/2 mode.

ROMCriticalSection sets a flag that prevents entry into the DOS compatibility box until the function called via **BIOSCommonEntry** has executed to completion. Since there is no way to determine in advance whether or not a given function is susceptible to suspension, the caller has two choices:

- Call OS/2 **ROMCriticalSection** before calling **BIOSCommonEntry**, or
- Test the function by calling it via **BIOSCommonEntry** and switching to the DOS compatibility box.

If **ROMCriticalSection** is called to prevent entry into the DOS compatibility box, then it must be called again after the BIOS function completes to re-enable entry.

ROMCriticalSection Input/Output

DS must point to the BIOS Device Driver's data segment. Reset DS if it has been previously used in a **PhysToVirt** call.

```
MOV AL, enter_or_exit      ;Critical Section Flag
                           ; = 0 exit
                           ; < > 0 enter
MOV DL,DevHlp_ROMCriticalSection
CALL [Device_Help]
```

For more information

For more information on calling BIOS functions via **BIOSCommonEntry**, refer to the IBM document *IBM Operating System/2 Technical Reference Volume 1*.

Accessing BIOS via BIOSCall

Description

The IBM OS/2 service **BIOSCall** is used to call an BIOS functions via the Operating System Transfer Convention, that is to say via the BIOS Service-Specific Entry Routines.

BIOSCall initializes the stack frame with pointers in the format required by the current processor mode. Then, it calls the Service-Specific Entry Routine specified in DH. On return, **BIOSCall** cleans up the stack before returning to the caller.

BIOSCall Input/Output

Input:

```
MOV AX, LID                ;Service's Logical ID
MOV SI, RB_Offset          ;Data Segment DS offset to
                           ;caller's Request Block
MOV DH, Entry_Point        ;Service-Specific Routine
                           ;00h = Start Routine
                           ;01h = Interrupt Routine
                           ;02h = Time-out Routine

MOV DL, DevHlp_BIOSCall
CALL [Device_Help]
```

Output:

```
CF      = 0  Call was successful
        = 1  Error occurred
AX      =    Error Code
        ABIOS not present.
        Unknown BIOS command.
```

To avoid suspension in the background

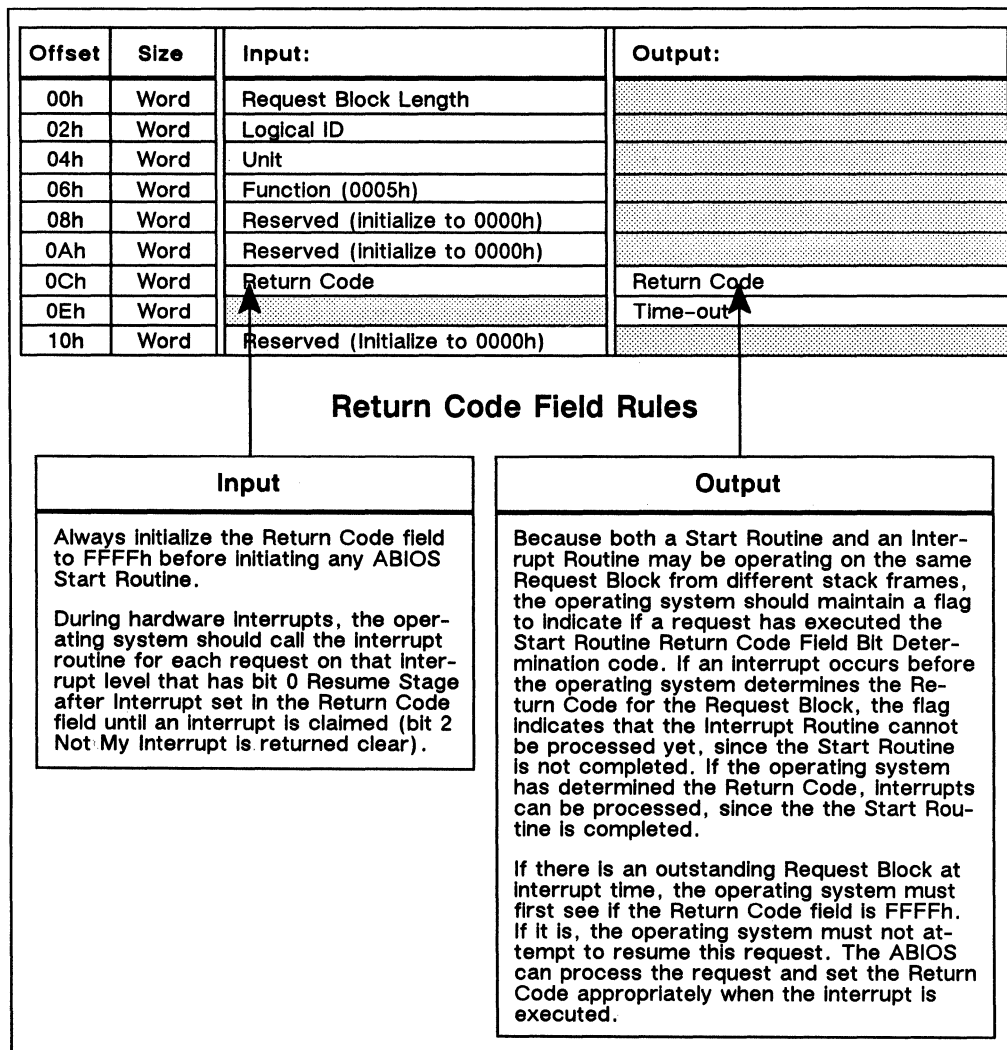
BIOS functions can sometimes be suspended if the operating environment is shifted from OS/2 mode to the DOS compatibility box. To avoid this, the caller may occasionally be advised to call OS/2 **ROMCriticalSection** before calling **BIOSCommonEntry**, as described on the previous page.

Return Code Handling

Introduction

BIOS signals the status (successful, unsuccessful, etc.) of a function call by returning a code to the Request Block for the operating system.

The rules that the operating system must follow when handling this field are listed in the illustration below.



continued

Return Code Handling, Continued

Return codes

The following table contains a general listing of the BIOS Return Codes. BIOS may generate any value that can occur in a 16-bit BIOS field, so all operating system routines that test BIOS Return Codes should be prepared for any value (that is, each bit in the Return Code field should be tested).

Code	Description
0000h	Successful
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
0005h	Not My Interrupt, Resume Stage after Interrupt
0009h	Attention, Resume Stage after Interrupt
0081h	Unexpected Reset, Resume Stage after Interrupt
8000h	Device In Use, Request Refused
8001h-8FFFh	Service-Specific Unsuccessful Operation
9000h-90FFh	Device Error
9100h-91FFh	Retryable Device Error
9200h-9FFFh	Device Error
A000h-A0FFh	Time-out Error
A100h-A1FFh	Retryable Time-out Error
A200h-AFFFh	Time-out Error
B000h-B0FFh	Device Error With Time-out
B100h-B1FFh	Retryable Device Error With Time-out
B200h-BFFFh	Device Error With Time-out
C000h	Invalid Logical ID
C001h	Invalid Function
C002h	Reserved
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h-C01Fh	Invalid Service-Specific Parameter
C020h-FFFEh	Service-Specific Unsuccessful Operation
FFFFh	Return Code Field Not Valid

ABIOS Extensions

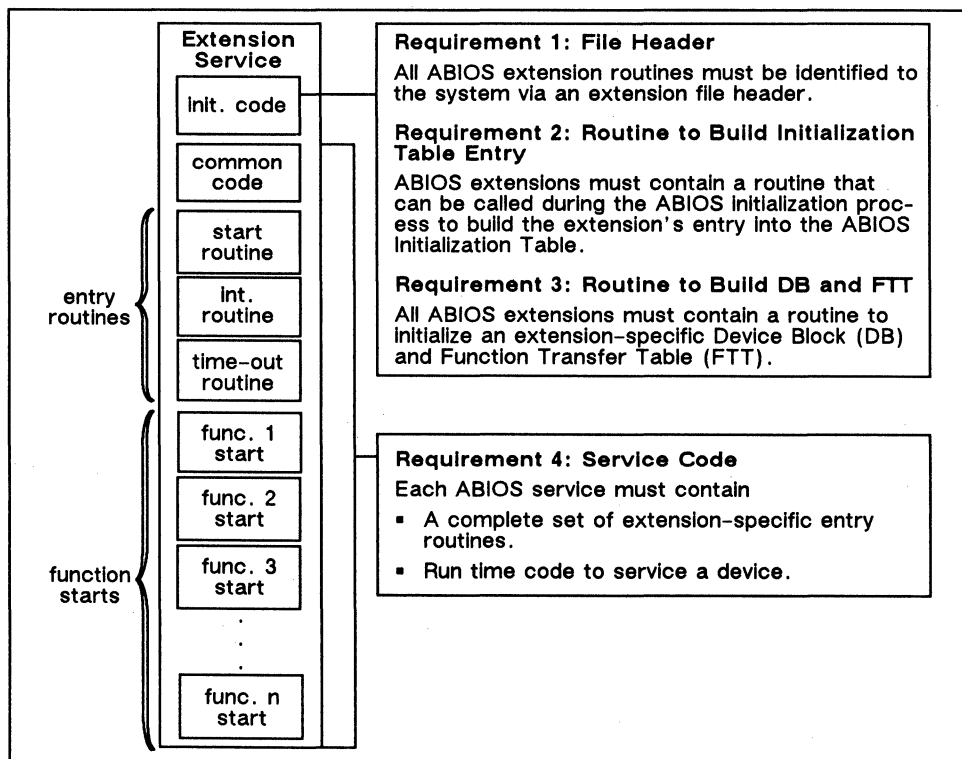
Introduction

Under OS/2, ABIOS data structure addresses and service/function entry points are only known to the operating system. As such, enhancements to system software or hardware that require ABIOS extensions must insure that:

- The undefined interstage state information and work areas contained in the pre-existing service's Device Block are not overwritten.
- The extension maintains control of all function entry points, and
- The extension will only be initialized if its revision level is greater than the revision level of the pre-existing service.

Requirements for ABIOS extensions

In order to satisfy the considerations listed above, all ABIOS extensions must meet the requirements outlined in the illustration below:



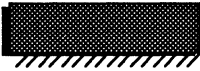
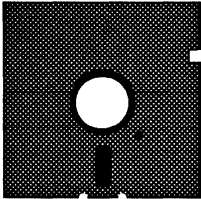
continued

ABIOS Extensions, Continued

Two types of ABIOS implementations

As long as it meets the four requirements for creating valid ABIOS extensions, an extension can be implemented either in ROM or in RAM.

The differences between the two types of implementations are described below.

Type	Description
<p>ROM extension</p> 	<p>ABIOS ROM extensions:</p> <ul style="list-style-type: none">▪ Exist in the same peripheral card ROM that contains their CBIOS counterparts.▪ Contain manufacturer-specific ABIOS device services.▪ Are located and initialized as part of the overall ABIOS ROM initialization process.
<p>RAM extension</p> 	<p>ABIOS RAM extensions exist as files that will be located and initialized into system RAM during the overall ABIOS ROM initialization process. If they are to be implemented under OS/2, extension file names must be listed in the file ABIOS.SYS. The following facts apply:</p> <ul style="list-style-type: none">▪ ABIOS.SYS must contain a list of filespecs separated by either blanks or new lines.▪ Both ABIOS.SYS and any files listed in ABIOS.SYS must reside in the root directory of the OS/2 IPL volume.▪ The files listed in ABIOS.SYS are loaded into memory in the order in which they are listed.▪ The sector size of all RAM extension files must be a multiple of 512K.▪ All ABIOS RAM extension files must have a .BIO extension.▪ RAM extensions are loaded after ROM extensions.

Where to Find More Information

Introduction

This chapter outlines the major concepts and design features of the BIOS. It is intended to leave the reader with a mental model of how the BIOS works. The details of any concept or feature presented here can be found in subsequent chapters of this book.

Where to find more information

For more information on...	Turn to...
BIOS Services	Chapters 8 through 23
BIOS Functions	Chapters 8 through 23
BIOS Request Blocks	Chapters 5 and 6
BIOS Data Structures	Chapter 3
BIOS Function Processing	Chapter 6
BIOS Request Types	Chapter 6
BIOS Multitasking Provisions	Chapters 6 and 7
BIOS Initialization	Chapter 4
BIOS Bimodal Support	Chapters 3 and 4
BIOS Extensions	Chapter 7

Chapter 2

Hardware Environment

Overview

Introduction

This chapter describes the hardware environment supported by the BIOS.

Note: The hardware environment described here also applies to the CBIOS portion of PS/2-compatible ROM BIOSs.

Hardware overview only

The hardware contained in IBM PS/2 and compatible systems provides the user with a rich and powerful computing environment. A precise description of each IBM PS/2 hardware component, however, is beyond the scope of this book. The material presented here is intended to give BIOS users a general survey of the PS/2 hardware and is not intended to substitute for the hardware data sheet or documentation available from individual hardware manufacturers.

continued

ABIOS benefits

By its very nature, the BIOS is designed to isolate the user from direct manipulation of hardware I/O ports, registers, and control words. As such, the BIOS provides a high level interface to hardware that is guaranteed to remain the same when hardware components change.

This has a number of benefits:

- If programs use BIOS calls instead of reads and writes to hardware, they will be more portable across differing hardware environments.
 - Using the BIOS services spares the programmer from having to master many details about hardware that may change.
-

In this chapter

The following hardware information is presented:

- 80286/80386/80386SX Microprocessors
 - Math Coprocessors
 - Micro Channel
 - I/O Devices
 - System Time-Related Devices
 - CMOS RAM Service
 - DMA Controller
 - Programmable Option Select (POS)
 - Intel 8259A Programmable Interrupt Controllers
 - System Control Port Definitions
 - Power-On Password
 - NMI Mask
 - Hardware I/O Port List
-

80286/80386/80386SX Microprocessors

Introduction

The BIOS must be implemented in systems that employ the Intel 80286, 80386SX, or 80386 microprocessor, an equivalent, or a superset.

The table below lists the BIOS microprocessor support.

Item	Support
Processor Speeds	6 to 20 MHz (80286); 6 to 33 MHz (80386, 80386SX) Note: Currently verified range of supported speeds.
Wait States	0 or 1 wait states
Address Modes	Real and protected address modes Note: See "Address mode support" below.

Address mode support

The ROM BIOS for IBM PS/2 and compatible systems can be addressed in both the real and the protected address modes of the microprocessor.

This statement holds true for both the BIOS and the CBIOS portions of the ROM BIOS. The difference between the BIOS and the CBIOS portions of the ROM BIOS lies not in the modes under which they can be addressed but in the kind of operating systems each is designed to support.

- The BIOS is designed specifically to support multitasking operating systems (e.g. IBM OS/2) that execute exclusively in the microprocessor's real mode, exclusively in protected mode, or bimodally — switching between real and protected modes.
 - CBIOS is primarily designed to support single-tasking operating systems (e.g. MS-DOS) that execute exclusively in the microprocessor's real mode.
-

Math Coprocessors

Introduction

The BIOS can be implemented in systems that include an Intel 80287 or 80387 math coprocessor (80387SX for 80386SX processors) or an equivalent math coprocessor.

Description

The math coprocessor performs high speed arithmetic, logarithmic, and trigonometric floating point arithmetic calculations, permitting much speedier processing for mathematically intensive processes.

The math coprocessor works in parallel with the microprocessor, allowing both to process instructions separately. See the user manual for the coprocessor for details about the extended data types, registers, and instructions available with this chip.

Math coprocessor hardware interface

The coprocessor operates in an asynchronous mode and can use the same clock generator as the microprocessor. It functions as an I/O device and can be accessed through I/O ports 00F8h, 00FAh, and 00FCh.

The coprocessor BUSY signal tells the microprocessor that the coprocessor is operating. The WAIT signal means that it is executing, and forces the microprocessor to wait until the coprocessor is done.

The coprocessor can operate in either the real or protected address mode. It is in real address mode after a power-on, a reset, or when returning from protected address mode.

The math coprocessor generates an error signal which sets IRQ 13, and the BUSY signal from the coprocessor is then held in the BUSY state. The BUSY signal can be cleared by an I/O Write to I/O port 00F0h or by a write of zero to I/O port address 00F0h with bits 0-7 set to zeros, which also clears IRQ 13.

continued

Math Coprocessors, Continued

ABIOS: INT 75h Numeric exception error handling

IRQ 13 (which is generated by a math coprocessor numeric exception error) is vectored to INT 75h hardware interrupt service routine as part of the BIOS initialization. When IRQ 13 goes high, INT 75h clears the BUSY signal and issues a software interrupt, INT 02h.

Note: User programs which enable numeric exceptions must provide a method for intercepting and processing either INT 75h or INT 02h.

Micro Channel

Introduction

The BIOS supports such IBM Micro Channel-compatible features as Programmable Option Select (POS) and configurable Direct Memory Access (DMA) arbitration levels.

The following table lists the Micro Channel hardware supported by the BIOS:

Feature	Description
I/O address width is 8-bit or 16-bits.	Allows either 8-bit or 16-bit I/O transfers.
Central arbitration control point.	Arbitrates among as many as 15 devices.
8 DMA channels.	Serial DMA protocol for 8 channels; either 8-bit or 16-bit DMA transfer.
Level-sensitive interrupts.	Interrupt controller operated in level-triggered mode to allow devices to share interrupt levels.
Programmable option select	Eliminates need for jumpers and configuration switches.
Channel extension connectors	Support future growth and additional channel features.
No support for PC-type adapters.	Supports only adapters designed for the Micro Channel.

continued

Micro Channel, Continued

Micro Channel connectors

The BIOS supports three types of channel connectors:

- 16-bit, made up of an 8-bit section, and a 16-bit extension,
 - 16-bit with auxiliary video extensions, and
 - 32-bit connectors in Intel 80386-based systems.
-

Programmable option select (POS)

POS data is accumulated in adapter description files (ADFs) which are created by adapter manufacturers for each adapter. The reference diskette supplied with MCA-compatible systems reads .ADF files and stores configuration information in CMOS RAM. The BIOS power-on self test (POST) reads CMOS RAM and writes the configuration information to the POS registers of the adapters.

The POS I/O addresses are 0094h – 0097h and 0100h – 0107h.

I/O Devices: Introduction

The BIOS supports the following I/O devices:

- a diskette controller,
- a fixed disk controller,
- a VGA video controller,
- an Intel 8042 or equivalent keyboard controller,
- a serial port, and
- a parallel port.

Each of these capabilities is discussed below.

I/O Devices: Diskette and Disk Hardware

Diskette controller

The BIOS supports an NEC 765 or equivalent diskette controller. Four types of diskette drives are supported:

- 720K 3.5-inch,
 - 1.44 MB 3.5-inch,
 - 360K 5.25-inch, and
 - 1.2 MB 5.25-inch drives.
-

Diskette drive configuration

The ROM BIOS supports a maximum of two diskette drives.

Fixed disk controller

The ROM BIOS supports up to two fixed disks. The BIOS supports an ST506 fixed disk adapter or equivalent. This adapter should be PS/2-compatible. The ST506 adapter is single-tasking and must complete one operation before starting another, even though the next operation may be for the other fixed disk. The hardware interrupt request for fixed disk is 14.

RLL and ESDI fixed disk controllers are also supported by the BIOS.

I/O Devices: Video Hardware

Introduction

The BIOS video service supports IBM VGA-compatible hardware, including:

- a VGA-compatible chip or chip set that includes a:
 - CRT Controller
 - Sequencer
 - Graphics Controller, and
 - Attribute Controller
 - DAC chip — INMOS G171 or compatible Digital-to-Analog Converter
 - Video RAM — 256K of dynamic read/write RAM configured as four 64K maps
 - Monochrome or color direct drive analog monitor
 - Monochrome or color multiple sync frequency monitors
-

VGA-compatible chip (or chip set)

The VGA chip (or chip set) provides all CRT control signals. It consists of four components: CRT Controller, Sequencer, Graphics Controller, and Attribute Controller.

The function of each VGA component is summarized in the table below.

Component	Function
CRT Controller	Generates horizontal and vertical CRT sync timings, cursor and underline timings, video buffer addressing, and refresh addressing.
Sequencer	Arbitrates system access to display RAM and fonts. The sequencer allows up to eight fonts with two fonts displayable at any one time.
Graphics Controller	Handles read/write operation on 4 parallel bit planes. Outputs data to Attribute Controller.
Attribute Controller	Converts incoming text mode attribute data or graphics mode pixel data into 8-bit indices into the Digital-to-Analog Converter (DAC) color registers (see below).

Note: For a complete description of VGA-compatible components, I/O ports, and registers, refer to the hardware documentation accompanying your particular VGA-compatible chip or chip set.

continued

I/O Devices: Video Hardware, Continued

DAC

The video DAC contains 256 individual color registers which can be accessed by the BIOS as either four 64-color registers or sixteen 16-color registers.

Each DAC color register contains one 18-bit RGB analog value. Six bits of each register are allocated to each primary color. Thus, the color represented in each DAC color register may be any of 256K possible colors (i.e. $2^3 \times 6 = 256K$).

Video RAM

The ABIOS Video Service requires 256K of read/write video RAM formatted into four banks (or maps) of 64K.

To maintain compatibility, display memory for each of the historical MDA, CGA, and EGA-compatible modes is mapped exactly as it was in the original display adapter. The display memory organization for the new VGA modes is outlined in the ABIOS Video Services chapter.

Analog monitor support

To display all modes, the Video Service requires either a monochrome or a color direct drive analog monitor with a 31.5 KHz horizontal scan frequency.

The display's vertical gain is adjusted automatically by the VGA-compatible circuitry. Thus, video modes with 350, 400, and 480 horizontal scan lines can be displayed without requiring manual adjustment.

Multiscan monitor support

In addition to 31.5 KHz direct drive analog monitors, the Video Service also supports multiscan rate monitors capable of operating in analog modes (e.g. NEC Multisync monitor). Monitors of this type require an adapter cable that matches the signal assignments and monitor ID circuitry of the DAC external video controller.

I/O Devices: Keyboard Hardware

Introduction

The BIOS supports an intelligent keyboard subsystem based on the Intel 8042 or equivalent keyboard controller.

The hardware interrupt level associated with the BIOS Keyboard Service is IRQ 1.

The 8042 controller chip

The Intel 8042 peripheral controller (or compatible) is a single chip micro-computer that can be programmed to allow bidirectional communication between the master microprocessor and up to two auxiliary serial input devices. The 8042 chip, generally, is mounted on the system motherboard. 8042 programs reside as firmware in the 8042 chip itself.

Device support

The kind of devices a given 8042 chip supports are dependent on how the 8042 is programmed.

On IBM PS/2-compatible systems, the 8042 is programmed to allow bidirectional communication between the system and the keyboard, as well as between the system and one other auxiliary serial device, such as a mouse, joystick, or trackball.

Pointing device interface

The pointing device in PS/2-compatible systems is controlled by the 8042. INT 74h handles interrupts from the pointing device. The BIOS Pointing Device Service controls the device operations.

I/O Devices: Parallel Port Hardware

Introduction

The BIOS Parallel Port Service is associated with hardware interrupt request 7. The BIOS supports a parallel port that can transfer eight bits of data at standard TTL levels. The parallel port can be called port 1 through 8, must be IBM PS/2-compatible, and must have a bidirectional mode, supporting both input and output. The parallel port also supports level-sensitive interrupts and a readable interrupt pending status.

Parallel port addresses

The following table lists the common parallel port addresses. Up to 8 parallel ports are supported.

Parallel Port Number	Data Address	Status Address	Control Address
1	03BCh	03BDh	03BEh
2	0378h	0379h	037Ah
3	0278h	0279h	027Ah

Parallel port extended mode

The extended mode of the parallel port can be selected through the port system-based POS registers. The extended mode adds a bidirectional interface.

I/O Devices: Serial Port Hardware

Introduction

The BIOS supports a National Semiconductor 16550 serial port controller or equivalent logic. The serial ports can be addressed as Serial 1–8. See (40:10h) to find out how many serial ports are available. Serial 1 and 3 interrupts are on IRQ 4; Serial 2 and 4 interrupts are on IRQ 3. The serial port base addresses are shown below. BIOS initializes the serial ports in the same order that they reside in the ROM BIOS data area, so the serial port Logical IDs will be in the same order as in the BIOS Data area (40:10h). Additional serial ports and Logical IDs may be initialized.

Serial port addresses/interrupt levels

The serial port addresses and interrupt levels are listed in the table below.

Serial Port Number	Base Address	Interrupt Level
1	03F8h	4
2	02F8h	3
3	3220h	3
4	3228h	3
5	4220h	3
6	4228h	3
7	5220h	3
8	5228h	3

NS 16550 characteristics

The NS 16550, which is functionally compatible with the NS 16450 and the NS 8250, supports:

- Characters of 5, 6, 7, or 8 bits,
 - 1, 1.5, or 2 stop bits, and
 - even, odd, or no parity modes.
-

continued

NS 16550 Serial Communications Controller

The NS 16550 does serial-to-parallel conversions on data received from a peripheral device or a modem, and parallel-to-serial conversion on data received from the system processor. The system processor can read the status of the NS 16550 at any time during its operation. The information furnished includes the type and condition of transfer operations in progress, as well as any error conditions (parity, overrun, framing, or break interrupt) present. The NS 16550 provides complete modem control, and has a user programmable processor-interrupt system.

NS 16550 Serial Controller Registers

The NS 16550 has 12 accessible registers:

- Receiver Buffer Register (Read Only)
- Transmitter Holding Register (Write Only)
- Interrupt Enable Register (Read/Write)
- Interrupt Identification Register (Read Only)
- FIFO (First in/First out) Control Register (Write Only)
- Line Control Register (Read/Write)
- Modem Control Register (Read/Write)
- Line Status Register (Read Only)
- Modem Status Register (Read Only)
- Scratch Register (Read/Write)
- Divisor Latch (LSB) (Read/Write)
- Divisor Latch (MSB) (Read/Write)

Information on the operation of these registers is contained in the National Semiconductor NS 16550 Data Sheet. However, to avoid any incompatibility problems introduced by direct hardware programming, use the access to the serial controller provided through the BIOS services.

Programmable baud rate generator

The serial port controller can operate at speeds of from 110 to 19,200 bps.

System Time-Related Devices: Introduction

System time-related components

The BIOS supports that the following time-handling chips, or their equivalents:

- Intel 82284 Clock Generator
 - Intel 8254A Programmable Interval Timer (PIT)
 - Motorola MC146818A Real Time Clock
-

System Time-Related Devices: 82284 Clock Generator

Description

The Intel 82284 (or compatible) Clock Generator chip interfaces directly to the system microprocessor (CPU). The 82284 chip:

- Provides the CPU with two clock inputs
 - Generates the READY input to the CPU
 - Synchronizes the system RESET input to the CPU.
-

System Time-Related Devices: 8254A PIT Chip

Introduction

The 8254A Programmable Interval Timer (PIT) is a counter and timer that provides three channel timers. All channels are driven by a 1.19 MHz oscillator signal. Each “tick” of the PIT generates hardware interrupt request 0.

The BIOS supports the Intel 8254A programmable counter chip or its equivalent. The timer chip need not include a timer/counter 1 but should provide a limited-function timer/counter 3.

Timer channel differences

There are some differences between the three timer channels.

Counters 0 and 2:

- are independent 16-bit counters,
- can be preset, and
- can count in BCD (binary coded decimal) or in binary.

Counter 3:

- is only 8 bits,
 - can be preset,
 - counts in binary only, and
 - can only count downward.
-

System Timer Modes

The system timer has six modes:

Mode	Name
0	Interrupt on Terminal Count
1	Hardware Retriggerable One-Shot
2	Rate Generator
3	Square Wave
4	Software Triggered Strobe
5	Hardware Retriggerable Strobe

continued

System Time-Related Devices: 8254A PIT Chip, Continued

Common timer mode operations

All modes have the following operations in common:

- All control logic resets when control bytes are written to a counter.
 - Counters do not stop when they reach zero.
 - In Modes 0, 1, 4, and 5 the counter wraps to the highest possible count, and continues to count.
 - In Modes 2 and 3, the counter reloads the initial count and continues to count.
-

Timer Channels

The following table describes the functions of the timer channels. The system timer is treated as a series of I/O ports. Three are count registers, and two are control registers.

Channel	Description	I/O Port	Read/Write Status
0	System Timer	0040h	R/W
2	Tone Generation for Speaker	0042h	R/W
3	Watchdog Timer	0044h	R/W
3	Control Register 3	0047h	W
0, 2	Control Register 0, 2	0043h	W

Watchdog Timer

The Watchdog Timer is used to find out if IRQ 0 is not being serviced, which can be used to detect a user program in a tight loop.

The Watchdog Timer is enabled or disabled by the CBIOS INT 15h, System Services, function C3h.

System Time-Related Devices: MC146818A RTC Chip

Introduction

The BIOS supports a Motorola MC146818A Real Time Clock, or equivalent. This chip is assumed to have at least 64 bytes of nonvolatile CMOS RAM available to store configuration data.

RTC CMOS RAM Addresses

The following table lists the CMOS RAM addresses:

I/O Address	Length	Description
0070h	1 Byte	CMOS RAM address, where: Bit 7 = 1 NMI disabled Bits 6-0 = 0 CMOS RAM address
0071h	1 Byte	CMOS RAM data port

- **To write to CMOS RAM:**
 - Inhibit interrupts.
 - Write the CMOS address to which the data is to be written to I/O port 0070h.
 - Write the data to be written to I/O port 0071h.
 - **To read from CMOS RAM:**
 - Inhibit interrupts.
 - Write the CMOS address from which the data is to be read to I/O Port 0070h.
 - Read from I/O port 0071h.
-

CMOS RAM Service

Introduction

Information may be stored in up to two areas of CMOS RAM. The table below describes the CMOS RAM areas available to the ROM BIOS.

Table of CMOS RAM areas

Data Area	I/O Location	Size	Description
CMOS RAM Data Area	0070h and 0071h	64 bytes	These bytes are located on the Motorola MC146818A Real Time Clock CMOS chip (or its equivalent). All implementations of the BIOS make use of this area to store real time clock, POST, and system configuration data.
Extended CMOS RAM Data Area	0074h, 0075h, and 0076h	2K	When implemented on systems that employ more than four adapter slots, the BIOS requires an additional 2K of CMOS RAM. This extended CMOS RAM is primarily used to store POS data.

DMA Controller

DMA functionality

The hardware environment for DMA transfers is described here in order to explain the background against which the DMA BIOS functions operate. The BIOS, however, serves as a shield between underlying hardware and requests of the operating system, obviating the need for the caller to directly access the DMA controller.

Direct Memory Access (DMA) allows large amounts of data to be transferred from a physical device to system memory or vice versa without microprocessor involvement. A program may initiate a DMA transfer and have no need to copy each byte or word individually, freeing the processor for more complex tasks. DMA transfers are typically from/to a fixed I/O port address to/from a continually incremented memory address.

DMA functionality in Micro Channel systems is a superset of the functionality of two Intel 8237 DMA Controllers, one addressed at every port, starting with Port 0000h, and one addressed at every other port, starting at port 00C0h. Access to 8237-compatible DMA functions and to additional functions *for all channels* is provided at I/O ports 0018h and 001Ah. Data output to port 0018h selects the channel and function, and data output to or input from 001Ah goes to or from the selected internal register.

Bus sharing

The system microprocessor and any currently-transferring DMA users can share the bus by taking turns directing bus cycles (driving the Micro Channel's address lines and certain control signals). An arbitration process determines which of these possible bus masters are ready to direct a cycle. Competing bus masters (DMA devices) are assigned varying priorities, which are weighed during arbitration. Each bus master gets control of the bus for a number of cycles as determined by the arbitration process.

DMA device

A DMA device (or bus master) is one that enters into arbitration for the channel. If it wins, it receives addresses and control signals from the DMA controller so it can read or write data.

continued

DMA Controller functions

A DMA controller is a device which monitors the arbitration process and gives addresses and control signals to the device that has won the bus through arbitration. The controller does not enter into the arbitration itself. PS/2-compatible Micro Channel-based systems provide a DMA controller that supports DMA transfers to/from up to eight devices at once.

DMA hardware registers

The DMA controller maintains several hardware registers for each DMA channel. The key registers are:

- a memory address where the next byte or word will be transferred to or from
 - a count of the remaining bytes to transfer (transfer count)
 - a flag (mode) controlling the transfer direction (to memory or to the device), and
 - transfer status flags for each channel (status)
-

continued

DMA Controller, Continued

DMA hardware registers

The DMA Controller has ten sets of registers, summarized below:

Register	Size (bits)	Number of Registers	How Allocated
Memory Address	24	8	1 per Channel
I/O Address	16	8	1 Per Channel
Transfer Count	16	8	1 Per Channel
Temporary Holding	16	1	All Channels
Mask	4	2	1 for Channels 7-4 1 for Channels 3-0
Arbus	4	2	1 for Channel 4 1 for Channel 0
Mode	8	8	1 per Channel
Status	8	2	1 for Channel 7-4 1 for Channel 3-0
Function	8	1	All Channels
Refresh	9	1	Independent of DMA

Mode Control Field

The Mode Control field provides an opportunity for the caller to use the Autoinitialization and Programmed I/O (PIO) features of the DMA Controller.

- **Autoinitialization**

Specifies if the DMA Controller will initialize automatically when the transfer reaches the terminal count.

- **Programmed I/O**

Specifies that the I/O address is to be programmed to the DMA Controller, driving the I/O address on the bus during the DMA cycles.

continued

Transfer Control Bytes

These fields provide an opportunity for the caller to specify the physical address of the memory and I/O fields for BIOS DMA Service functions 10h, 11h, and 12h.

- **Count Control**

Specifies if the physical address is decremented or incremented during a transfer.

- **Device Size**

Specifies whether this is an 8-bit or 16-bit transfer.

DMA controlled by microprocessor

The microprocessor can address the DMA controller and access the DMA registers. The microprocessor can control the DMA modes, transfer addresses, transfer counts, channel masks, and page registers.

Direct DMA Controller access

Reading directly from or writing directly to any DMA Controller port may cause unpredictable results.

DMA data transfer

After a DMA device has won the arbitration bus and the DMA controller is programmed to service the request, a transfer can take place.

DMA transfers can be:

- single transfer,
 - multiple transfer (burst mode), or
 - read verification.
-

continued

Burst mode

Burst mode is a method of DMA transfer that allows a device to remain inactive for long periods and then send large amounts of data in a short time. Some peripheral devices, e.g. a fixed disk, transfer their data in bursts that are frequently separated by long periods of inactivity. Burst mode is a way of making these devices more efficient. The device asks to be serviced only when it has data to transfer and then does so in large quantities.

Arbitration levels

Arbitration is a process through which devices compete for control of the Micro Channel on a prioritized basis. Arbitration levels are predefined or programmable levels of priority assigned to devices that compete for possession of the channel.

Central Arbitration Control is a hardware function that allows intelligent peripherals to share and control access to the system. Arbitration is organized in levels of priority, and on each level there can be a number of competing devices.

Arbitration Levels are numbered from 00h to 0Fh. In addition, there are Arbitration Levels -1 and -2, which exist only on the system board. Of the former set, Arbitration Level 00h has the highest priority; Level 0Eh has the lowest. All arbitration level priorities are assigned sequentially; 00h through 0Eh are the highest through lowest priorities. Level 0Fh is reserved.

Note: The BIOS DMA service reads the fixed disk arbitration level from the CBIOS extended data area at BIOS initialization and uses this arbitration level throughout the BIOS session.

continued

DMA Controller, Continued

Arbitration levels, cont'd

The following table summarizes the arbitration levels:

Arbitration Levels	Primary Assignment
FEh	Memory Refresh
FFh	NMI
00h	DMA Channel 0*
01h	DMA Channel 1
02h	DMA Channel 2
03h	DMA Channel 3
04h	DMA Channel 4*
05h	DMA Channel 5
06h	DMA Channel 6
07h	DMA Channel 7
08h	Reserved
09h	Reserved
0Ah	Reserved
0Bh	Reserved
0Ch	Reserved
0Dh	Reserved
0Eh	Reserved
0Fh	System Microprocessor
* These DMA Channels can be programmed to any arbitration level	

BIOS functions allow these arbitration levels to be allocated, deallocated or disabled.

continued

DMA Controller, Continued

DMA channel flags

Additional channel flags control whether

- a transfer is to be repeated forever,
 - the memory address is to be decremented or incremented after each cycle,
 - the DMA controller maintains I/O address (this is hardwired in most devices that use DMA), or
 - a byte or a word is transferred at each cycle.
-

Physical and Virtual DMA channels

DMA channels can be either physical or virtual. A physical channel can only have one arbitration level, but a virtual channel can be programmed to own any arbitration level not currently assigned to a different channel. Thus, a virtual DMA channel can have many arbitration levels.

Functionally, there is no difference between physical and virtual channels. Priority is determined by the arbitration level only, where level 00h is the highest priority and level 0Eh the lowest.

Virtual DMA Channels and the Arbus register

The arbitration level assignment for channels 0 and 4 can be programmed using the two 4-bit Arbus registers. The Arbus registers permit dynamic reassignment of the arbitration ID value by which the DMA controller responds to DMA requests for bus arbitration. Channels 0 and 4 can then service devices at any arbitration level.

continued

DMA Controller, Continued

DMA Channels

The DMA channel I/O addresses are defined in the I/O Port Address section later in this chapter. They range from 0000h – 000Fh, 0018h, 001Ah, 0081h – 008Fh, and 00C0h – 00DEh.

DMA extended mode

An extended mode register is available for each programmable DMA channel and is used when a DMA channel requests a DMA data transfer. DMA channels must match the transfer size of the DMA slave, which is programmed by Bit 6 of the extended mode register. DMA read transfers of 16 bits from 8-bit memory or 8-bit memory-mapped I/O devices are not supported.

The following table describes the DMA extended mode register:

Bit Number	Description
7	= 0 Reserved
6	= 0 8-bit transfer = 1 16-bit transfer
5	= 0 Reserved
4	= 0 Reserved
3	= 0 Read memory transfer = 1 Write memory transfer
2	= 0 Verify = 1 Transfer data
1	= 0 Reserved
0	= 0 I/O address 00h = 1 User programs the I/O address

Programmable Option Select (POS)

Introduction

Because it is an integral part of the Micro Channel Architecture (MCA), Programmable Option Select support is assumed.

Adapter slots

The A BIOS supports up to eight adapter cards (or more on systems with a customized BIOS). The A BIOS will work with any system regardless of the number of adapter slots available.

Adapter card Identification

Each adapter card must have a unique 2-byte identifier.

Adapter description files

POS data is accumulated in adapter description files (ADFs) that are created by adapter manufacturers for each adapter. The reference diskette supplied with MCA-compatible systems reads the .ADF files and stores the configuration information in CMOS RAM. This information is read by the BIOS power-on self test (POST), which writes it to the POS registers of the adapters and the system board.

Intel 8259A Programmable Interrupt Controllers

Introduction

The BIOS supports two cascaded Intel 8259A Programmable Interrupt Controller (PIC) chips, or their functional equivalent.

Description

The PIC handles all maskable hardware interrupts. The BIOS insures compatibility with hardware interrupts expected by existing software. Formerly reserved, IRQ2 is now associated with the 8042 Auxiliary Device (mouse) controller.

Levels of Interrupt

There are 16 levels of interrupts. Interrupts are level-sensitive, allowing several devices to share the same hardware interrupt. This reduces the interrupt controller's sensitivity to a transient signal on the Micro Channel bus.

Programmable Interrupt Controller Addresses

The following table lists all PIC addresses:

I/O Address	Functional Equivalents	IRQs Routed	Read/Write Status	Description
20h	master 8259A	IRQ 0 - IRQ 7	R/W	Base
21h	master 8259A	IRQ 0 - IRQ 7	W	Mask
A0h	slave 8259A	IRQ 8 - IRQ 15	R/W	Base
A1h	slave 8259A	IRQ 8 - IRQ 15	W	Mask

continued

Table of IRQ assignments

The following table shows the assignment of interrupt requests to commonly interrupting devices, listed from highest to lowest priority. Other devices not listed may use interrupt requests and may share requests with these devices.

Request	Device
IRQ 0	Timer tick
IRQ 1	Keyboard
IRQ 2	IRQ 8-15 are cascaded through IRQ 2
IRQ 8	Real time clock
IRQ 9	Redirect cascade
IRQ 10	Reserved
IRQ 11	Reserved
IRQ 12	Mouse
IRQ 13	80x87 math coprocessor exception
IRQ 14	Fixed disk controllers
IRQ 15	Reserved
IRQ 3	Serial port 2
IRQ 4	Serial port 1
IRQ 5	Parallel port 2
IRQ 6	Diskette
IRQ 7	Parallel port 1

System Control Port Definitions

Introduction

The BIOS supports certain system control functions at I/O Port addresses 0061h and 0092h.

Port 61h: System Control Port B

The read/write definitions for I/O port address 0061h, System Control Port B, are:

Port 61h: Write Operations	
Bit	Description
7	= 1 Reset timer 0 output latch (IRQ 0)
6-4	= 0 Reserved
3	= 0 Enable channel check
2	= 0 Enable parity check
1	= 1 Speaker data enable
0	= 1 Enable Timer 2 gate to speaker

Port 61h: Read Operations	
Bit	Description
7	= 1 Parity check
6	= 1 Channel check
5	= 1 Timer 2 output
4	= 1 Toggles with each refresh request
3	= 0 Channel check enabled
2	= 0 Parity check enabled
1	= 1 Speaker data enabled
0	= 1 Timer 2 gate to speaker enabled

continued

System Control Port Definitions, Continued

Port 0092h: System Control Port A

I/O port 0092h (R/W), System Control Port A, contains system control flag data. System control Port A is defined below:

Bit	Description
7	1 = Fixed disk activity light A on
6	1 = Fixed disk activity light B on
5	0 = Reserved (must be zero)
4	1 = Watchdog time-out occurred
3	1 = Security lock latch is locked
2	0 = Reserved (must be zero)
1	1 = Alternate gate A20 line active
0	1 = Alternate hot CPU reset

Power-On Password

There are eight bytes in CMOS RAM reserved for a 1–7 byte password and a check character for that password. The microprocessor accesses these bytes during POST. These bytes cannot be accessed by a user program.

These eight bytes are initialized to zeros and are changed to the password during password installation. Passwords can only be changed during POST via the program on the reference diskette which is supplied with MCA-compatible systems.

Models 25 and 30: Power-on password is not supported in IBM PS/2 Models 25 and 30, and compatibles.

NMI Mask

The NMI (nonmaskable interrupt) input to the microprocessor is masked off at power-on reset. Bit 7 of I/O address 0070h is set to zero to enable the NMI. A power-on reset sets this bit to one.

Hardware I/O Port List

Table: Hardware I/O Port Definitions

I/O Address	Read/Write Status	Description
0000h	R/W	DMA channel 0, memory address register
0001h	R/W	DMA channel 0, transfer count register
0002h	R/W	DMA channel 1, memory address register
0003h	R/W	DMA channel 1, transfer count register
0004h	R/W	DMA channel 2, memory address register
0005h	R/W	DMA channel 2, transfer count register
0006h	R/W	DMA channel 3, memory address register
0007h	R/W	DMA channel 3, transfer count register
0008h	R	DMA channel 0-3, status register, where: Bit 7 = 1 Channel 3 request Bit 6 = 1 Channel 2 request Bit 5 = 1 Channel 1 request Bit 4 = 1 Channel 0 request Bit 3 = 1 Terminal count on channel 3 Bit 2 = 1 Terminal count on channel 2 Bit 1 = 1 Terminal count on channel 1 Bit 0 = 1 Terminal count on channel 0
000Ah	R/W	DMA channel 0-3, mask register, where: Bits 7-3 = 0 Reserved Bit 2 = 0 Clear mask Bit = 1 Set mask Bit Bits 1-0 = 00b Select channel 0 = 01b Select channel 1 = 10b Select channel 2 = 11b Select channel 3
000Bh	W	DMA channel 0-3, mode register, where: Bits 7-6 = 00b Demand mode = 01b Signal mode = 10b Block mode = 11b Cascade mode Bits 5-4 = 0 Reserved Bits 3-2 = 00b Verify operation = 01b Write operation = 10b Read operation = 11b Reserved Bits 1-0 = 00b Select channel 0 = 01b Select channel 1 = 10b Select channel 2 = 11b Select channel 3
000Ch	W	DMA Clear Byte Pointer
000Dh	W	DMA Master Clear Byte

continued

Hardware I/O Port List, Continued

Table: Hardware I/O Port Definitions, cont'd

I/O Address	Read/Write Status	Description
000Eh	W	DMA Channel 0-3 Clear Mask Register
000Fh	W	DMA channel 0-3, write mask register, where: Bits 7-4 = 0 reserved Bit 3 = 0 unmask channel 3 mask bit = 1 set channel 3 mask bit Bit 2 = 0 unmask channel 2 mask bit = 1 set channel 2 mask bit Bit 1 = 0 unmask channel 1 mask bit = 1 set channel 1 mask bit Bit 0 = 0 unmask channel 0 mask bit = 1 set channel 0 mask bit
0018h	W	DMA extended function register, where: Bits 7-4 = Progress command, where: 00h I/O address register 01h Reserved 02h Memory address register write 03h Memory address register read 04h Transfer count register write 05h Transfer count register read 06h Status register read 07h Mode register 08h Arbus register 09h Mask register set single bit 0Ah Mask register reset single bit 0Bh-0Ch Reserved 0Dh Master clear 0Eh-0Fh Reserved Bits 3-0 = 0 Reserved
001Ah	R/W	DMA extended function execute register
0020h	R	PIC, Interrupt request/In-service registers programmed by Operation Command Word 3 (OCW3): Interrupt request register, where: Bits 7-0 = 0 No active request for the corresponding interrupt line = 1 Active request for the corresponding interrupt line Interrupt in-service register, where: Bits 7-0 = 0 The corresponding interrupt line is not currently being serviced = 1 The corresponding interrupt line is currently being serviced

continued

Hardware I/O Port List, Continued

Table: Hardware I/O Port Definitions, cont'd

I/O Address	Read/Write Status	Description
0020h	W	<p>PIC, Initialization Command Word 1 (ICW1) (Bit 4 is one), where:</p> <ul style="list-style-type: none"> Bits 7-5 = 000 Only used in 80/85 mode Bit 4 = 1 Reserved Bit 3 = 0 Edge triggered mode = 1 Level triggered mode Bit 2 = 0 Successive interrupt vectors are separated by eight bytes = 1 Successive interrupt vectors are separated by four bytes Bit 1 = 0 Cascade mode = 1 Single mode — no ICW3 needed Bit 0 = 0 No ICW4 needed = 1 ICW4 needed
0021h	W	<p>PIC, ICW2, ICW3, or ICW4 in sequential order after ICW1 written to Port 0020h</p> <p>ICW2, where:</p> <ul style="list-style-type: none"> Bits 7-3 = Address lines A0-A3 of base vector address for interrupt controller Bits 2-0 = 000 Reserved <p>ICW3, where:</p> <ul style="list-style-type: none"> Bits 7-0 = 0 Slave controller not attached to corresponding interrupt pin = 0 Slave controller attached to corresponding interrupt pin <p>ICW4, where:</p> <ul style="list-style-type: none"> Bits 7-5 = 000 Reserved Bit 4 = 0 No special fully-nested mode = 1 Special fully-nested mode Bits 3-2 = 00 Non-buffered mode = 01 Non-buffered mode = 10 Buffered mode/slave = 11 Buffered mode/master Bit 1 = 0 Normal EOI = 1 Auto EOI Bit 0 = 0 80/85 mode = 1 8086/8088 mode
0021h	R/W	<p>PIC, Interrupt mask register (OCW1), where:</p> <ul style="list-style-type: none"> Bit 7 = 0 Enable parallel printer interrupt Bit 6 = 0 Enable diskette interrupt Bit 5 = 0 Enable fixed disk interrupt Bit 4 = 0 Enable serial port 1 interrupt Bit 3 = 0 Enable serial port 2 interrupt Bit 2 = 0 Enable video interrupt Bit 1 = 0 Enable keyboard interrupt Bit 0 = 0 Enable timer interrupt

continued

Hardware I/O Port List, Continued

Table: Hardware I/O Port Definitions, cont'd

I/O Address	Read/Write Status	Description
0021h	W	<p>PIC, OCW2 (Bit 4 is zero, Bit 3 is zero), where:</p> <p>Bits 7-5 = 000 Rotate in automatic EOI mode (clear)</p> <p> = 001 Non-specific EOI</p> <p> = 010 No operation</p> <p> = 011 Specific EOI</p> <p> = 100 Rotate in automatic EOI mode (set)</p> <p> = 101 Rotate on non-specific EOI command</p> <p> = 110 Set priority command</p> <p> = 111 Rotate on specific EOI command</p> <p>Bit 4 = 0 Reserved</p> <p>Bit 3 = 0 Reserved</p> <p>Bits 2-0 = Interrupt request to which the command applies</p>
0020h	W	<p>PIC, OCW3 (Bit 4 is zero, Bit 3 is one), where:</p> <p>Bit 7 = 0 Reserved</p> <p>Bits 6-5 = 00 No operation</p> <p> = 01 No operation</p> <p> = 10 Reset special mask</p> <p> = 11 Set special mask</p> <p>Bit 4 = 0 Reserved</p> <p>Bit 3 = 1 Reserved</p> <p>Bit 2 = 0 No poll command</p> <p> = 1 Poll command</p> <p>Bits 1-0 = 00 No operation</p> <p> = 01 No operation</p> <p> = 10 Read interrupt request register on next read at Port 0020h</p> <p> = 11 Read interrupt in-service register on next read at Port 0020h</p>
0040h	R/W	Programmable Interrupt Timer - Read/write counter 0
0042h	R/W	Programmable Interrupt Timer - Read/write counter 2

continued

Hardware I/O Port List, Continued

Table: Hardware I/O Port Definitions, cont'd

I/O Address	Read/Write Status	Description
0043h	W	<p>PIT, control word register for counters 0 and 2, where:</p> <p>Bits 7-6 = 00b Select counter 0 = 01b Reserved = 10b Select counter 2</p> <p>Bits 5-4 = 00b Counter latch command = 01b Read/write counter bits 0-7 only = 10b Read/write counter bits 8-15 only = 11b Read/write counter bits 0-7 first, then bits 8-15</p> <p>Bits 3-0 = 000b Mode 0 select = 001b Mode 1 select = X10b Mode 2 select = X11b Mode 3 select = 100b Mode 4 select = 101b Mode 5 select</p> <p>Bit 0 = 0 Binary counter 16 bits = 1 Binary coded decimal counter</p>
0044h	W	PIT, read/write counter 3
0047h	W	<p>PIT, control word register for counter 3, where:</p> <p>Bits 7-6 = 00b Select counter 3 = 01b Reserved = 10b Reserved = 11b Reserved</p> <p>Bits 5-4 = 00b Counter latch command select counter 0 = 01b Read/write counter bits 0-7 only = 10b Reserved = 11b Reserved</p>
0060h	R/W	Keyboard/auxiliary data port
0061h	R	<p>System control port B, where:</p> <p>Bit 7 = 1 Parity check Bit 6 = 1 Channel check Bit 5 = 1 Timer 2 output Bit 4 = 1 Toggle with each refresh request Bit 3 = 0 Channel check enabled Bit 2 = 0 Parity check enabled Bit 1 = 1 Speaker data enabled Bit 0 = 1 Timer 2 gate to speaker enabled</p>

continued

Hardware I/O Port List, Continued

Table: Hardware I/O Port Definitions, cont'd

I/O Address	Read/Write Status	Description
0061h	W	System control port B, where: Bit 7 = 1 Reset timer 0 output latch (IRQ 0) Bits 6-4 = Reserved Bit 3 = 0 Enable channel check Bit 2 = 0 Enable parity check Bit 1 = 1 Speaker data enable Bit 0 = 1 Enable timer 2 gate to speaker
0064h	W	8042 Commands
0064h	R	8042 Status, where: Bit 7 = 1 Parity error Bit 6 = 1 General time out Bit 5 = 1 Auxiliary output buffer full Bit 4 = 1 Inhibit switch Bit 3 = 1 Command/data Bit 2 = System flag Bit 1 = 1 Input buffer full Bit 0 = 1 Output buffer full
0070h	W	CMOS RAM address register port, where: Bit 7 = 1 NMI disable = 0 NMI enabled Bits 6-0 = 0 CMOS RAM address
0071h	R/W	CMOS RAM data register port
0074h	W	Extended CMOS RAM address register port, least significant byte
0075h	W	Extended CMOS RAM address register port, most significant byte
0076h	R/W	Extended CMOS RAM data register port
0080h	R	DMA access
0081h	R/W	DMA channel 2, page table address register
0082h	R/W	DMA channel 3, page table address register
0083h	R/W	DMA channel 1, page table address register
0087h	R/W	DMA channel 0, page table address register
0089h	R/W	DMA channel 6, page table address register
008Ah	R/W	DMA channel 7, page table address register
008Bh	R/W	DMA channel 5, page table address register
008Fh	R/W	DMA channel 4, page table address register

continued

Hardware I/O Port List, Continued

Table: Hardware I/O Port Definitions, cont'd

I/O Address	Read/Write Status	Description
0090h	R	DMA arbitration register, where: Bit 7 = 1 System microprocessor cycles enabled Bit 6 = 1 Arbitration mask by NMI Bit 5 = 1 Bus timeout Bit 4 = 0 Reserved Bits 3-0 = Arbitration level
0090h	W	DMA arbitration register, where: Bit 7 = 1 Enable system microprocessor cycle Bit 6 = 1 Arbitration mask Bit 5 = 1 Enable extended arbitration Bits 4-0 = 0 Reserved
0091h	W	DMA card selected feedback register, where: Bits 7-1 = Reserved Bit 0 = 1 Card selected feedback signal active on previous cycle or system board I/O functions accessed by an I/O cycle
0092h	R/W	System control port A, where: Bit 7 = 1 Fixed disk activity light bit A on Bit 6 = 1 Fixed disk activity light bit B on Bit 5 = X Reserved Bit 4 = 1 Watchdog timeout occurred Bit 3 = 1 Security lock latch locked Bit 2 = X Reserved Bit 1 = 1 Alternate gate A20 active Bit 0 = 1 Alternate hot reset
0094h	R/W	System board setup enable register, where: Bit 7 = 0 Enable system board setup = 1 Disable system board setup Bit 6 = 1 Reserved Bit 5 = 0 Enable VGA setup = 1 Disable VGA setup Bits 4-0 = 1 Reserved
0096h	R/W	POS channel position select register, where: Bit 7 = 1 Channel 1 reset Bits 6-4 = Reserved (written as 0, read as 1) Bit 3 = 1 Channel select Bits 2-0 = 0 Channel number
00A0h	R/W	Programmable Interrupt Controller 2

continued

Hardware I/O Port List, Continued

Table: Hardware I/O Port Definitions, cont'd

I/O Address	Read/Write Status	Description
00A1h	R/W	Programmable Interrupt Controller 2 mask, where: Bit 7 = 0 Reserved Bit 6 = 0 Enable fixed disk interrupt Bit 5 = 0 Enable 80387 exception interrupt Bit 4 = 0 Enable mouse interrupt Bit 3 = 0 Reserved Bit 2 = 0 Reserved Bit 1 = 0 Enable redirect cascade Bit 0 = 0 Enable real time clock interrupt
00C0h	R/W	DMA channel 4, memory address register
00C2h	R/W	DMA channel 4, transfer count register
00C4h	R/W	DMA channel 5, memory address register
00C6h	R/W	DMA channel 5, transfer count register
00C8h	R/W	DMA channel 6, memory address register
00CAh	R/W	DMA channel 6, transfer count register
00CCh	R/W	DMA channel 7, memory address register
00CEh	R/W	DMA channel 7, transfer address register
00D0h	R	DMA channel 4-7, status register, where: Bit 7 = 1 Channel 7 request Bit 6 = 1 Channel 6 request Bit 5 = 1 Channel 5 request Bit 4 = 1 Channel 4 request Bit 3 = 1 Terminal count on channel 7 Bit 2 = 1 Terminal count on channel 6 Bit 1 = 1 Terminal count on channel 5 Bit 0 = 1 Terminal count on channel 4
00D4h	R/W	DMA channel 4-7, mask register, where: Bits 7-3 = 0 Reserved Bit 2 = 0 Clear mask bit = 1 Set mask bit Bits 1-0 = 00b Select channel 4 = 01b Select channel 5 = 10b Select channel 6 = 11b Select channel 7

continued

Hardware I/O Port List, Continued

Table: Hardware I/O Port Definitions, cont'd

I/O Address	Read/Write Status	Description
00D6h	R/W	DMA channel 4–7, Mode Register, where: Bits 7–6 = 00b Demand mode = 01b Single mode = 10b Block mode = 11b Cascade mode Bit 5 = 0 Reserved Bit 4 = 0 Reserved Bits 3–2 = 00b Verify operation = 01b Write operation = 10b Read operation = 11b Reserved Bits 1–0 = 00b Select channel 4 = 01b Select channel 5 = 10b Select channel 6 = 11b Select channel 7
00D8h	W	DMA Clear Byte Pointer
00DAh	W	DMA Master Clear
00DCh	W	DMA Channel 4–7, Clear Mask Register
00DEh	W	DMA Channel 4–7, Write Mask Register, where: Bits 7–4 = 0 Reserved Bit 3 = 0 Unmask channel 7 mask bit = 1 Set channel 7 mask bit Bit 2 = 0 Unmask channel 6 mask bit = 1 Set channel 6 mask bit Bit 1 = 0 Unmask channel 5 mask bit = 1 Set channel 5 mask bit Bit 0 = 0 Unmask channel 4 mask bit = 1 Set channel 4 mask bit
F0–FFh	R/W	Math Coprocessor
0100h	R	POS Adapter Identification (least significant byte)
0101h	R	POS Adapter Identification (most significant byte)

continued

Hardware I/O Port List, Continued

Table: Hardware I/O Port Definitions, cont'd

I/O Address	Read/Write Status	Description
0102h	R/W	<p>POS register 2 for the system board setup, where:</p> <p>Bit 7 = 0 Enable parallel port extended mode = 1 Disable parallel port extended mode</p> <p>Bits 6, 5 = Parallel port select, where: = 00 Parallel 1 3BC-3BE = 01 Parallel 2 278-37A = 10 Parallel 3 278-27A = 11 Reserved</p> <p>Bit 4 = 1 Enable parallel port, if bit 0 = 1 = 0 Disable parallel port</p> <p>Bit 3 = 1 System board serial is serial 1 = 0 System board serial is serial 2</p> <p>Bit 2 = 1 Enable serial port, if bit 1 = 1 = 0 Disable serial port</p> <p>Bit 1 = 1 Enable diskette, if bit 0 = 1 = 0 Disable diskette</p> <p>Bit 0 = 1 Allows bits 4, 2, and 1 to enable/disable devices = 0 Disables system board devices</p> <p>For an adapter card, the following bit is defined: Bit 0 = 1 Enable adapter card</p>
0103h	R/W	<p>POS register 3, where:</p> <p>Bits 7-2 = Reserved Bit 1 = 0 Password disable Bit 0 = Reserved</p>
0104h	R/W	POS register 4
0105h	R/W	<p>POS register 5 for an adapter card, where:</p> <p>Bit 7 = 0 Channel check condition occurred = 1 Channel reset</p> <p>Bit 6 = 0 Channel check exception status available</p>
0106h	R/W	POS subaddress extension (least significant byte)
0107h	R/W	POS subaddress extension (most significant byte)
1F0-1F8h	R/W	Fixed disk
200-20Fh	R/W	Game port
0278h	R/W	Parallel 3, Data Port

continued

Hardware I/O Port List, Continued

Table: Hardware I/O Port Definitions, cont'd

I/O Address	Read/Write Status	Description
0279h	R/W	Parallel 3, Status Port, where: Bit 7 = 0 Busy Bit 6 = 0 Acknowledge Bit 5 = 1 Out of paper Bit 4 = 1 Printer is selected Bit 3 = 0 Error Bit 2 = 0 IRQ has occurred Bits 1-0 = Reserved
027Ah	R/W	Parallel 3, Control Port, where: Bits 7-6 = Reserved Bit 5 = 0 Direction is write to port = 1 Direction is read from port Bit 4 = 1 Enable IRQ Bit 3 = 1 Select printer Bit 2 = 0 Initialize printer Bit 1 = 1 Automatic line feed Bit 0 = 1 Strobe
02F8h	W	Serial 2, transmitter holding register, where: Bits 7-0 = Data bits 7-0, respectively, when Divisor Latch Access Bit = 0
02F8h	R	Serial 2, receiver buffer register, where: Bits 7-0 = Data bits 7-0, respectively, when DLAB = 0
02F8h	R/W	Serial 2, divisor latch, low byte, where: Bits 7-0 = Bits 7-0 of divisor, when DLAB = 1
02F9h	R/W	Serial 2, divisor latch, high byte Bits 7-0 = Bits 15-8 of divisor, when DLAB = 1
02F9h	R/W	Serial 2, interrupt enable register, where: Bits 7-4 = 0 Reserved Bit 3 = 1 Modem status interrupt enable Bit 2 = 1 Receiver line status interrupt enable Bit 1 = 1 Transmitter holding register empty interrupt enable Bit 0 = 1 Received data available interrupt enable when DLAB = 0
02FAh	R	Serial 2, interrupt identification register, where: Bits 7-3 = 0 Reserved Bits 2-1 = 00b Modem status interrupt = 01b Transmitter holding register empty interrupt = 10b Received data available register interrupt = 11b Receiver line status interrupt Bit 0 = 0 Interrupt pending

continued

Hardware I/O Port List, Continued

Table: Hardware I/O Port Definitions, cont'd

I/O Address	Read/Write Status	Description
02FAh	W	Serial 2, FIFO control register, where: Bits 7-6 = Receiver FIFO register trigger 00b = 1 byte 01b = 4 bytes 10b = 8 bytes 11b = 14 bytes Bits 5-3 = 0 Reserved Bit 2 = Transmitter FIFO register reset 1 = transmit FIFO register cleared, counter cleared, bit is self-clearing Bit 1 = Receiver FIFO register reset 1 = receiver FIFO register cleared, counter cleared, bit is self-clearing Bit 0 = FIFO enable 1 = receiver and transmitter FIFOs enabled, must be 1 to program FIFO registers 0 = clears receive and transmit FIFO registers, enters character mode
02FBh	R/W	Serial 2, line control register, where: Bit 7 = 1 Divisor latch access = 0 Receiver buffer, transmitter holding, or interrupt enable registers access Bit 6 = 1 Set break enable Bit 5 = Stick parity Bit 4 = Even parity select Bit 3 = Parity enable Bit 2 = Number of stop bits Bits 1-0 = 00b Word length is 5 bits = 01b Word length is 6 bits = 10b Word length is 7 bits = 11b Word length is 8 bits
02FCh	R/W	Serial 2, modem control register, where: Bits 7-5 = 0 Reserved Bit 4 = 1 Loopback mode Bit 3 = 1 Enable OUT2 interrupt Bit 2 = 1 Force OUT1 active Bit 1 = 1 Force request-to-send active Bit 0 = 1 Force data-terminal-ready active

continued

Hardware I/O Port List, Continued

Table: Hardware I/O Port Definitions, cont'd

I/O Address	Read/Write Status	Description
02FDh	R	Serial 2, line status register, where: Bit 7 = 0 Reserved Bit 6 = 1 Transmitter shift and holding registers empty Bit 5 = 1 Transmitter holding register empty Bit 4 = 1 Break interrupt Bit 3 = 1 Framing error Bit 2 = 1 Parity error Bit 1 = 1 Overrun error Bit 0 = 1 Data ready
02FEh	R	Serial 2, modem status register, where: Bit 7 = 1 Data carrier detect Bit 6 = 1 Ring indicator Bit 5 = 1 Data set ready Bit 4 = 1 Clear to send Bit 3 = 1 Delta to carrier detect Bit 2 = 1 Trailing edge ring indicator Bit 1 = 1 Delta data set ready Bit 0 = 1 Delta clear to send
02FFh	R/W	Serial 2, scratch register
0320h	R/W	Fixed Disk Adapter Register (8 or 16 bit)
0322h	W	Fixed Disk Adapter Control Register, where: Bit 7 = 1 Reset Bit 6 = 1 Reserved (except during reset) Bit 5 = 1 16-bit mode (must match bit 2) = 0 8-bit mode Bit 4-3 = 0 Reserved Bit 2 = 1 16-bit mode (must match bit 5) = 0 8-bit mode Bit 1 = 1 Enable interrupt through Programmable Interrupt Controller (hardware interrupt) = 0 enable interrupt through Interrupt Status Register (port 324h) Bit 0 = 1 DMA mode = 0 PIO mode
0322h	R	Fixed Disk Adapter Status Register, where: Bit 7-6 = 0 Reserved Bit 5 = 1 16-bit mode = 0 8-bit mode Bit 4 = 1 Data transfer requested by adapter Bit 3 = 1 Direction is adapter to system = 0 Direction is system to adapter Bit 2 = 1 Busy Bit 1 = 1 Interrupt request (notification) Bit 0 = 1 Transfer in progress

continued

Hardware I/O Port List, Continued

Table: Hardware I/O Port Definitions, cont'd

I/O Address	Read/Write Status	Description
0324h	W	Fixed Disk Adapter Attention Register, where: Bit 7 = 1 Command control block Bit 6 = 1 Command specify block Bit 5 = 1 Sense summary block Bit 4 = 1 Data transfer requested by system Bit 3 = 0 Reserved Bit 2 = 0 Drive 0 select = 1 drive 1 select Bit 1 = 0 Reserved Bit 0 = 1 Abort current command
0324h	R	Fixed Disk Adapter Interrupt Status Register where: Bit 7 = 1 Termination error, bits 0-6 indicate what the error is Bit 6 = 1 Invalid command Bit 5 = 1 Command reject Bit 4-3 = 0 Reserved Bit 2 = 0 Drive 0 selected = 1 Drive 1 selected Bit 1 = 1 Error recovery procedure invoked Bit 0 = 1 Equipment check
0378h	R/W	Parallel 2, data port
0379h	R/W	Parallel 2, status port, where: Bit 7 = 0 Busy Bit 6 = 0 Acknowledge Bit 5 = 1 Out of paper Bit 4 = 1 Printer is selected Bit 3 = 0 Error Bit 2 = 0 IRQ has occurred Bits 1, 0 = Reserved
037Ah	R/W	Parallel 2, control port, where: Bit 7-6 = Reserved Bit 5 = 0 Direction is write to port = 1 Direction is read from port Bit 4 = 1 Enable IRQ Bit 3 = 1 Select printer Bit 2 = 0 Initialize printer Bit 1 = 1 Automatic line feed Bit 0 = 1 Strobe
03B4h	R/W	VGA CRT controller index register (mono)
03B5h	R/W	Other VGA CRT controller registers (mono)
03BAh	R	VGA input status register 1 (mono)
03BAh	W	VGA feature control register (mono)

continued

Hardware I/O Port List, Continued

Table: Hardware I/O Port Definitions, cont'd

I/O Address	Read/Write Status	Description
03BCh	R/W	Parallel 1, data port
03BDh	R/W	Parallel 1, status port, where: Bit 7 = 0 Busy Bit 6 = 0 Acknowledge Bit 5 = 1 Out of paper Bit 4 = 1 Printer is selected Bit 3 = 0 Error Bit 2 = 0 IRQ has occurred Bits 1, 0 = Reserved
03BEh	R/W	Parallel 1, control port, where: Bit 6 = Reserved Bit 5 = 0 Direction is write to port = 1 Direction is read from port Bit 4 = 1 Enable IRQ Bit 3 = 1 Select printer Bit 2 = 0 Initialize printer Bit 1 = 1 Automatic line feed Bit 0 = 1 Strobe
03C0h	R/W	VGA attribute address register
03C0h	W	Other VGA attribute registers
03C1h	R	Other VGA attribute registers
03C2h	W	VGA miscellaneous output register
03C2h	R	VGA input status register 0
03C3h	R/W	VGA video subsystem enable
03C4h	R/W	VGA sequencer address register
03C5h	R/W	Other VGA sequencer registers
03C6h	R/W	Video DAC PEL mask
03C7h	W	Video DAC PEL address, read mode
03C7h	R	Video DAC state register
03C8h	R/W	Video DAC PEL address, write mode
03C9h	R/W	Video DAC PEL data register
03CAh	R	VGA feature control register
03CCh	R	VGA miscellaneous output register
03CEh	R/W	VGA graphics registers, address register
03CFh	R/W	VGA and other graphics registers

continued

Hardware I/O Port List, Continued

Table: Hardware I/O Port Definitions, cont'd

I/O Address	Read/Write Status	Description
03D4h	R/W	VGA CRT controller index register (color)
03D5h	R/W	Other VGA CRT controller registers (color)
03DAh	R	VGA input status register 1 (color)
03DAh	W	VGA feature control register (color)
03F0h	R	Diskette controller status register A, where: Bit 7 = 1 Interrupt pending Bit 6 = 0 Second drive installed Bit 5 = 1 Step Bit 4 = 1 Track 0 Bit 3 = 1 Head 1 select Bit 2 = 0 Index Bit 1 = 0 Write protect Bit 0 = 0 Data received by controller
03F1h	R	Diskette controller status register B, where: Bit 7-6 = 0 Reserved Bit 5 = Drive select Bit 4 = Write data Bit 3 = Read data Bit 2 = Write enable Bit 1 = 1 Motor enable 1 Bit 0 = 1 Motor enable 0
03F2h	W	Diskette controller digital output register, where: Bit 7-6 = 0 Reserved Bit 5 = 1 Motor enable 1 Bit 4 = 1 Motor enable 0 Bit 3 = 0 Allow interrupts Bit 2 = 0 Controller reset Bit 1 = 0 Reserved Bit 0 = 0 Drive select 0 = 1 Drive select 1
03F4h	R	Diskette controller status register, where: Bit 7 = 1 Data register is ready Bit 6 = 1 Transfer is from controller to system = 0 Transfer is from system to controller Bit 5 = 1 Non-DMA mode Bit 4 = 1 Diskette controller busy Bit 3-2 = Reserved Bit 1 = 1 Drive 1 busy Bit 0 = 0 Drive 0 busy
03F5h	R/W	Diskette controller data registers

continued

Hardware I/O Port List, Continued

Table: Hardware I/O Port Definitions, cont'd

I/O Address	Read/Write Status	Description
03F7h	R	Diskette controller digital input register where: Bit 7 = Diskette change Bits 6-1 = Reserved Bit 0 = 0 High density select
03F7h	W	Diskette controller configuration control register, where: Bits 7-2 = Reserved Bits 1-0 = 00b 500 kbs mode = 01b Reserved = 10b 250 kbs mode = 11b Reserved
03F8h	W	Serial 1, transmitter holding register, where: Bits 7-0 = Data bits 7-0, respectively, when Divisor Latch Access Bit (DLAB) = 0
03F8h	R	Serial 1, receiver buffer register, where: Bits 7-0 = Data bits 7-0, respectively, when DLAB = 0
03F8h	R/W	Serial 1, divisor latch, low byte, where: Bits 7-0 = Bits 7-0 of divisor, when DLAB = 1
03F9h	R/W	Serial 1, divisor latch, high byte, where: Bits 7-0 = Bits 15-8 of divisor, when DLAB = 1
03F9h	R/W	Serial 1, interrupt enable register, where: Bits 7-4 = 0 Reserved Bit 3 = 1 Modem status interrupt enable Bit 2 = 1 Receiver line status interrupt enable Bit 1 = 1 Transmitter holding register empty interrupt enable Bit 0 = 1 Received data available interrupt enable when DLAB = 0
03FAh	R	Serial 1 Interrupt ID Register, where: Bits 7-3 = 0 Reserved Bits 2-1 = 00b Modem status interrupt 01b Transmitter holding register empty interrupt 10b Received data available register interrupt 11b Receiver line status interrupt Bit 0 = 0 Interrupt pending

continued

Hardware I/O Port List, Continued

Table: Hardware I/O Port Definitions, cont'd

I/O Address	Read/Write Status	Description
03FAh	W	<p>Serial 1, FIFO Control Register, where:</p> <p>Bits 7-6 = Receiver FIFO register trigger 00b = 1 byte 01b = 4 bytes 10b = 8 bytes 11b = 14 bytes</p> <p>Bits 5-3 = 0 Reserved</p> <p>Bit 2 = Transmitter FIFO register reset 1 = transmit FIFO register cleared, counter cleared, bit is self-clearing</p> <p>Bit 1 = Receiver FIFO register reset 1 = receiver FIFO register cleared, counter cleared, bit is self-clearing</p> <p>Bit 0 = FIFO enable 1 = receiver and transmitter FIFOs enabled, must be 1 to program FIFO registers 0 = clears receive and transmit FIFO registers, enters character mode</p>
03FBh	R/W	<p>Serial 1, Line Control Register, where:</p> <p>Bit 7 = 0 Receiver Buffer, Transmitter Holding, or Interrupt Enable Registers Access = 1 Divisor Latch Access</p> <p>Bit 6 = 1 Set Break Enabled</p> <p>Bit 5 = Stick Parity</p> <p>Bit 4 = Even Parity Select</p> <p>Bit 3 = Parity Enable</p> <p>Bit 2 = 0 1 Stop Bit = 1 0 Stop Bits</p> <p>Bits 1-0 = 00b 5 Bit Word Length 01b 6 Bit Word Length 10b 7 Bit Word Length 11b 8 Bit Word Length</p>
03FCh	R/W	<p>Serial 1, Modem Control Register, where:</p> <p>Bits 7-5 = 0 Reserved</p> <p>Bit 4 = 1 Loopback mode</p> <p>Bit 3 = 1 Enable OUT2 interrupt</p> <p>Bit 2 = 1 Force OUT1 active</p> <p>Bit 1 = 1 Force request to send active</p> <p>Bit 0 = 1 Force data terminal ready active</p>

continued

Hardware I/O Port List, Continued

Table: Hardware I/O Port Definitions, cont'd

I/O Address	Read/Write Status	Description
03FDh	R/W	Serial 1, Line Status Register, where: Bit 7 = 0 Reserved Bit 6 = 1 Transmitter shift and holding registers empty Bit 5 = 1 Transmitter holding register empty Bit 4 = 1 Break interrupt Bit 3 = 1 Framing error Bit 2 = 1 Trailing edge ring indicator Bit 1 = 1 Overrun error Bit 0 = 1 Data ready
03FEh	R	Serial 1 Modem Status Register, where: Bit 7 = 1 Data carrier detect Bit 6 = 1 Ring indicator Bit 5 = 1 Data set ready Bit 4 = 1 Clear to send Bit 3 = 1 Delta data carrier detect Bit 2 = 1 Trailing edge ring indicator Bit 1 = 1 Delta data set ready Bit 0 = 1 Delta clear to send
03FFh	R	Serial 1, Scratch register
0680h	W	Manufacturing checkpoint port
3220-3227h	See 03F8h-03FFh	Serial Port 3 (see description for addresses 03F8h-03FFh for details).
3228-322Fh	See 03F8h-03FFh	Serial Port 4 (see description for addresses 03F8h-03FFh for details).
4220-4227h	See 03F8h-03FFh	Serial Port 5 (see description for addresses 03F8h-03FFh for details).
4228-422Fh	See 03F8h-03FFh	Serial Port 6 (see description for addresses 03F8h-03FFh for details).
5220-5228h	See 03F8h-03FFh	Serial Port 7 (see description for addresses 03F8h-03FFh for details).
5228-522Fh	See 03F8h-03FFh	Serial Port 8 (see description for addresses 03F8h-03FFh for details).

Chapter 3

BIOS Data Structures

Overview

Description

BIOS makes use of data structures to link BIOS services/functions to the operating system. The data structures reside in system memory and are initialized during BIOS initialization. The BIOS data structures are the

- Common Data Area,
 - Function Transfer Tables, and
 - Device Blocks.
-

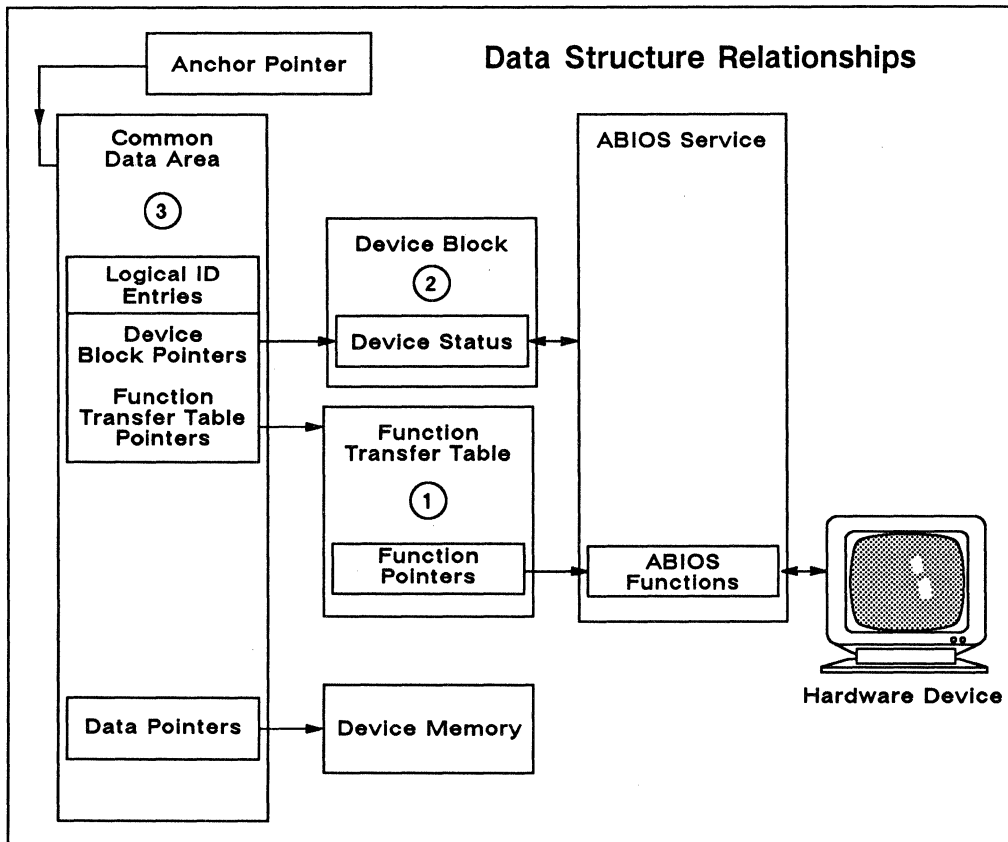
continued

Overview, Continued

Data structure relationships

Each BIOS service has a Function Transfer Table and a Device Block associated with it.

1. The Function Transfer Table contains address pointers to each BIOS function.
2. The Device Block is used by the BIOS to store interrupt levels, device status information, and hardware port addresses.
3. The Common Data Area contains a linked list of the Function Transfer Table and Device Block pointer pairs associated with each service, as well as device memory pointers (if any) associated with a given device.



continued

Overview, Continued

In this chapter

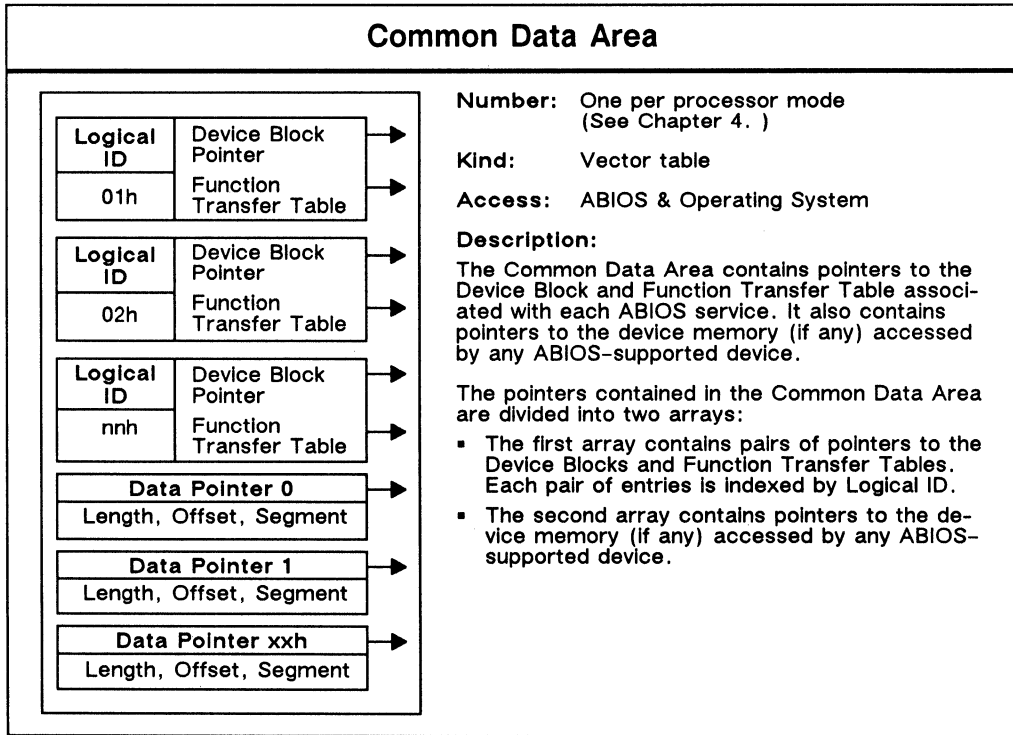
This chapter defines the internal components of each kind of BIOS data structure. The following topics are discussed:

- Common Data Area
 - Function Transfer Tables
 - Device Blocks
 - Related Information
-

Common Data Area

Introduction

Created during BIOS initialization, the Common Data Area (CDA) is provided so that operating systems that use both the real and protected address modes (or a combination of both) of the system microprocessor can be implemented. The main features of the Common Data Area are outlined below:



Real mode/protected mode CDAs

The Common Data Area acts as a link between all BIOS pointers. It gathers them all in one structure, allowing an operating system to manage the BIOS requests in both real and protected modes. There must be a Common Data Area in both the real mode and the protected mode if both address modes are used by the operating system.

continued

Common Data Area, Continued

The Anchor Pointer

Every time the operating system makes a request of the BIOS, it passes a segment or selector (the offset is assumed to be zero) to the BIOS. This selector points to the Common Data Area and is called the Anchor Pointer to the Common Data Area.

Device IDs and Logical IDs

Device IDs are physical device identifiers that are attached to each device known to the BIOS. Each Device ID may have one or more Logical IDs. A Logical ID is a device handle used by the operating system to make a call to the BIOS. Logical IDs are assigned during BIOS initialization.

Device Data pointers

The Common Data Area data pointers point to system data areas such as the CBIOS Data Area. Additional data pointers may be defined and used by individual services. The data areas pointed to by the data pointers are permanently resident in RAM data areas reserved by the operating system.

Null entries into the Common Data Area

If the Function Transfer Table Pointer and the Device Block Pointer Fields are both set to 0000:0000h after initialization, the associated Logical ID must be thought of as a null Common Data Area entry. The entire entry for this Logical ID is used as a temporary place holder at initialization time.

continued

Common Data Area, Continued

Common Data Area structure

Offset	Bytes	Description
00h	2	OFFSET TO DATA POINTER 0 Combined with the Anchor Pointer, is a pointer to the Data Pointer 0 Length field.
02h	2	COUNT OF LOGICAL IDS Number of Device Block and Function Transfer Table pointer pairs
04h	4	Reserved
08h	4	Pointer to Device Block for Logical ID 2.
0Ch	4	Pointer to Function Transfer Table for Logical ID 1.
10h	4	Pointer to the Device Block for Logical ID 2.
14h	4	Pointer to the Function Transfer Table for Logical ID 2.
(08h*n)	4	Pointer to the Device Block for Logical ID n.
(08h*n) + 04h	4	Pointer to the Function Transfer Table for Logical ID n.
(08h*n) + 08h	2	LENGTH OF THE INDICATED DATA POINTER Data pointer length of the data areas indicated by an associated data pointer.
(08h*n) + 0Ah	2	OFFSET OF THE INDICATED DATA AREA Combined with associated data pointer segment, indicates the location of the beginning of the data area.
(08h*n) + 0Ch	2	DATA AREA POINTER SEGMENT Combined with its associated offset, indicates the location of the beginning of the data area.
(08h*n) + 0Eh	2	Length of the data pointer for pointer p - 1.
(08h*n) + 10h	2	DATA AREA OFFSETS Combined with an associated segment to construct the pointer to the data area for pointer p - 1.
(08h*n) + 12h	2	DATA AREA SEGMENTS Combined with an associated data area offset to construct the pointer to the data area for pointer p - 1.
(08h*n) + (06h*p) + 08h	2	Length of data pointer 0.
(08h*n) + (06h*p) + 0Ah	2	DATA AREA OFFSET Combined with an associated segment to construct the pointer to the data area.
(08h*n) + (06h*p) + 0Ch	2	DATA AREA SEGMENTS Combined with an associated data area offset to construct the pointer to the data area.
(08h*n) + (06h*p) + 0Eh	2	The total number of data pointers.
<p>Note: n is the number of Logical IDs. p is the number of Data Pointers minus one.</p>		

Function Transfer Table

Description

The Function Transfer Table is a table of vectors to BIOS entry points. Each entry contains the four-byte address pointer for each BIOS function. Reserved function pointers are initialized to 0000:0000h.

Each logical ID, or each entry in the Common Data Area, has a Function Transfer Table Pointer. Multiple Logical IDs can have Function Transfer Table Pointers that point to the same Function Transfer Table and thus establish its relationship with the Common Data Area.

The main features of the Function Transfer Table are outlined below:

Function Transfer Table	
Start Routine	Vector →
Interrupt Routine	Vector →
Time-out Routine	Vector →
Count of functions in service	
Function 00h	Vector →
Function 01h	Vector →
Function 02h	Vector →
Function 03h	Vector →
Function 04h	Vector →
Function 05h	Vector →
Function 06h	Vector →
Function 07h	Vector →
Function 08h	Vector →
Function 09h	Vector →
Function nnh	Vector →
Number:	One per BIOS service and one per processor mode used (See Chapter 4 in this volume.)
Kind:	Double Word Vector table
Access:	BIOS and operating system
Description:	The Function Transfer Table contains a list of double word vectors that supply the entry points into one BIOS service. The first three entries in each BIOS Function Transfer Table are vectors to the start, interrupt, and time-out routines contained in an BIOS service. The remaining entries in each Function Transfer Table are vectors to specific BIOS functions. These functions are referred to by function number in the Request Block Function field. The start and interrupt routines check for valid entry conditions before transferring control to the function requested. The time-out routine ends a function that does not receive an interrupt in a specified amount of time.

continued

Function Transfer Table, Continued

Function Transfer Table structure

The structure of the Function Transfer Table is described in the following table:

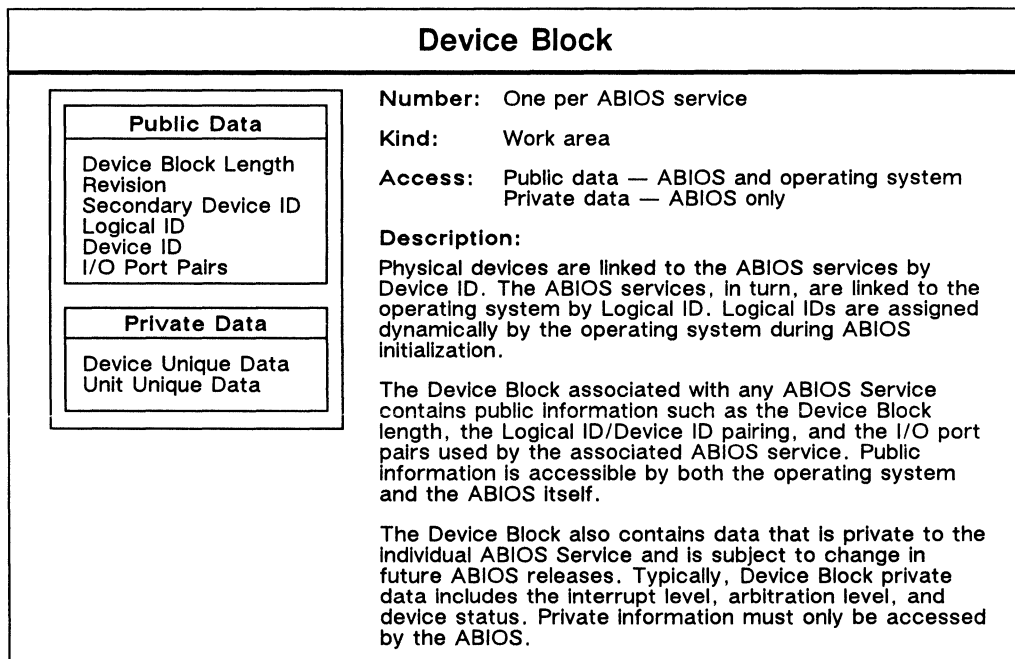
Offset	Bytes	Description
00h	4	START ROUTINE POINTER Pointer to a routine that validates the Function field, Request Block Length field, and the Unit field. The routine saves all registers before this field is used and restores them after it returns. The routine is called using a CALL FAR indirect to start an BIOS request.
04h	4	INTERRUPT ROUTINE POINTER Pointer to a routine that resumes all multistaged requests if the operation is not completed in one request. The routine saves all registers before this field is used and restores them after it returns. If this Function Transfer Table corresponds to a device that does not interrupt, the Interrupt Routine Pointer Field is initialized to 0000:0000h. This routine is called using a CALL FAR indirect to resume a multistaged request after an interrupt is cleared.
08h	4	TIME-OUT ROUTINE POINTER Pointer to a routine that aborts the request, and, as a result, sets the affected hardware controller in a known state. The routine saves all registers before this routine executes and restores them after it returns. The Time-out Routine is called using a CALL FAR indirect. It is used to terminate a request that does not receive a hardware interrupt within a specified time period. If this Function Transfer Table corresponds to a device that does not generate interrupts, or a device that generates interrupts but never times out, the Time-out Routine Pointer field must be initialized by the operating system to 0000:0000h.
0Ch	2	FUNCTION COUNT Contains a count of the number of functions supported by a device.
0Eh	2	Reserved
10h	4	Pointer to the Function 1 Routine.
14h	4	Pointer to the Function 2 Routine.
0Ch + (4*n)	4	Pointer to the Function n Routine.
Note: n is equal to the number of functions specified in this Function Transfer Table.		

Device Block

Description

Each BIOS device service requires a permanent work area for storing device interrupt levels, device status information, and hardware port addresses. This work area is called the Device Block.

The main features of the Device Block are summarized in the graphic below:



Public data

Public Data in the Device Block

- is readable,
- has a format common to all Device Blocks, and
- should not be changed by the caller.

continued

Device Block, Continued

Private data

Private data in the Device Block

- is used internally by the BIOS,
- may not have the same format from one Device Block to another,
- may not contain the same kind of data from one Device Block to the next, and
- should not read or be written to by the caller.

Private data in the Device Block may be altered in future BIOS releases.

Device Block structure

The structure of the Device Block is described in the following table.

Offset	Type	Bytes	Description
00h	Public	2	DEVICE BLOCK LENGTH BIOS returns the required size of the Device Block here during BIOS initialization. The maximum length is 65,535 bytes.
02h	Public	1	REVISION Indicates the BIOS version/revision level.
03h	Public	1	SECONDARY DEVICE ID Indicates the hardware level that BIOS supports. When this field is incremented, the Revision Field reverts to zero.
04h	Public	2	LOGICAL ID The logical identifier of the device associated with this Device Block. Logical IDs are set at BIOS initialization time. The Logical ID for a given device is determined by the index of its entry into the Common Data Area. The operating system must reserve the first n Logical IDs for BIOS use. The number of Logical IDs to be reserved for BIOS use is passed to the operating system at BIOS initialization time via the Initialization Table. (A table describing how the BIOS Logical IDs are used follows this table.)
06h	Public	2	DEVICE ID The type of device that the function request addresses, or the BIOS internal call, is specified in this field. (A table of valid BIOS Device IDs follows.)

continued

Device Block, Continued

Device Block structure, cont'd

Offset	Type	Bytes	Description
08h	Public	2	<p>COUNT OF LOGICAL ID EXCLUSIVE PORT PAIRS Logical ID exclusive port pairs are two ports (input and output) used by a certain Logical ID. For example, I/O ports 0061h and 0062h are used by the diskette services Logical ID.</p> <p>If the entry in this field is zero, no space is allocated for the Logical ID Exclusive Port Pairs Fields. This field contains the number of Logical ID Exclusive Port Pairs.</p>
0Ah	Public	2	<p>COUNT OF LOGICAL ID COMMON PORT PAIRS Logical ID common port pairs are ports that are shared by more than one Logical ID. For example, the DMA controller ports or Keyboard Controller ports are used by other Logical IDs. Each Logical ID that uses one of these ports contains an entry in the Logical ID Common Port pairs fields of the Device Block. If this field is set to zero, no space is allocated for the Logical ID Common Port Pairs Field. This field contains the number of Logical ID Common Port Pairs for this Request Block.</p>
Varies	Private	4	<p>LOGICAL ID EXCLUSIVE PORT PAIRS Contains the Logical ID Exclusive Port Pairs for this Request Block. The first word contains the starting I/O port number; the last word contains the ending I/O port number.</p> <p>Note: Every port that an ABIOS Logical ID writes to or reads from must be accounted for in either the Logical ID Exclusive Port Pairs or Logical ID Common Port Pairs Fields.</p>
Varies	Private	4	<p>LOGICAL ID COMMON PORT PAIRS Points to a list of the I/O Port addresses of the Logical ID Common Port Pairs. The first word contains the starting I/O port number; the last word contains the ending I/O port number.</p> <p>Note: Every port that an ABIOS Logical ID writes to or reads from must be accounted for in either the Logical ID Exclusive Port Pairs or Logical ID Common Port Pairs Fields.</p>
Varies	Private	2	<p>DEVICE-UNIQUE DATA AREA LENGTH Contains the length in bytes of the Device-Unique Data Area for this device.</p>

continued

Device Block, Continued

Device Block structure, cont'd

Offset	Type	Bytes	Description
Varies	Private	Varies	DEVICE-UNIQUE DATA AREA Information about arbitration levels, interrupt levels, and device status is stored in this field. Parameters that describe the device and working data that concerns this Device ID are stored in this field. The information in this field is data that is unique to this particular device. The content and format of the information in this field are Private Data and may change.
Varies	Private	2	COUNT OF UNITS The number of Unit Unique Data Area fields in the Device Block is stored in this field. If the entry is zero, this field is the last field in the Device Block.
Varies	Private	2	UNIT-UNIQUE DATA AREA LENGTH The length in bytes of a single entry in the repeatable Unit-Unique Data Area, not including the Unit-Unique Data Area Length, is stored here. This field will be here only if the Count of Units field is greater than zero.
Varies	Private	Varies	UNIT-UNIQUE DATA AREA This repeatable BIOS Private Data area is reserved for each unit of the Device ID. For example, if the diskette Logical ID is 01h, a particular diskette drive will be the Unit. This area contains parameters describing the unit and the working data for individual requests. The contents and format of the data in this field may change, since it is BIOS Private Data. This field will be here only if the Count of Units Field is greater than zero.

Note: n is the last Logical ID reserved for BIOS internal calls. This value will vary depending on how many Logical IDs BIOS must reserve for internal use, extending, and adding functions. BIOS passes this value to the operating system at BIOS initialization by noting it in the initialization table.

continued

Device Block, Continued

Table of Logical ID Values

Logical ID	How used . . .
00h	Reserved for use by BIOS.
01h	Reserved for use by BIOS.
02h-xxh	Reserved for use by BIOS. xxh is specified by the operating system in the Initialization Table when the BIOS is initialized. xxh is the number of BIOS logical IDs.
greater than xxh	System board ROM, adapter ROM, RAM extension.

Device IDs

The following table lists all valid BIOS Device IDs.

Device ID	Device Type/Service	Device ID	Device Type/Service
00h	BIOS Internal Calls	0Bh	Pointing Device
01h	Diskette	0Ch	Reserved
02h	Fixed Disk	0Dh	Reserved
03h	Video	0Eh	CMOS RAM
04h	Keyboard	0Fh	Direct Memory Access
05h	Parallel Port	10h	Programmable Option Select
06h	Serial Port	11h	Error Log
07h	System Timer	12h-15h	Reserved
08h	Real Time Clock Timer	16h	Keyboard Security
09h	System Services	17h-FFFFh	Reserved
0Ah	Nonmaskable Interrupt		

Related Information

Data structure initialization

The discussion of BIOS data structures presented in this chapter is confined to a detailed presentation of data structure internal components. No mention was made, however, of how these structures are initialized.

For more information on how BIOS Data Structures are initialized, refer to Chapter 4.

Request Blocks

In addition to the BIOS data structures, the BIOS also interfaces with RAM-resident parameter blocks called Request Blocks.

The BIOS caller must initialize one Request Block per function call. The input, output, and status parameters associated with a given function call are stored in the Request Block dedicated to that call. Request Blocks associated with completed function calls can be reused.

For a complete discussion of Request Block structure, turn to Chapter 5.

Transfer conventions

Each time an BIOS device function is called, the caller must build a function-specific Request Block, push a set of pointers to the BIOS Data Structures onto the stack, and invoke the appropriate function entry routine.

BIOS functions can be called via either of two transfer conventions: the operating system transfer convention and the BIOS transfer convention. The difference between these two conventions lies in the stack information and the entry routine called.

For more information on how BIOS functions are called, turn to Chapter 7.

Chapter 4

BIOS Initialization

Overview

Initialization facts

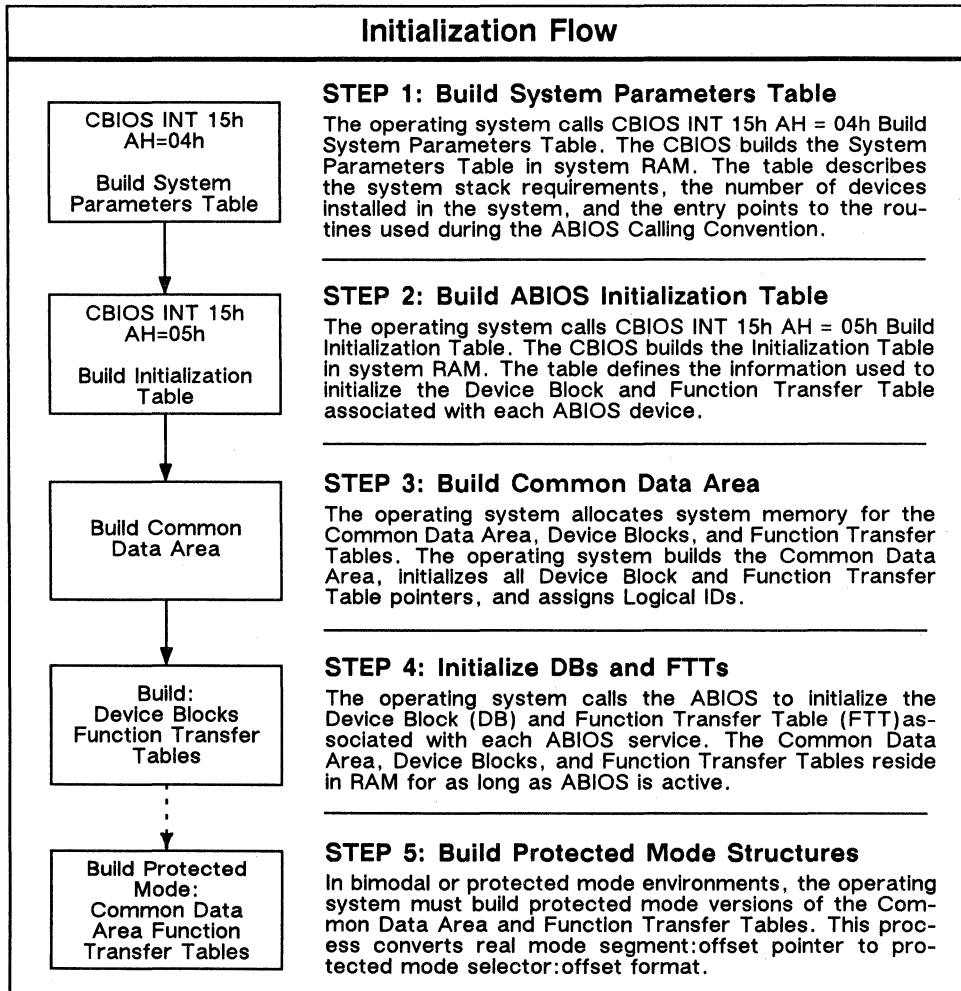
Before BIOS can be used, it must be initialized. Here are the key facts surrounding BIOS initialization:

- Before BIOS can be initialized, CBIOS must be initialized and the operating system must be booted.
 - BIOS can only be initialized in the microprocessor's real mode.
 - Initializing BIOS is largely a matter of initializing the BIOS data structures.
 - In bimodal environments, the operating system must initialize parallel sets of Common Data Areas and Function Transfer Tables.
-

continued

Initialization steps

ABIOS initialization flow is illustrated below.



continued

Overview, Continued

About Request Blocks

Request Blocks, which are initialized by the operating system as each function request arises, are not built as part of the BIOS initialization process.

For more information on Request Blocks, please refer to Chapter 5.

In this chapter

This chapter details each of the steps required to initialize BIOS. The following topics are discussed:

- Step 1: Build System Parameters Table
 - Step 2: Build BIOS Initialization Table
 - Step 3: Build the Common Data Area
 - Step 4: Initialize Device Blocks/Function Transfer Tables
 - Step 5: Build Protected Mode Tables
 - Initializing Logical ID 2: BIOS Internal Calls
 - How BIOS Supports Multiple Instances of a Device
-

Step 1: Build the System Parameters Table

Introduction

In step 1 of the BIOS initialization process, the operating system calls the CBIOS INT 15h AH = 04h Build System Parameters Table function.

The System Parameter table describes the number of devices available in the system, the BIOS common entry points and the system stack requirements.

Once the BIOS initialization process is complete, the operating system may deallocate the memory space occupied by the System Parameters Table.

System Parameter table

The structure of the System Parameter Table is listed below:

Offset	Length	Description
00h	4 Bytes	COMMON START ROUTINE POINTER Contains the address of the Common Start Routine entry point.
04h	4 Bytes	COMMON INTERRUPT ROUTINE POINTER Contains the address of the Common Interrupt Routine entry point.
08h	4 Bytes	COMMON TIME-OUT ROUTINE POINTER Contains the address of the Common Time-out Routine entry point.
0Ch	2 Bytes	STACK REQUIRED Contains the number of bytes of stack memory that is required for the particular BIOS implementation.
0Eh	4 Bytes	RESERVED
12h	4 Bytes	RESERVED
16h	4 Bytes	RESERVED
1Ah	4 Bytes	RESERVED
1Eh	2 Bytes	NUMBER OF ENTRIES The BIOS gathers configuration information by polling the Common Data Area and by scanning for the presence of ROM and RAM extensions to the BIOS.

continued

Step 1: Build the System Parameters Table, Continued

Operating system tasks

To build the System Parameters table, the Operating System must

1. Allocate 20h bytes for CBIOS to build the system parameters table.
 2. Call CBIOS INT 15h, function AH = 04h, Build System Parameters Table.
 3. Optionally deallocate the memory used for the System Parameters Table after the BIOS initialization process is complete.
-

CBIOS Function: INT 15h AH = 04h Build System Parameters Table

The CBIOS Build System Parameters Table function:

1. Establishes the value of the System Parameters Table's Number of Initialization Table Entries field by examining configuration information in CMOS RAM, the system equipment data area, and by testing for the presence of those devices with operating code that resides in system ROM.
 2. Scans in 2K increments absolute addresses C0000h to DF800h to determine the presence of adapter ROM (if any), updating the Number of Initialization Table Entries field accordingly.
 3. Scans the RAM extension area pointed to by the parameter passed by the operating system in the DS register to count the number of entries (if any) to add to the Number of Initialization Table Entries field.
 4. Builds the System Parameter Table in the 20h byte location allocated by the operating system.
-

Input/Output

Input: AH = 04h
ES:DI= Pointer to 20h byte location in RAM where the table is to be built.
DS = Segment where BIOS RAM Extensions reside. If there are no RAM extensions, this register must be set to 0000h. (BIOS RAM extension offset is assumed to be 0000h)

Output: System Parameter Table is output to RAM
AH = 00h If function was successful
CF = 1 Exception error has occurred
All registers except AX and flags are restored.

Step 2: Build BIOS Initialization Table

Introduction

Once the System Parameters Table is built, the operating system must call CBIOS INT 15H function AH = 05h to build the BIOS Initialization Table.

The information contained in the Initialization Table is referred to later in the BIOS initialization process in two ways:

- In step 3, the operating system refers to the BIOS Initialization Table to determine the size of the Common Data Area, as well as the size of each Device Block and Function Transfer Table.
 - In step 4, the BIOS refers to the Initialization Table when initializing Device Blocks and Function Transfer Tables.
-

Initialization Table Structure

The Initialization Table is made up of fields of 18h bytes. The information in each field is repeated for each device found by the CBIOS INT15h AH = 05h Build BIOS Initialization Table function. The entries making up each field are described in the following table.

Offset	Length	Description
00h	2 Bytes	DEVICE ID The same Device ID can appear more than one time within the Initialization table. Device IDs are listed in Chapters 1 and 3.
02h	2 Bytes	NUMBER OF LOGICAL IDs The maximum number of Logical IDs used by the operating system is stored in this field. This field stores the maximum number of devices that require Device Blocks but are used by the same code.
04h	2 Bytes	DEVICE BLOCK LENGTH Contains the allocation of storage needed by the Device Block for this device. If a Device Block Length is zero, this entry is for an BIOS patch or extension. A zero length entry means that a Device Block need not be built for this entry. If the Device Block Length is zero, the operating system must initialize the Device Block Pointer in the Common Data Area to 0000:0000h. See Chapter 7 BIOS Extensions for more detail on modifying the BIOS.

continued

Step 2: Build the Initialization Table, Continued

Initialization Table Structure, cont'd

Offset	Length	Description
06h	4 Bytes	<p>INITIALIZE DEVICE BLOCK AND FUNCTION TRANSFER TABLE ROUTINE POINTER</p> <p>Contains the address, in real address mode, segment:offset format, of the routine that will initialize the Device Blocks and Function Transfer Tables for an entry in the Initialization Table. Adapter ROMs and RAM extensions must also provide this routine to patch, add, extend, or replace services. See Chapter 7 ABIOS Extensions for more detail on modifying ABIOS functions.</p>
0Ah	2 Bytes	<p>REQUEST BLOCK LENGTH</p> <p>Contains the length of the Request Block required for this device. Any Request Block size equal or greater to this size is valid when making a request to the ABIOS.</p>
0Ch	2 Bytes	<p>FUNCTION TRANSFER TABLE LENGTH</p> <p>Contains the Function Transfer Table length, in bytes. If the Function Transfer Table length is zero,</p> <ul style="list-style-type: none"> ▪ this entry is for a patch, ▪ the data area for the Function Transfer Table need not be allocated, and ▪ the operating system must initialize the Function Transfer Table Pointer field in the Common Data Area to 0000:0000h.
0Eh	2 Bytes	<p>DATA POINTERS LENGTH</p> <p>Contains the required length of the data pointer fields in the Common Data Area.</p>
10h	1 Byte	<p>SECONDARY DEVICE ID</p> <p>Indicates the level of hardware that this ABIOS version supports.</p>
11h	1 Byte	<p>REVISION</p> <p>Indicates the device driver revision level that ABIOS supports for this device.</p>
12h	2 Bytes	RESERVED
14h	2 Bytes	RESERVED
16h	2 Bytes	RESERVED

continued

Step 2: Build the Initialization Table, Continued

Operating system tasks

The operating system must perform the following steps to help build the Initialization Table:

1. Allocate an area of memory where the CBIOS will build the Initialization Table. The area must be 18h times the number of entries in the Initialization Table. The information about the number of entries in the Initialization Table can be found in the System Parameters Table under the Number of Entries field.
2. Call INT 15h, function AH = 05h, Build Initialization Table.

Memory allocated for the Initialization Table may be reused by the operating system after the initialization process is complete.

CBIOS Function: INT 15h AH = 05h Build Initialization Table

The CBIOS Build Initialization Table function must:

1. Use the following order of precedence for Initialization Table entries:
 - a. system board devices
 - b. adapter ROM devices (CBIOS uses POST for this search)
 - c. extension RAM devices.
 2. Construct the Initialization Table using information found in system ROMs, adapter ROMs (if any), and the RAM extensions (if any).
-

Input/Output

Input: AH = 05h
ES:DI= Pointer to location in RAM where the table is to be built.
DS = Segment of the Extended System RAM data area.
(Offset is assumed to be 0000h)

Output: A BIOS Initialization Table is output to RAM
AH = 00h If function was successful
CF = 1 Exception error has occurred
All registers except AX and flags are restored.

Step 3: Build the Common Data Area

Introduction

Once the System Parameters Table and BIOS Initialization Table are built, the operating system has all the information it needs to build the Common Data Area.

Operating system tasks

The Operating System builds the Common Data Area in the following sequence:

1. Allocates memory at offset 0 within a segment for the Common Data Area. The operating system saves a pointer (called the Anchor Pointer) to this segment.
2. Allocates memory within the Common Data Area for pointers to the Device Block and Function Transfer Table associated with each BIOS device service.
3. Allocates memory for pointers to the data area (if any) associated with each BIOS device service.
4. Initializes the offset to the Data Pointer 1 Field so that it points to the Data Pointer Length 0 Field in the Common Data Area.
5. Initializes the Data Pointer Count Field to zero.
6. Places the number of Device Block and Function Transfer Table pointer pairs in the Count of Logical IDs Field in the Common Data Area.
7. Initializes each Device Block Pointer to point to the memory location that has been allocated for that particular Device Block.
8. Initializes each Function Transfer Table Pointer to point to the memory location that has been allocated for that particular Function Transfer Table.
9. Assigns a Logical ID to each pair of Device Block/Function Transfer Table pointers.

continued

Step 3: Build the Common Data Area, Continued

Logical IDs

A Logical ID identifies the logical name of the device associated with one Device Block.

Logical ID numbering

During step 3 of the BIOS initialization process, the operating system assigns Logical ID numbers ordinally, according to the rules defined in the table below:

Logical ID	Description
00h	Reserved for use by BIOS.
01h	Reserved for use by BIOS.
02h-xxh	Reserved for BIOS Internal Calls. Logical IDs 2 to nnh are reserved for BIOS internal calls. xxh is the number of the last Logical ID reserved for BIOS internal calls. Logical ID 2 is always defined. All remaining entries in the Initialization Table with Device ID 0 must be initialized as reserved for BIOS internal calls. This insures support for patches to the BIOS Common Routines.
greater than xxh	Reserved for BIOS Services. The starting Logical ID assigned to an BIOS device must be one greater than nnh. System board, adapter devices, RAM extension devices are assigned Logical ID numbers in this range.

Device 00h and Logical ID 02h

Device 00h is not associated with a particular hardware device but is provided to allow for BIOS internal calls.

In the BIOS system ROM, code associated with the BIOS Common Entry Routines is contained in the Device 00h service. The operating system always assigns Logical ID 02h to the system ROM Device 00h service.

For more on Common Entry Routines

Control can be optionally transferred to any given BIOS service via the Common Entry Routines. A complete discussion is found in Chapter 6.

continued

Step 3: Build the Common Data Area, Continued

Device 00h and Logical IDs 02h + xxh

Subsequent instances, if any, of Device 00h correspond to BIOS extensions designed to patch the Common Entry Routines contained in the system ROM Device 00h service. The operating system ordinarily assigns a Logical ID to each subsequent Common Data Area entry with Device 00h.

Logical ID nnh

Once it has assigned Logical IDs to Device 00h, the operating system assigns a Logical ID, ordinarily, to each pair of Device Block/Function Transfer Table pointers in the Common Data Area.

It is possible for a single BIOS service to be assigned more than one Logical ID. The number of Logical IDs assigned to a service lies in the relationship between the number of hardware devices that will access the service's code, and the interrupt and/or arbitration level associated with each device.

For more on multiple Logical IDs

For more information on multiple Logical IDs, see How BIOS Supports Multiple Instances of a Device later in this chapter.

Step 4: Initialize Device Blocks and Function Transfer Tables

Introduction

Whether it is in system ROM, adapter ROM, or in an BIOS RAM extension, each BIOS service contains a routine to initialize the Device Block (DB) and Function Transfer Table (FTT) associated with that service. Once it has built the Common Data Area, the operating system calls each of these routines.

Operating system tasks

For each field in the Initialization Table, the operating system must

1. load the CX, DX, and DS registers with the parameters required for entry into the Device Block and Function Transfer Table initialization routine associated with the Initialization Table.
 2. invoke the initialization routine contained in the Initialize Device Block and Function Transfer Table Routine pointer entry for the field under consideration.
 3. after the initialization routine completes, convert all 32-bit Data Pointers contained in the Common Data Area to segment:offset address format.
-

Input/Output

Here is a definition of the input parameters that must be passed, and the output parameter that is returned.

Input: CS:IP= Pointer to Initialization Routine
CX = Number of Logical IDs to initialize. This number must be less than or equal to the number of Logical IDs allowed by the Number of Logical IDs field in the Initialization Table.
DX = Starting Logical ID
DS = Anchor Pointer to Common Data Area (Segment value: Offset is assumed to be 0000h)

Output: AL = 00h Success
01h Device initialization failure.

All other registers are preserved.

continued

Step 4: Initialize Device Blocks and Function Transfer Tables, Continued

Initialization routine processing

In initializing Device Blocks and Function Transfer Tables, BIOS performs the following actions:

Step	Description
1.	<p>COMPLETES THE FUNCTION TRANSFER TABLE</p> <p>When the operating system calls the Initialize Device Block and Function Transfer Table Routine, BIOS Completes the Function Transfer Table.</p> <ul style="list-style-type: none"> ▪ The Function Transfer Table is completed at the location defined by the Function Transfer Table pointer in the Starting Logical ID parameter in Register DX. ▪ When the Initialize Device Block and Function Transfer Table routine is called for adapter ROMs or RAM extensions, each segment value that is placed in the Function Transfer Table must equal the segment of its corresponding adapter ROM header or RAM extension header. ▪ When the headers and segment values are equal, an operating system can access the Length Field (given in 512-byte blocks) of either the adapter ROM or RAM extension header so that the segment limit can be determined in either bimodal or protected mode environments. ▪ When building the protected mode Common Data Area, if offset 0 of the ROM header segment or RAM extension header segment contains the RAM or ROM signature, offset 2 will contain the length of the segment (in 512-byte blocks up to a maximum of 7Fh 512-byte blocks). The operating system can use this value to calculate the segment limit.
2.	<p>FILLS IN THE DEVICE BLOCK</p> <ol style="list-style-type: none"> a. After completing the Function Transfer Table, the Initialize Device Block and Function Transfer Table Routine completes the Device Block for the Starting Logical ID Parameter (which is contained in DX). b. The Initialize Device Block and Function Transfer Table Routine builds a Device Block for each Logical ID up to the Number of Logical IDs To Initialize Parameter (this value is contained in CX). It may affect multiple services.
3.	<p>WRITES 32-BIT DEVICE DATA POINTERS TO THE COMMON DATA AREA</p> <ol style="list-style-type: none"> a. BIOS stores any Data Pointers associated with an BIOS service (if any) into the appropriate Data Pointer field in the Common Data Area. b. BIOS increments the Data Pointer Count field as each Data Pointer is stored. c. BIOS stores a handle to the Data Pointer in the Device Block. This handle is an offset to the location in the Common Data Area where the Data Pointer is stored.

continued

Step 4: Initialize Device Blocks and Function Transfer Tables,

Continued

How the operating system handles error conditions

If the Return Code Parameter (in AL) is not zero upon return from the Initialize Device Block and Function Transfer Table Routine, the operating system should deallocate the associated Device Blocks and Function Transfer Table Areas and replace the associated Device Block pointers and Function Transfer Table pointers with 0000:0000h, making all affected entries into null Common Data Area entries.

Order of initialization

The Initialize Device Block And Function Transfer Table Routines are called in the exact order that their respective pointers appear in the Initialization Table. The order of initialization is system board ROMs, adapter ROM extensions, and RAM extensions.

Because BIOS services contained in system board ROMs are initialized first, the Initialize Device Block and Function Transfer Table routines for adapter ROM and RAM extension devices can then easily identify the system board device services they need.

The initialization routines associated with adapter ROMs and BIOS RAM extensions must scan the Common Data Area using the Device ID in the public portion of the Device Block to identify the system board services that they need. Once the Device ID is found, the Logical ID, stored in the public portion of the Device Block, should be used for all additional requests to the system board BIOS service.

BIOS services and initialization

The operating system is free to decide not to initialize an BIOS service based on the model, device, and revision information for each Initialization Table entry (OS/2 uses model and submodel number). If the operating system decides that this information indicates that the BIOS code is not appropriate for current hardware, then the initialization of these BIOS routines may not be performed. Initialization routines are also free not to initialize themselves.

continued

Step 4: Initialize Device Blocks and Function Transfer Tables,

Continued

Support for BIOS extensions

To properly support BIOS extensions, the operating system must insure that all entries in the Initialization Table with the same Device ID and Secondary Device ID are initialized.

Logical ID 02h initialization

In the BIOS system ROM, the code associated with the BIOS Common Entry Routines is contained in the Device 00h service. The operating system always assigns Logical ID 02h to the system ROM Device 00h service. The process of initializing Logical ID 02h is a separate subject in itself and is discussed further in this chapter under the heading "Logical ID 02h Initialization."

Physical addresses must be converted to segment:offset

Unless they are altered by the operating system, all Data Pointer fields contained in the Common Data Area remain in 32-bit physical address format. Before calling the BIOS, the operating system must translate the 32-bit physical address of each data pointer into segment:offset format.

Data Pointer/Pointer length facts

Here are the most important things to remember about data pointers and data pointer lengths:

- Data Pointers are stored in the BIOS service as 32-bit physical addresses in Intel low word/high word format.
 - The Initialization Routine initializes the Data Pointer field with this 32-bit physical address.
 - All Data Pointer fields in the Common Data Area are preceded by a Data Pointer Length Field.
 - The Data Pointer Length field specifies the segment limit for a protected mode or bimodal environment.
 - In a bimodal environment, if the Data Pointer Length field is 0000:0000h in the real mode Common Data Area, then an address above 1 MB is assumed.
-

Step 5: Build Protected Mode Tables

Introduction

In bimodal or protected mode environments, protected mode versions of the real mode Common Data Area and Function Transfer Tables must be built by the operating system. The operating system can use the information already gathered in the real mode Common Data Area and Function Transfer Tables to build these protected mode tables.

Building the protected mode Common Data Area

The protected mode Common Data Area and the protected mode Function Transfer Tables pointers are stored in protected mode selector:offset format. The operating system must initialize all protected mode pointers so that their effective addresses are identical to their real mode, segment:offset counterparts.

The operating system must perform the following the steps in order to build the protected mode Common Data Area:

Step	Description
1	Build the real mode Common Data Area.
2	Allocate memory for the protected mode Common Data Area.
3	Allocate memory for the protected mode Function Transfer Tables.
4	Convert each real mode Device Block Pointer to a protected mode Device Block Pointer.
5	Create a protected mode Function Transfer Table Pointer for each protected mode Function Transfer Table.
6	Convert each real mode Function Pointer within each real mode Function Transfer Table to a protected mode Function Pointer.
7	Convert each real mode Data Pointer to a protected mode data pointer.

continued

Step 5: Build Protected Mode Tables, Continued

Building selector descriptors

To build descriptors associated with each selector, the operating system must know

- the actual physical address of the descriptor,
 - the access rights of each segment, and
 - the segment limits of each segment.
-

Pointer characteristics

The Function Transfer Table pointers and the Device Block pointers are data segment descriptors that can be written to by the operating system. The expansion direction and limit of each pointer must be maintained by the operating system.

Function Transfer Table length and Device Block length are returned to the operating system via the Initialization Table.

Each data pointer's selector must reference a data segment descriptor that can be written to and has an expansion direction of *up*. Segment limit is stored in the Data Pointer Length Field associated with each Data Pointer entry in the Common Data Area.

All Function Transfer Table pointers must be readable code segment descriptors. The Conforming Bit for each pointer is set by the operating system.

If offset zero of either the ROM header segment or the RAM extension header segment contains the ROM/RAM signature, then offset 2 contains the length (in 512-byte increments with a limit of 7Fh) of the segment. This value should be used as the segment limit by the operating system.

The segment limit for ROM/RAM signatures that do not exist must be FFFFh.

Note: A complete description of both ROM and RAM extension headers is found in Chapter 7.

continued

Step 5: Build Protected Mode Tables, Continued

ABIOS I/O privileges

The ABIOS must have I/O privilege at all times. Insuring ABIOS I/O privilege is the responsibility of the operating system, particularly when the operating system calls the ABIOS using multiple privilege levels or as a conforming code segment.

Null Common Data Area entries

Each null entry in the real mode Common Data Area must have a corresponding null entry in the protected mode Common Data Area.

When the protected mode version of each Function Transfer Table is initialized, each entry in the protected mode version that has a corresponding real mode entry equal to 0000:0000h must also equal 0000:0000h.

The offset fields in the Function Transfer Table must be the same for the corresponding entries in both the real mode and the protected mode tables.

The Device Block pointers for each Logical ID entry in both the real mode and the protected mode Common Data Areas must point to the same Device Block.

Initializing Logical ID 2: BIOS Internal Calls

Introduction

The data structures associated with Logical ID 2 are reserved for BIOS internal calls.

How reserved Data Pointers are initialized

Data Pointers 0, 1, and 2 in the Common Data area are initialized by Logical ID 2. The operating system initializes reserved Data Pointers in the Common Data Area when the Initialize Device Block and Function Transfer Table Routine is called for Logical ID 2.

The purpose of these data pointers is to allow a single common data pointer to be used in common by several BIOS devices. The reserved data pointers are:

Data Pointer Number	Physical Value	Limit	Description
0	400h	0100h	BIOS Data Area
1	E000:0000h	FFFFh	First 64K of System Board ROM
2	F000:0000h	FFFFh	Second 64K of System Board ROM

How the Function Transfer Table is initialized

When the Initialize Device Block and Function Transfer Table Routine is called for Logical ID 2, this routine must perform the following actions:

Place the address of the...	at...
Common Start Routine	the Start Routine pointer in the Function Transfer Table for Logical ID 2.
Common Interrupt Routine	the Interrupt Routine pointer in the Function Transfer Table for Logical ID 2.
Common Time-out Routine Pointers	the Time-out Routine pointer in the Function Transfer Table for Logical ID 2.

continued

Initializing Logical ID 2: BIOS Internal Calls, Continued

Logical ID 2 Function Transfer Table entry

The Initialization Table entry for the first Device ID that is a zero must have a Function Transfer Table length of 10h or greater, since the first 10h bytes are reserved. The Function Transfer Table length must be more than 10h if there are BIOS internal functions.

The Function Transfer Table entry for Logical ID 2 is described in the following table:

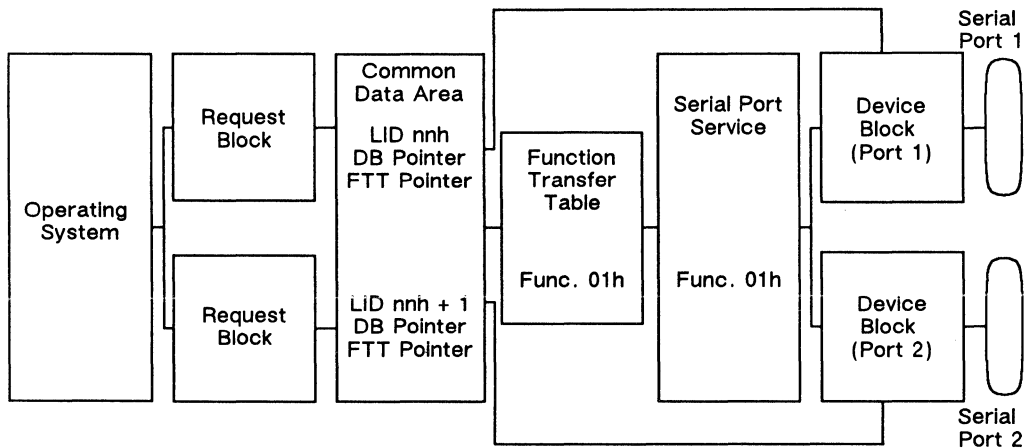
Offset	Length	Description
00h	4 Bytes	Common Start Routine Pointer
04h	4 Bytes	Common Interrupt Routine Pointer
08h	4 Bytes	Common Time-out Routine Pointer
0Ch	2 Bytes	Function Count (should be zero)
0Eh	2 Bytes	Reserved

How BIOS Supports Multiple Instances of a Device

When a service supports more than one device

Once it has initialized all Device ID 00h structures, the operating system moves on to nonzero Device IDs, assigning Logical IDs ordinarily. Occasionally, an BIOS service supports more than one instance of a device. The number of Device Blocks required by the service depends on how the individual devices are distinguished. There are two cases.

Case 1: Separate interrupt and arbitration levels



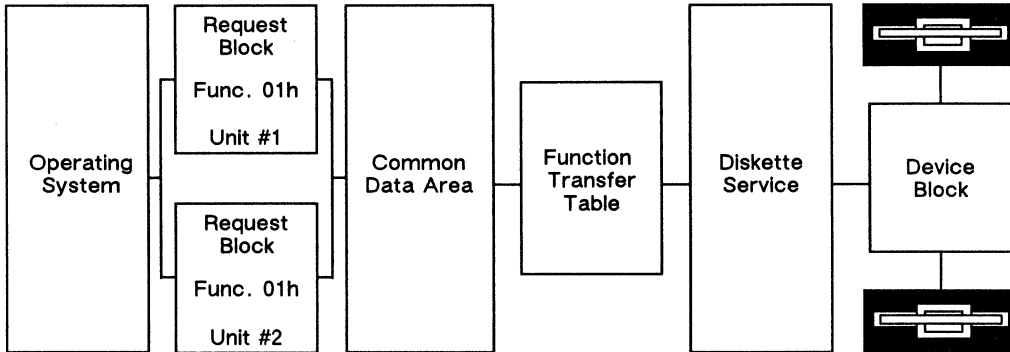
When multiple copies of a device require separate interrupt or arbitration levels, each device requires its own Device Block. Thus each device is logically identified with a separate Logical ID. Even though they require multiple Device Blocks and Logical IDs, each device must share the same device driver code. As such, the Function Transfer Table pointer associated with each Device Block can be initialized to point to the same structure. In other words, device services that support separate interrupt and arbitration levels still only require one Function Transfer Table.

The BIOS Serial Port Service, for example, supports up to eight physical serial ports. Since each serial port requires a separate interrupt and arbitration level, each serial port is associated with a separate Logical ID. The BIOS Serial Port Service interfaces with up to eight Device Blocks; all Serial Port Service functions, however, are located by one Function Transfer Table.

continued

How BIOS Supports Multiple Instances of a Device, Continued

Case 2: Share interrupt and arbitration levels



When multiple copies of a device share the same interrupt and arbitration levels, they are distinguished from each other by the Device Block unit number field. Device Services of this type require only one Device Block, and one function Transfer Table.

The BIOS Diskette Service supports two physical diskette drives. Each diskette drive is identified by unit number.

Related Information

Request Blocks

In addition to the BIOS data structures, the BIOS also interfaces with RAM-resident parameter blocks, called Request Blocks.

The BIOS caller must initialize one Request Block per function call. The input, output, and status parameters associated with a function call are stored in the Request Block dedicated to that call. Request Blocks associated with completed function calls can be reused.

For a complete discussion of Request Block structure, turn to Chapter 5.

Transfer conventions

Each time an BIOS device function is called, the caller must build a function-specific Request Block, push a set of pointers to the BIOS Data Structures onto the stack, and invoke the appropriate function entry routine.

BIOS functions can be called via either of two transfer conventions: the operating system transfer convention and the BIOS transfer convention. The difference between these two conventions lies in the stack information and the entry routine called.

For more information on how BIOS functions are called, turn to Chapter 6.

BIOS extensions

The discussion of BIOS initialization presented in this chapter is kept on a fairly generic level. Although some mention was made of the process of initializing BIOS extensions, the details of extension structure and requirements was kept to a minimum.

For more information on how BIOS extensions, refer to Chapter 7.

Chapter 5

Request Block Structure

Overview

Introduction

Request Blocks are parameter blocks resident in system RAM which are created on demand by the operating system. There must be one Request Block per active BIOS function request.

Limits of this discussion

This discussion is limited strictly to a definition of Request Block internal structure. The process of initializing and using Request Blocks is integrally related to the process of calling BIOS functions.

All discussion of Request Blocks beyond the definition of Request Block internal structures is found in Chapter 6 Calling BIOS.

continued

Overview, Continued

Summary of internal structure

Request Block				
Offset	Size	Input:	Output:	
00h	Word	Request Block length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function		
08h	Word	Reserved		
0Ah	Word	Reserved		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	Word	Reserved		
12h	Word	Data pointer 1		
16h	Word	Reserved		
18h	Word	Reserved		
1Ah	Word	Data pointer 2		
1Eh	Word	Reserved		
20h	Word	Relative block address		
24h	Word	Reserved		
28h				
2Ch		Number of blocks read		
2Eh		Caching Bits 7-1 = Reserved Bit 0 = Caching 0 Yes 1 No		

Number: One per function request

Kind: Parameter block

Access: BIOS and Operating System

Description:
All input and output parameters are passed between the operating system and the BIOS function via the Request Block.

Request Blocks are built and controlled by the operating system.

Because Request Blocks reside in system RAM, input and output parameters are designated as offsets from the Request Block segment.

Offsets 00h-0Eh are defined the same way across all BIOS functions. Offsets 10h-xxh bear definitions that are function-specific.

Request Blocks associated with completed functions can be reused.

In this chapter

This chapter describes the internal structure of BIOS Request Blocks. The following topics are presented:

- Request Block Parameters
- Request Block Structure
- Error Codes

Request Block Parameters

Types of Request Block parameters

The Request Block has two types of parameters:

- Functional Parameters
 - Service-Specific Parameters.
-

Functional parameters

Request Block functional parameters are the same for every BIOS service request. Information about the service requested and the device to be called on is placed in these parameters by the caller and sent to the BIOS.

Each parameter is initialized by the caller and must not be changed while the request is being processed by the BIOS, especially through multistaged BIOS requests. The first 10h bytes in a Request Block are reserved for the Functional Parameters. See the Request Block Structure table in this chapter for a list of all Request Block functional parameters.

Service-Specific parameters

Request Block service-specific parameters may change with every BIOS request. The information passed in these parameters varies with each type of BIOS request. See the specific BIOS service explanations in Chapters 8 through 23 for detailed information about the contents of these parameters. See the Request Block Structure table in this chapter for a list of all Request Block service-specific parameters.

Request Block Structure

Request Block structure table

Offset	Input/Output	Bytes	Description																										
FUNCTIONAL PARAMETERS																													
00h	Input	2	REQUEST BLOCK LENGTH Contains the length in bytes of the Request Block. The maximum is 65,535. The caller initializes this field for a specific Logical ID. The BIOS gives the size of the Request Block for a Logical ID in the Initialization Table. Function 01h, Return Logical ID Parameters, also gives the size, but the Request Block may actually be larger than the returned size from this function call.																										
02h	Input	2	LOGICAL ID Specifies the particular device to be operated on for this service request. Used as an index to the Device Block/Function Transfer Table pair located in the Common Data Area.																										
04h	Input	2	UNIT Addresses only a certain unit of a device within a Logical ID. The range of values for this field is specific to each device that is identified as a Logical ID. The number of units attached to a single controller determines how many units are in a Logical ID. Units are numbered from zero, so the maximum unit number is one less than the number of devices attached to the controller associated with this Logical ID.																										
06h	Input	2	FUNCTION The entry in this field requests a particular BIOS function. The following list details the common BIOS functions. Refer to Chapter 8 for an example of how these are used in the Request Block. <table border="1" style="margin-top: 10px;"> <thead> <tr> <th>Function</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>00h</td> <td>Default Interrupt Handler</td> </tr> <tr> <td>01h</td> <td>Return Logical ID Parameters</td> </tr> <tr> <td>02h</td> <td>Reserved</td> </tr> <tr> <td>03h</td> <td>Read Device Parameters</td> </tr> <tr> <td>04h</td> <td>Set Device Parameters</td> </tr> <tr> <td>05h</td> <td>Reset/Initialize</td> </tr> <tr> <td>06h</td> <td>Enable</td> </tr> <tr> <td>07h</td> <td>Disable</td> </tr> <tr> <td>08h</td> <td>Read</td> </tr> <tr> <td>09h</td> <td>Write</td> </tr> <tr> <td>0Ah</td> <td>Additional Data Transfer</td> </tr> <tr> <td>0Bh-FFh</td> <td>Other service-specific functions</td> </tr> </tbody> </table>	Function	Description	00h	Default Interrupt Handler	01h	Return Logical ID Parameters	02h	Reserved	03h	Read Device Parameters	04h	Set Device Parameters	05h	Reset/Initialize	06h	Enable	07h	Disable	08h	Read	09h	Write	0Ah	Additional Data Transfer	0Bh-FFh	Other service-specific functions
Function	Description																												
00h	Default Interrupt Handler																												
01h	Return Logical ID Parameters																												
02h	Reserved																												
03h	Read Device Parameters																												
04h	Set Device Parameters																												
05h	Reset/Initialize																												
06h	Enable																												
07h	Disable																												
08h	Read																												
09h	Write																												
0Ah	Additional Data Transfer																												
0Bh-FFh	Other service-specific functions																												

continued

Request Block Structure, Continued

Request Block structure table, cont'd

Offset	Input/ Output	Bytes	Description
08h	--	2	RESERVED
0Ah	--	2	RESERVED
0Ch	Input/ Output	2	<p>RETURN CODE</p> <p>Indicates the result of the current stage of the BIOS function request. For single-stage requests or those requests in the final stage of a multistage request, this field indicates the final result of the entire function request. For more information, see the Return Code Handling heading in Chapter 3.</p>
0Eh	Output	2	<p>TIME-OUT</p> <p>The expected time that a requested stage will take. The time-out value is used to ensure that no operation or stage of an operation takes too much microprocessor time.</p> <p>This field is valid if the Return Code is Resume Stage after Time Delay (Bit 1 is set).</p> <p>This field is structured as follows:</p> <p>Bits 15-3 = Units of time before time-out (in seconds)</p> <p>Bits 2-0 = Reserved</p>
SERVICE SPECIFIC PARAMETERS			
10h	Output	2	RESERVED
12h	Input	4	<p>DATA POINTER 1</p> <p>Contains a pointer to an I/O buffer used for this BIOS request. The effective address, a 16-bit segment in the high word and a 16-bit offset in the low word, must be addressable from within the current mode of the microprocessor (real or protected) when in a bimodal environment.</p> <p>The address may be:</p> <ul style="list-style-type: none"> ▪ a 32-bit physical address for DMA, or ▪ a segmented address for programmed I/O. <p>The Return Logical ID Parameters Function (01h) returns a parameter (Transfer Data Pointer Mode) that indicates the mode (physical or logical) of the data pointer for the</p> <ul style="list-style-type: none"> ▪ Read (08h), ▪ Write (09h), and ▪ Additional Data Transfer (0Ah) functions.

continued

Request Block Structure, Continued

Request Block structure table, cont'd

Offset	Input/Output	Bytes	Description
12h (cont'd)			<p>DATA POINTER 1, continued</p> <p>If the Transfer Data Pointer Mode Parameter specifies that this must be a logical pointer, Data Pointer 1 is a logical pointer and Data Pointer 2 is reserved.</p> <p>If the Transfer Data Pointer Mode Parameter specifies that this must be a physical pointer, Data Pointer 2 is a physical pointer and Data Pointer 1 is reserved.</p> <p>If the Transfer Data Pointer Mode parameter specifies that both a logical pointer and a physical pointer will be passed, Data Pointer 1 is the logical pointer and Data pointer 2 is the physical pointer.</p> <p>If the Transfer Data Pointer Mode parameter specifies none of the above, this Logical ID does not support functions 08h, 09h, and 0Ah, or these functions do not need address pointers, and no space will be allocated in the Request Block for any data pointers.</p>
16h	--	2	RESERVED
18h	--	2	RESERVED
1Ah	Input	4	<p>DATA POINTER 2</p> <p>See Data Pointer 1 above.</p>
1Eh-nnh	Input/Output	varies	<p>SERVICE-SPECIFIC PARAMETERS</p> <p>Operands and the results of BIOS functions may be communicated in these fields. Their contents depend on the device service and the requested function. See Chapters 8 through 23 for detailed descriptions, by service and function, of the contents of the service-specific parameter fields.</p>
varies	Private	varies	<p>WORK AREA</p> <p>Optional data area for BIOS use only. User data may not be stored in this field. This field is not initialized. The caller must not alter the contents of this field, particularly while BIOS is processing a multistaged request.</p>

Related Information

Transfer conventions

Each time an ABIOs device function is called, the caller must build a function-specific Request Block, push a set of pointers to the ABIOs Data Structures onto the stack, and invoke the appropriate function entry routine.

ABIOs functions can be called via either of two transfer conventions: the operating system transfer convention and the ABIOs transfer convention. The difference between these two conventions lies in the stack information and the entry routine called.

For more information on how ABIOs functions are called, turn to Chapter 6.

Request Block handling

Each time the caller builds a Request Block, the rules governing Request Block initialization and reuse must be observed. Because the rules for handling Request Blocks are so integrally related to the act of calling ABIOs functions, a complete discussion of the rules for handling Request Blocks are delineated in Chapter 6.

Return Code handling

The specifics of Return Code handling can be viewed as an integral part of calling ABIOs functions. A more detailed discussion of can be found in Chapter 6.

Chapter 6

Calling BIOS

Overview

Introduction

This chapter describes the tasks, conventions, and operating system responsibilities involved in calling BIOS functions.

In this chapter

The following topics are presented:

- BIOS Processing Model
 - Request Block Initialization
 - Transfer Conventions
 - BIOS Transfer Conventions
 - Operating System Transfer Convention
 - Return Code Handling
 - Hardware Interrupt Handlers
 - Default Interrupt Handler
 - Time-out Handler
 - BIOS and Program Access
 - Accessing BIOS via BIOSCommonEntry
 - Accessing BIOS via BIOSCall
-

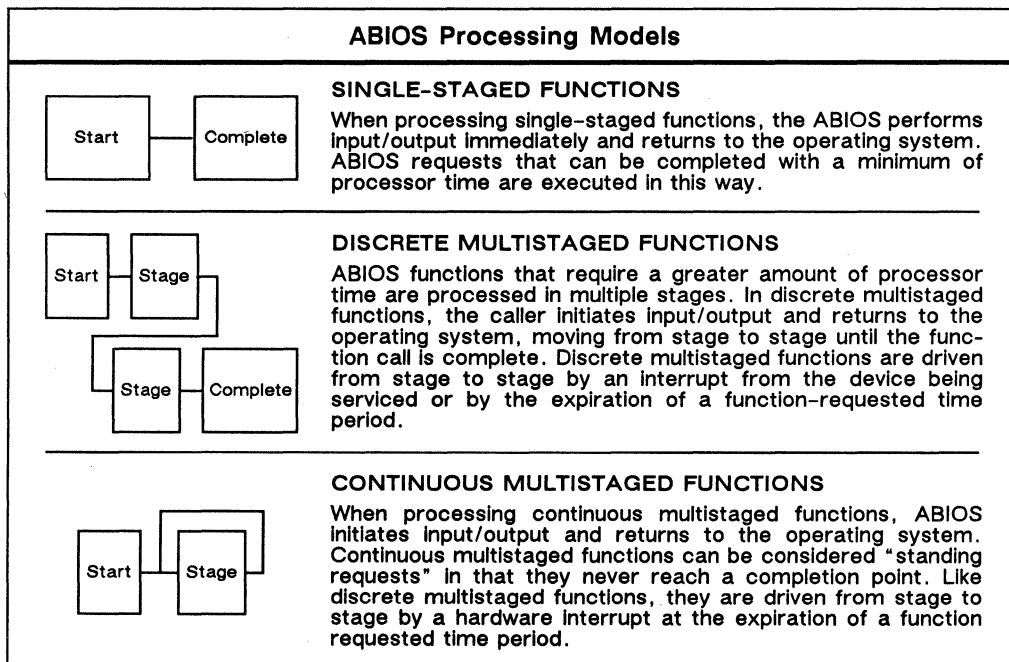
ABIOS Processing Model

Introduction

When an operating system is interfaced with a ROM BIOS, care must be taken to insure that BIOS functions calls do not tie up the processor. In the ABIOS, this is accomplished by processing all functions that could potentially monopolize the processor time in multiple stages.

How ABIOS processes functions

The following illustration depicts the three ways ABIOS function calls are processed.



How functions are interfaced

In order to further liberate the processor, all function parameters are passed between the operating system and the ABIOS function via a function-specific parameter block, called the Request Block. Before starting a given ABIOS function, the operating system must initialize that function's Request Block.

continued

ABIOS Processing Model, Continued

Return Codes: how function status is reported

The BIOS signals the status (successful, unsuccessful, etc.) of a function stage by placing a Return Code in offset 0Ch of the Request Block. The BIOS assumes that all Return Code handling is under the control of the operating system.

What drives multistaged functions

Multistaged functions are driven from stage to stage by either:

- A hardware interrupt or
- The elapse of a function requested period of time.

Some multistaged functions are driven purely by hardware interrupt. Others are driven purely by time period. Others are driven by some combination of both hardware interrupt and time period.

Interrupt-driven stages

Each interrupt-driven BIOS service is associated with one hardware interrupt level. The BIOS assumes that all hardware interrupt handlers are under the control of the operating system.

When a hardware interrupt occurs, the operating system must call the function associated with the interrupt so that the interrupting condition can be serviced. The BIOS resets the interrupting condition at the hardware level. The operating system's hardware interrupt handler must perform end-of-interrupt processing at the interrupt controller level.

A service can have more than one active function request. When this happens, the hardware interrupt handler must call each function until the BIOS replies that the hardware interrupt has been serviced.

continued

Time period driven stages

Some BIOS functions are driven from stage to stage by the elapse of a function-requested period of time. The BIOS assumes that time-period stage handlers are under the control of the operating system. When a time period requested by the function expires, the operating system's time period handler must call the given BIOS function.

Hardware interrupt stages and hardware time-out

All hardware interrupt-driven stages of a function indicate a maximum time (in seconds) to wait for the hardware interrupt. The BIOS assumes that all hardware time-out handlers are under the control of the operating system. Should the hardware time-out period associated with a given interrupt driven function elapse, the operating system is responsible for terminating the function and resetting the hardware.

Hardware time-out vs. time period stages

The terminology surrounding time period-driven stages and hardware time-out handling is similar. However, it is important not to confuse the processing associated with the hardware time-out handling and time-period stage handling.

Hardware time-out handling is associated exclusively with those stages of a multistaged function that are driven by hardware interrupt and is designed to handle function termination cleanly. Execution of a time-out handling routine is symptomatic of a hardware error.

Time-period handling is associated with those stages of a multistaged function that are driven by time periods. Execution of a time-period handling routine indicates the elapse of a function-requested time delay and should not be associated with a hardware error.

continued

ABIOS Processing Model, Continued

How BIOS functions are called

Each BIOS Service is associated with a set of function entry routines. There are three kinds of entry routines:

- **Start Routine**

The start routine associated with a service is called when a function is first started.

- **Interrupt Routine**

The Interrupt Routine associated with a service must be called when the function interrupts or when a time period-driven function requires servicing.

- **Time-out Routine**

The Time-out Routine associated with a service must be called when an interrupt-driven function suffers a hardware time-out.

To reduce caller overhead, the BIOS also contains a set of Common Entry Routines: a Common Start, Interrupt, and Time-out Routine. The Common Entry Routines do some initial processing then transfer control to the entry routine tied to the specific service.

Conventions for calling BIOS

Whether it is a single-staged or a multistaged function, the caller has the option of starting functions via either of two conventions: the Operating System Transfer Convention or the BIOS Transfer Convention. Each transfer convention is described in this chapter.

Program accessibility

The program accessibility of any ROM BIOS depends on the architecture of the operating system interfaced with the BIOS.

Starting with IBM OS/2 Version 1.1, programs have full access to the BIOS through the operating system via the IBM OS/2 **DevHlp** services: **ABIOSCommonEntry** and **ABIOSCall**.

BIOS accessibility for non-IBM versions of OS/2 varies from vendor to vendor. Programmers who are not using IBM OS/2 should refer to their OS/2 documentation to determine if their version of OS/2 supports program access to the BIOS.

Request Block Initialization

Description

The Request Block, resident in system RAM, is a parameter block used to communicate information to and from the calling routine (usually the operating system) and the BIOS. Each BIOS function is interfaced to its caller via a function-specific Request Block. The caller passes input parameters and the BIOS passes output parameters and any other information concerning the completion state of a function through that function's Request Block.

Request Blocks and function requests

The Request Block associated with a function must be initialized by the caller (usually the operating system) before the function can be started.

When the operating system calls more than one function contained in the same BIOS service, it must create one Request Block for each function requested. When each function is entered, the BIOS references the service's Device Block to determine the state of the device as it pertains to the function being processed.

Defining Request Blocks

The Request Block is defined differently for each BIOS Service. The length of a Request Block is fixed for all functions numbered above 01h for a service. Request Block length is checked by BIOS upon entry into an BIOS function.

Request Block lifespan

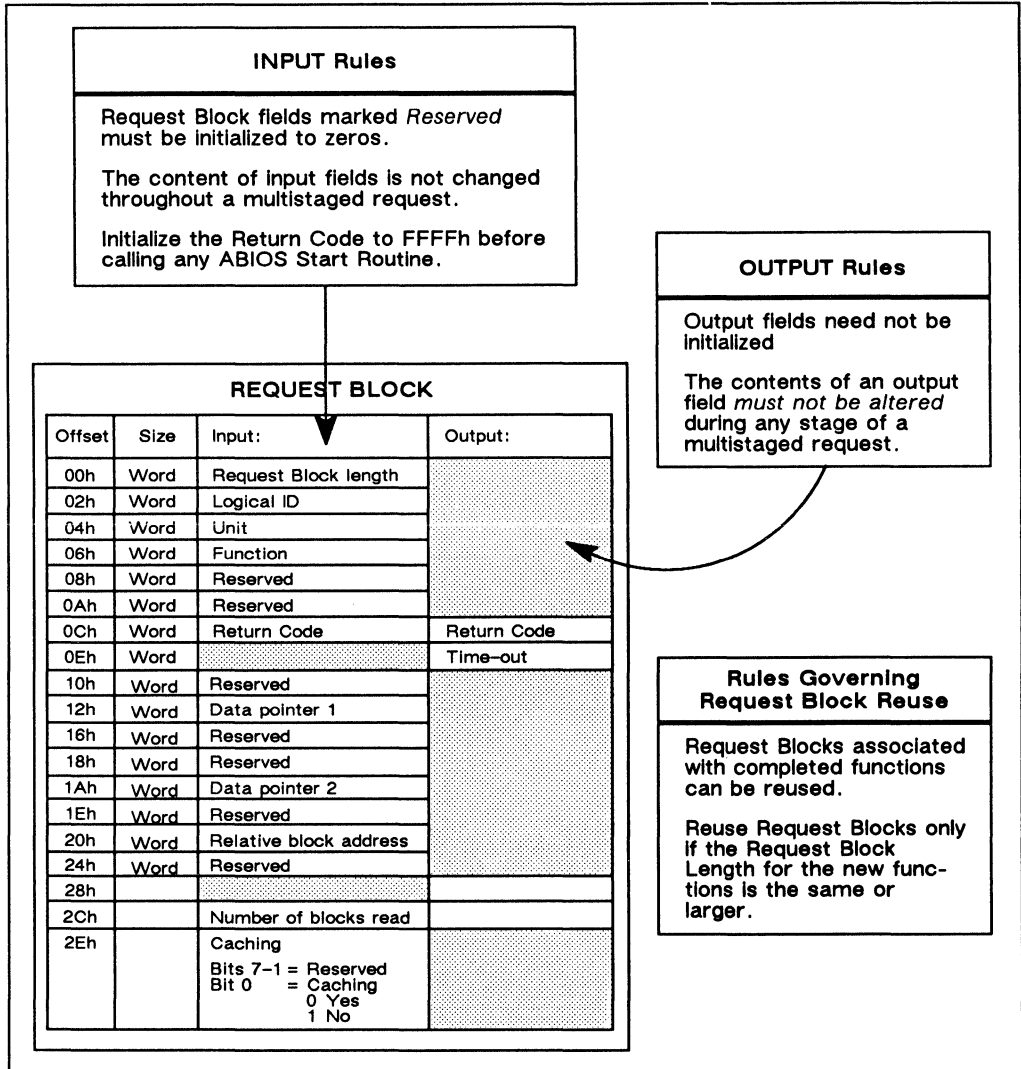
Once initialized, Request Blocks stay resident in system RAM for as long as the function call is active. Those Request Blocks associated with multistaged function stay resident in system RAM for the life of the function. Request Blocks associated with completed functions may be reused, or the memory they occupy can be deallocated. Reuse Request Blocks only if the Request Block length for the new function is the same or longer.

continued

Request Block Initialization, Continued

Rules governing Request Block use

The general rules associated with Request Block use are illustrated below.



A Generalized Look at Control Transfer

Introduction

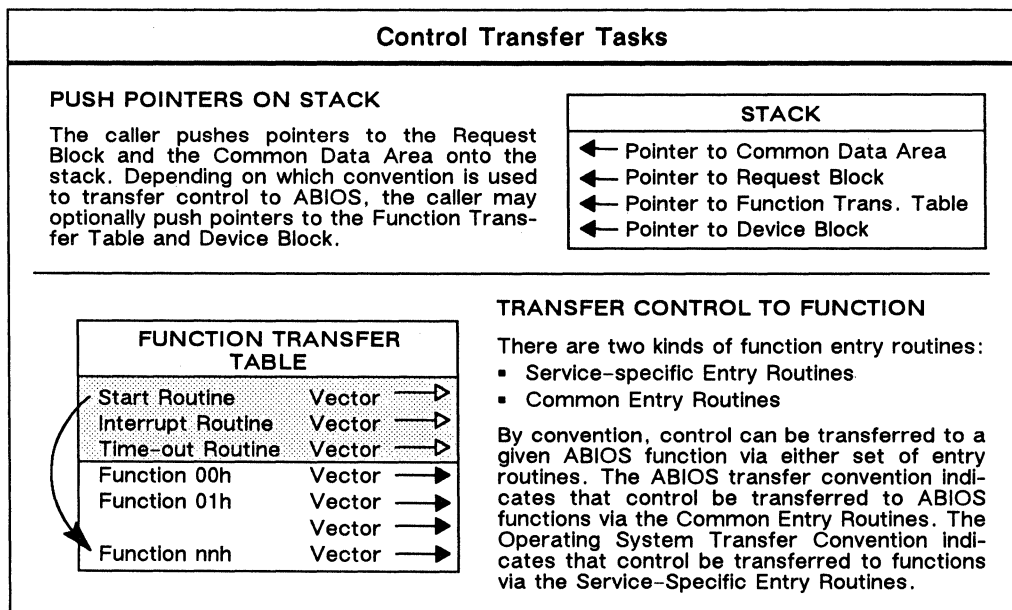
Once its Request Block has been initialized, the caller is free to transfer control to the BIOS function. The process of transferring control to an BIOS function is divided into two basic tasks:

1. PUSH required entry pointers onto stack frame,
2. CALL function entry routine.

Control can be transferred to the function via the BIOS Common Entry Routines or the BIOS Service-Specific Entry Routines. The difference between the two methods of calling BIOS lies in the process used to locate pointers and load them onto the stack.

Two tasks

The graphic below details each control transfer task.



continued

A Generalized Look at Control Transfer, Continued

Common Entry Routine processing

The Common Entry Routines — Start, Interrupt, and Time-out — provide the operating system with a common entry point to the BIOS Service-Specific Entry Routines. The Common Entry Routines:

- Test the Logical ID field contained in the Request Block associated with the function being requested. If an invalid Logical ID is found, the Common Entry Routine returns to the caller with the Return Code field set to C000h Invalid Logical ID. The Logical ID input into the Common Data Area can be invalid for two reasons:
 - The Logical ID is greater than the count of Logical ID field contained in the Common Data Area, or
 - The Logical ID corresponds to a null Logical ID entry in the Common Data Area. (Logical IDs 0 and 1 are reserved).
- Index into the Common Data Area based on the Logical ID and the Anchor Pointer to determine which pair of Device Block and Function Transfer Table pointers are required.
- Copy the Device Block address into the stack placeholder.
- Copy the address into the Function Transfer Table stack placeholder.
- Transfer control to the Service-Specific Entry Routine.

Service-Specific Entry Routine processing

Every BIOS service contains a set of service-specific entry routines — Start, Interrupt, and Time-out. The entry point to each of these routines is contained in the first three entries in the service's Function Transfer Table.

The Service-Specific Entry Routines:

- save the state of the registers,
 - test the Request Block for valid function,
 - test the Request Block for valid Unit Number (if any), and
 - transfer control to function called.
 - The function executes and returns to the operating system with a Return Code, and
 - restores the state of the registers
-

ABIOS Transfer Convention

ABIOS Transfer Convention

Under the ABIOS Transfer Convention, the operating system loads the stack frame with pointers to the caller's return address, Common Data Area, and Request Block. However, only placeholders are supplied for the pointers associated with the function's Device Block and Function Transfer Table.

With the stack frame loaded, control is then transferred to the ABIOS function via its Common Entry Routine. The Common Entry Routine fills in the placeholders in the stack frame then transfers control to the function via the function's Service-Specific Entry Routine.

Advantages

From the caller's point of view, the ABIOS Transfer Convention is the simpler of the two transfer conventions in that it reduces programming overhead when ABIOS functions are called in a bimodal environment.

Caller responsibilities on function return

Once the function returns, the caller is responsible for

- supplying error handling routines, and
 - removing arguments from the stack.
-

Pseudocode

The pseudocode below shows the suggested method for transferring control to an ABIOS function via the ABIOS Transfer Convention.

PUSH	Pointer to Anchor Pointer Segment or Selector
PUSH	Pointer to Request Block Segment or Selector
PUSH	Offset to Request Block
SUB	Stack Pointer, 8
CALL	Common Entry Routine (Start, Interrupt, or Time-out)

Operating System Transfer Convention

The Operating System Transfer Convention

Under the Operating System Transfer Convention, the operating system assumes the responsibility for loading the stack frame with pointers to the caller's return address and to the Common Data Area, as well as to the function's Request Block, its Device Block, and its Function Transfer Table. With the stack frame loaded, control is transferred to the BIOS function via its service-specific entry routine.

Two ways to load the stack frame

The Operating System Transfer Convention supports two methods for loading the stack frame.

- **Direct CDA access method**

The caller indexes directly into the Common Data Area to locate the Device Block and Function Transfer Tables pointers associated with the Logical ID contained in the function's Request Block. This is similar to the BIOS Transfer Convention except that the Common Routines are not used, giving this method a slight performance edge.

- **Stored pointer method**

The caller (i.e. usually the operating system) stores the necessary pointers and loads the stack frame directly without indexing the Common Data Area.

OS Transfer Convention: Advantage #1

The Operating System Transfer Convention involves more programming overhead, but may result in some improvement in speed of execution. This calling convention's main advantage is realized in real mode only or protected mode only operating environments.

Where the BIOS Transfer Convention relies on the Common Entry Routines to index into the Common Data Area to determine which Device Block/Function Transfer Table pointer pair to use, there is a small performance disadvantage to this method in a single mode environment. The Common Data Area is provided as an aid so that functions can be easily located in a bimodal environment.

continued

Operating System Transfer Convention, Continued

OS Transfer Convention: Advantage #2

Whether the operating environment is single or bimodal, the Operating System Transfer Convention is most effective when used to handle interrupts from programmed I/O devices (such as the keyboard controller) that are called repeatedly by a single function.

Caller responsibilities on function return

Once the function returns, the operating system is responsible for:

- supplying error handling routines, and
 - removing arguments from the stack
-

Pseudocode

The pseudocode below shows the suggested method for transferring control to an BIOS function via the Operating System Transfer Convention.

```
PUSH    Pointer to Anchor Pointer Segment or Selector
PUSH    Pointer to Request Block Segment or Selector
PUSH    Offset to Request Block
PUSH    Pointer to Function Transfer Table Segment
        or Selector
PUSH    Offset to Function Transfer Table
PUSH    Pointer to Device Block Segment or Selector
PUSH    Offset to Device Block
CALL    Service-Specific Entry Routine
        (Start, Interrupt, or Time-out)
```

Return Code Handling

Introduction

ABIOS signals the status (successful, unsuccessful, etc.) of a function call by writing a Return Code to offset 0Ch of the function's Request Block. The ABIOS assumes that all Return Code handlers are under the control of the operating system.

Return Code bit testing

The operating system's Return Code Handling routine should always test certain Return Code bits. Depending on the settings for bits 15, 13, 12, and 8, all other bits, *including some bits marked reserved*, may be meaningful.

Return Code bit definitions

The following table lists the primary and secondary definitions for Return Code bit settings.

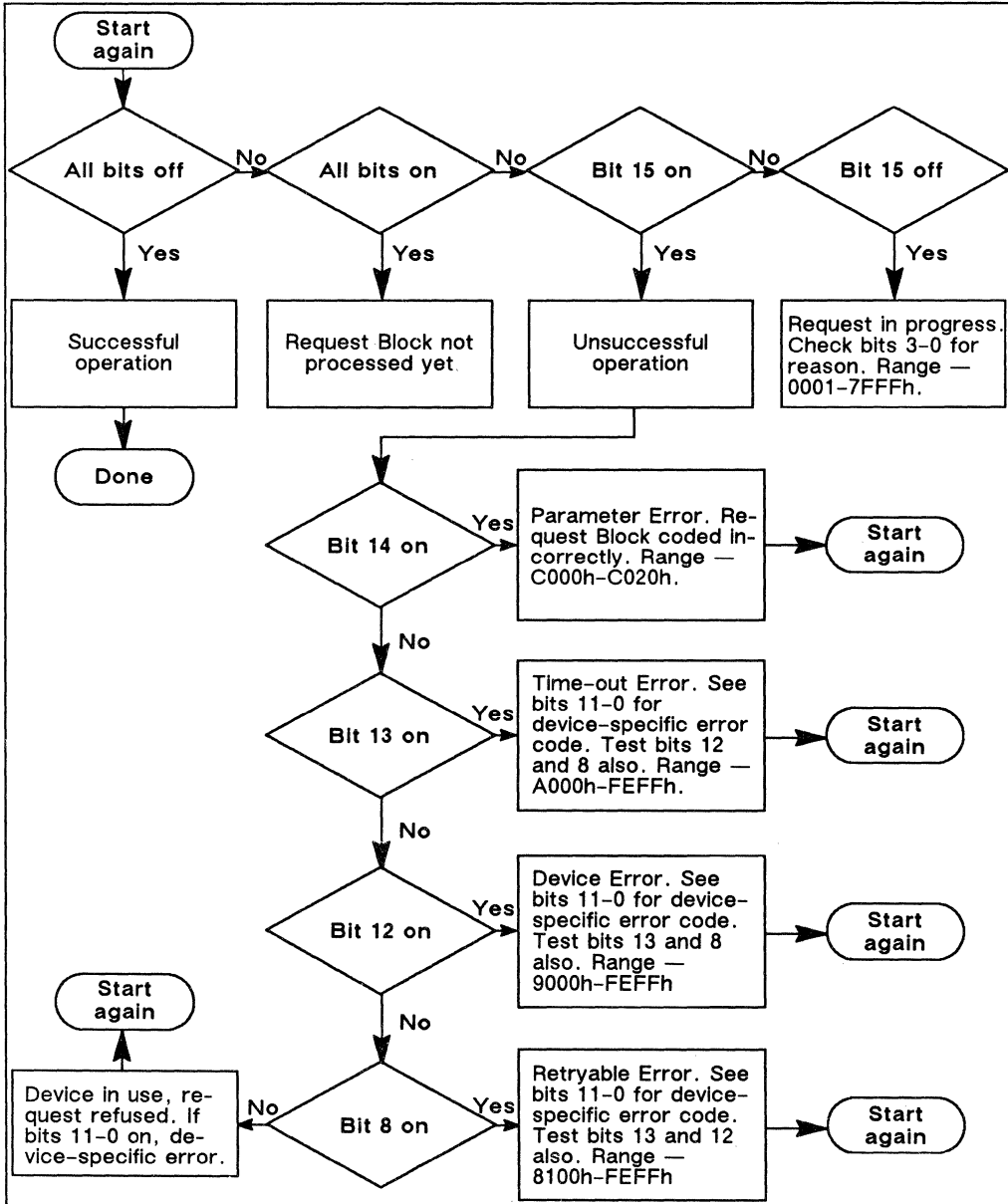
Bits	Primary Definition	Secondary Definition
15	Unsuccessful	None
14	Parameter Error	None
13	Time-out error	None
12	Device Error	None
11-9	Reserved	Device-specific error
8	Retryable Error	Device-specific error
7	Unexpected Interrupt Reset	Device-specific error
6-4	Reserved	Device-specific error
3	Attention	Device-specific error
2	Not My Interrupt	Device-specific error
1	Resume Stage after Time Delay	Device-specific error
0	Resume Stage after Interrupt	Device-specific error
Notes:	<ol style="list-style-type: none">1. Bits 14-8 take the Primary Definition only when Bit 15 is set.2. Bits 11-0 take the Secondary Definition only when Bit 15 is on and any combination of bits 13, 12, and 8 are also on.3. Bits 7-0 take the Primary Definition only when Bit 15 is not on.4. If all bits are on, this Request Block has not been processed yet.	

continued

Return Code Handling, Continued

Flow chart

The following flow chart shows a method of testing the Return Code field.



continued

Return Code Handling, Continued

Return codes

The following table contains a general listing of the BIOS Return Codes. BIOS may generate any value that can occur in a 16-bit BIOS field, so all operating system routines that test BIOS Return Codes should be prepared for any value. That is, each bit in the Return Code field should be tested.

Code	Description
0000h	Successful
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
0005h	Not My Interrupt, Resume Stage after Interrupt
0009h	Attention, Resume Stage after Interrupt
0081h	Unexpected Reset, Resume Stage after Interrupt
8000h	Device in Use, Request Refused
8001h-8FFFh	Service-Specific Unsuccessful Operation
9000h-90FFh	Device Error
9100h-91FFh	Retryable Device Error
9200h-9FFFh	Device Error
A000h-A0FFh	Time-out Error
A100h-A1FFh	Retryable Time-out Error
A200h-AFFFh	Time-out Error
B000h-B0FFh	Device Error With Time-out
B100h-B1FFh	Retryable Device Error With Time-out
B200h-BFFFh	Device Error With Time-out
C000h	Invalid Logical ID
C001h	Invalid Function
C002h	Reserved
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h-C01Fh	Invalid Service-Specific Parameter
C020h-FFFEh	Service-Specific Unsuccessful Operation
FFFFh	Return Code Field Not Valid

Hardware Interrupt Handlers

Introduction

The BIOS assumes that the operating system provides hardware interrupt handlers that receive control via a hardware interrupt vector.

Logical IDs and the hardware interrupt handler

Once BIOS initialization is complete, all BIOS services are identified to the operating system by unique Logical ID. The operating system's hardware interrupt handler must keep track of the Logical IDs that operate on each interrupt level. And it must maintain the validity of all Request Block pointers currently tied to each Logical ID.

When a hardware interrupt occurs, the BIOS Common Entry and Service-Specific Entry Routines, in turn, provide the operating system with a way to transfer control to the given BIOS service.

Request Block polling

When a given hardware interrupt occurs, a Logical ID may have more than one active Request Block. The operating system's hardware interrupt handler must process each active Request Block by calling the BIOS service involved at its Interrupt Routine entry point. Depending on the transfer convention used, this can be done through either the common interrupt routine or through a service-specific interrupt routine.

If bit 0 of the Return Code field is set, the Request Block is waiting for an interrupt. As the hardware interrupt handler polls each Request Block, the BIOS is responsible for setting the Return Code field to indicate whether or not the hardware interrupt was associated with the given Request Block.

The operating system's interrupt handler can call the BIOS with interrupts either enabled or disabled. If interrupts must be disabled, the BIOS restores the state of the interrupt flag when interrupts are disabled and re-enabled.

continued

Hardware Interrupt Handlers, Continued

Initializing the Return Code Field before starting functions

The operating system should always initialize the Return Code field in a Request Block to FFFFh before calling the start routine.

If there is an outstanding Request Block at interrupt time, the operating system must first see if the Return Code field is FFFFh. If it is, the hardware interrupt handler must not attempt to resume this request. In this way, the BIOS can process the function request and set the Return Code appropriately when the interrupt is executed.

Preventing interruption of a routine while it is being started

When a Start Routine transfers control to a function, it does not reset the registers to their pre-call state until the function returns with a valid (i.e. other than FFFFh) Return Code.

When an interrupt occurs before a Start Routine has interrogated the Return Code field and reset the register state, it is possible for the Start Routine and an Interrupt Routine to be accessing the same Request Block from within different stack frames.

To prevent this, the operating system should maintain a flag to indicate whether or not a Start Routine has processed completely. An Interrupt Routine should not be allowed to call a function until its Start Routine has finished execution.

ABIOS processing

If the interrupt is not associated with the Request Block being polled, the BIOS generates the Return Code 0005h Not My Interrupt, Resume Stage After Interrupt.

When the interrupt is associated with the Request Block being polled, the BIOS:

- processes the interrupt,
 - resets the device condition causing the interrupt at the *device level*, The return code handler routine and
 - returns to the caller via the Request Block with an updated Return Code.
-

continued

End-of-interrupt (EOI) processing

Although it resets the interrupting condition at the device, the BIOS does not reset the interrupt controller. Therefore, all end of interrupt processing must be handled by the operating system at the interrupt controller.

Once all Request Blocks for a given Logical ID have been processed through the operating system's hardware interrupt handler and at least one Request Block responds that the interrupt was serviced, the operating system is free to reset the interrupt controller.

The BIOS informs the caller that a Request Block has been serviced by placing any value other than 0005h Not My Interrupt, Resume Stage After Interrupt into the Request Block's Return Code field (offset 0Ch).

Default interrupt handler

If the hardware interrupt handler has called all Request Blocks waiting on an interrupt level and none have "claimed" the interrupt (i.e. sent a Return Code other than 0005h Not My Interrupt, Resume Stage after Interrupt), a rogue interrupt has occurred.

The hardware interrupt handler should call each Default Interrupt Handler for each Logical ID using the interrupt level. For more information on the Default Interrupt Handler see the next topic in this chapter.

Interrupt sharing

No Logical ID can operate on more than one interrupt level. Logical IDs with multiple units operate on the same interrupt level.

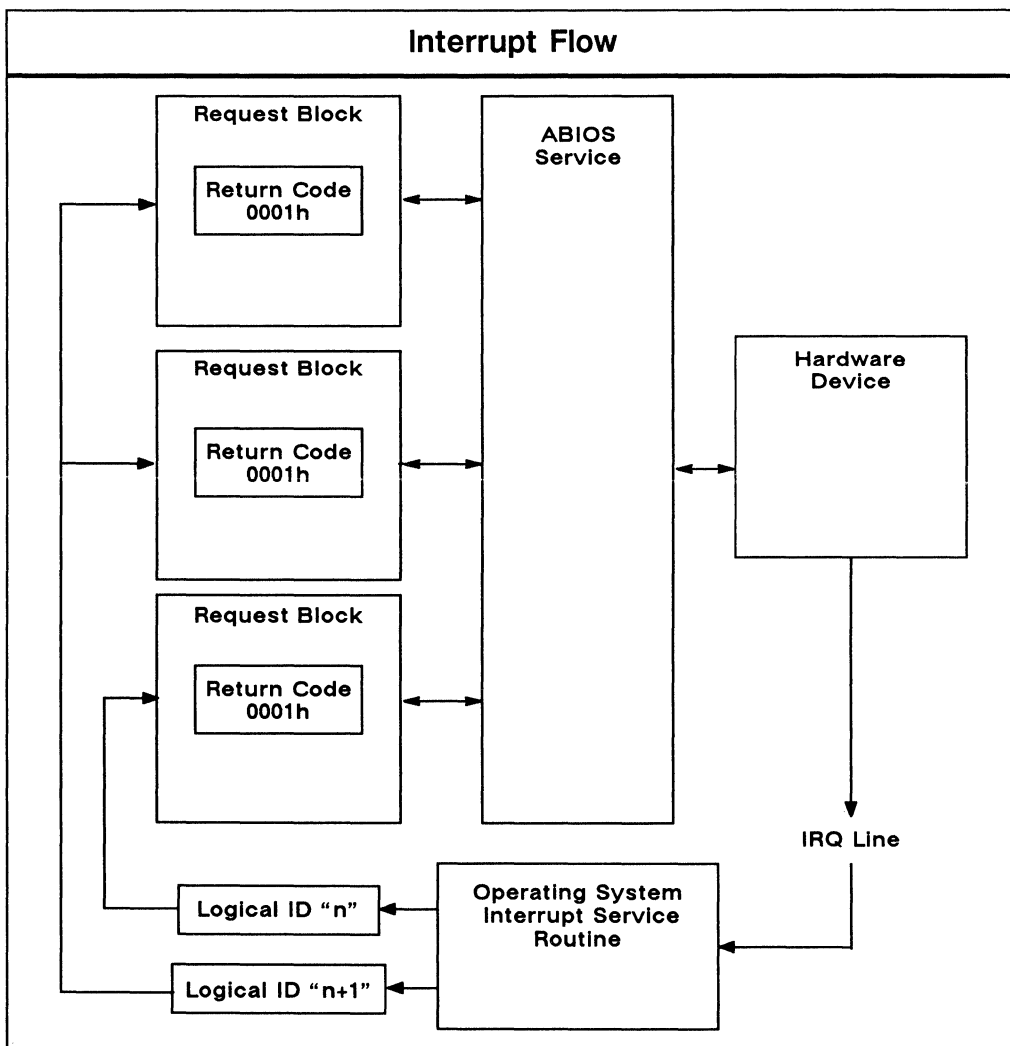
Should more than one Logical ID or Logical ID/Unit combination share the same interrupt level, the hardware interrupt handler associated with that interrupt level should repeat the polling process until all the Logical IDs are processed, or until one Logical ID is processed to completion.

continued

Hardware Interrupt Handlers, Continued

Interrupt sharing

The graphic below depicts the situation where more than one Request Block is active on a given Logical ID.



continued

Hardware Interrupt Handlers, Continued

ABIOS bimodal support for interrupt handlers

When operating in a bimodal environment, it is possible for one Logical ID (i.e. one BIOS service) to be associated with a number of real mode and protected mode function requests.

When this is the case, the BIOS supports hardware interrupt handlers in the following way:

- **Microprocessor mode is maintained**

After a function polls its hardware interrupt handler, the BIOS is returned to the microprocessor mode that was in effect before the hardware interrupt call was made.

- **Interrupt routine can be called bimodally**

When servicing the next Request Block, the hardware interrupt handler calls the interrupt routine from the current mode in effect. The microprocessor mode is hidden from the BIOS. Control is transferred to the next function called whether or not that function was originally started in the current mode. For example, if the system is operating in real mode, the hardware interrupt handler can call a function that was initiated in protected mode.

Default Interrupt Handler

Description

Each BIOS service that uses hardware interrupts contains a Default Interrupt Handler. The Default Interrupt Handler resets the interrupting condition at the device level when a hardware device generates an unexpected interrupt.

Determining which services contain Default Interrupt Handlers

To determine whether or not a given Logical ID (BIOS service) contains a Default Interrupt Handler, place a call to function 01h Return Logical ID Parameters to determine the Logical ID.

The value returned in the Request Block at offset 10h will indicate whether or not the Default Interrupt Handler is supported. The values may be:

Return Code	Description
FFh	Default Interrupt Handler is not supported
FEh	Reserved for NMI Service. Default Interrupt Handler is not supported
nnh	Default Interrupt Handler supported for this Logical ID, where nnh = hardware interrupt level.

How the Default Interrupt Handler is invoked

When an interrupt occurs while there are no active Request Blocks, the operating system must

- build the Default Interrupt Handler's Request Block,
 - load the stack frame with the appropriate pointers,
 - call either the Common Interrupt Routine or Service-Specific Interrupt Routine, depending on the transfer convention being used. The Default Interrupt Handler resets the unexpected interrupt at the device level, and
 - upon return, the operating system performs EOI processing, resetting the interrupt at the interrupt controller.
-

continued

Default Interrupt Handler, Continued

Request Block Structure

The structure of the Default Interrupt Handler's Request Block is described in the table below:

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0000h)	
08h	Word	Reserved (initialize to 0000h)	
0Ah	Word	Reserved (initialize to 0000h)	
0Ch	Word	Return Code	Return Code
0Eh	Word		Time-out

Return Codes

The Default Interrupt Handler Return Codes are listed in the table below:

Code	Description
0000h	Successful Operation
0005h	Not My Interrupt, Resume Stage after Interrupt
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Time-Out Handlers

Introduction

The BIOS informs the operating system of the maximum period it is to wait for a hardware interrupt-driven stage by outputting a value (in seconds) to the the time-out field (offset 0Eh) in the function's Request Block. The BIOS assumes that all hardware time-out handlers are under the control of the operating system.

A time-out value of 00h indicates no time-out checking should be done by the caller. The keyboard is an example of a device for which the BIOS device service is willing to wait forever for the next interrupt.

Time-out handler duties

The operating system's time-out handler must refer to function time-out value and set up a function-specific time-out counter. If the hardware interrupt associated with a given function does not occur in this amount of time, the time-out handler transfers control to the BIOS Time-Out Routine. The Time-Out Routine disables further interrupts for the device, aborts the function, and cleanly returns control to the operating system.

Either transfer convention will do

Depending on the transfer convention used, the operating system's time-out handler can handle hardware time-outs via either the Common Interrupt Routine or through a Service-Specific Interrupt Routine.

continued

Time-out Routine vs. Resume Stage after Time Delay value

It is important not to confuse the processing associated with the Time-out Routine with the processing associated with the Resume Stage After Time Delay value (Return Code 0002h) returned by the BIOS for those stages of a function that are driven by a function-requested time delay.

The BIOS Time-out Routine is associated exclusively with those stages of a multistage function that are driven by hardware interrupts. It is designed to handle function termination cleanly.

The Resume Stage after Time Delay (Return Code 0002h) value is associated exclusively with those stages of a multistaged function that are not driven by a hardware interrupt. Error handling resulting from this stage is handled by the BIOS function itself. Generally, the function indicates an error in a time delay-driven stage by returning a Return Code of 0002h to its Request Block.

ABIOS and Program Access

Introduction

For now, OS/2 is the primary operating system using ABIOS. OS/2 itself provides end user programs with powerful functionality which makes bypassing the operating system in favor of the ABIOS largely unnecessary. However, the need may arise for the greater hardware control provided by direct access to the ABIOS Services.

The Anchor Pointer, OS/2, and program access

When ABIOS is initialized under OS/2, the segment address of the Common Data Area is stored by the operating system in a non-public variable called the Anchor Pointer. See Chapter 3 for more information about Anchor Pointers and public and private data.

Control cannot be transferred to an ABIOS function without first accessing the information contained in the Common Data Area. Therefore, access to the ABIOS is impossible from a program running under OS/2 unless it is supported by the operating system.

IBM OS/2 allows program access to ABIOS

Although it does not make the Anchor Pointer public, IBM OS/2 (versions 1.1 and beyond), support program access of the ABIOS via two OS/2 **DevHlp** services: **ABIOSCommonEntry** and **ABIOSCall**.

Programmers who are not using IBM OS/2 should refer to their OS/2 documentation to determine if their version of OS/2 supports direct access to the ABIOS.

Accessing BIOS via BIOSCommonEntry

Description

The IBM OS/2 service **BIOSCommonEntry** is used to call an BIOS functions via the BIOS Transfer Convention, that is to say via the BIOS Common Entry Routines.

BIOSCommonEntry initializes the stack frame with pointers in the format required by the current processor mode. Then it calls the Common Entry Routine specified in DH. On return, **BIOSCommonEntry** cleans up the stack before returning to the caller.

Caller must locate Logical ID

Before invoking **BIOSCommonEntry**, the caller must first initialize the Request Block associated with the BIOS function to be called. Since a service's Logical ID is a mandatory input into each function Request Block, the caller is responsible for determining the Logical ID assigned to the service being called.

Locating Logical ID via function 01h

Because the Anchor Pointer to the Common Data Area is a non public variable, the only way for the caller to determine a service's Logical ID is to invoke function 01h Return Logical ID Parameters for each entry in the Common Data Area.

To do this, the caller must use **BIOSCommonEntry** to invoke function 01h, Return Logical ID Parameters for Logical IDs 03h to nnh. The Request Block associated with function 01h of each BIOS service is fixed at 20h bytes. When called, function 01h returns to offset 12h the hardware Device ID associated with the service. From this value, the caller can determine which device service is linked to a given Logical ID.

continued

Accessing BIOS via BIOSCommonEntry, Continued

BIOS supported devices

The BIOS supports 16 kinds of physical devices. There is one BIOS device service for each device. The table below lists the physical device ID and the BIOS device services tied to those devices.

Device ID	Device Type/Service	Device ID	Device Type/Service
00h	BIOS Internal Calls	0Bh	Pointing Device
01h	Diskette	0Ch	Reserved
02h	Fixed Disk	0Dh	Reserved
03h	Video	0Eh	CMOS RAM
04h	Keyboard	0Fh	Direct Memory Access
05h	Parallel Port	10h	Programmable Option Select
06h	Serial Port	11h	Error Log
07h	System Timer	12h-15h	Reserved
08h	Real Time Clock Timer	16h	Keyboard Security
09h	System Services	17h-FFFFh	Reserved
0Ah	Nonmaskable Interrupt		

BIOSCommonEntry Input/Output

Input:

```
MOV SI, Request_Block_Offset ; Offset in DS of Request Block
MOV DH, Which_Com_Routine    ; Indicate in DH which Common
                              ; Routine to call, where:
                              ; 00h = Common Start Routine
                              ; 01h = Common Interrupt Routine
                              ; 02h = Common Time-out Routine

MOV DL, DevHlp_BIOSCommonEntry
CALL [Device_Help]
```

Output:

```
CF      = 0 If call was successful
        = 1 If error occurred
AX      = Error Code
        BIOS not present.
        Unknown BIOS command.
```

continued

Accessing BIOS via BIOSCommonEntry, Continued

To avoid suspension in the background

BIOS functions can sometimes be suspended if the operating environment is shifted from OS/2 mode to the DOS compatibility box. This occurs when functions executed in the DOS compatibility box put the service's operating environment in a state that is unknown to the function called in OS/2 mode.

ROMCriticalSection sets a flag that prevents entry into the DOS compatibility box until the function called via **BIOSCommonEntry** has executed to completion. Since there is no way to determine in advance whether or not a function is susceptible to suspension, the caller has two choices:

- call OS/2 **ROMCriticalSection** before calling **BIOSCommonEntry**, or
- test the function by calling it via **BIOSCommonEntry** and switching to the DOS compatibility box.

If **ROMCriticalSection** is called to prevent entry into the DOS compatibility box, then it must be called again after the BIOS function completes to re-enable entry.

ROMCriticalSection Input/Output

DS must point to the BIOS Device Driver's data segment. Reset DS if it has been previously used in a **PhysToVirt** call.

```
MOV AL, enter_or_exit      ;Critical Section Flag
                           ; = 0 exit
                           ; < > 0 enter
MOV DL,DevHlp_ROMCriticalSection
CALL [Device_Help]
```

For more information

For more information on calling BIOS functions via **BIOSCommonEntry**, refer to *IBM Operating System/2 Technical Reference, Volume 1*.

Accessing BIOS via BIOSCall

Description

The IBM OS/2 service **BIOSCall** is used to call an BIOS functions via the Operating System Transfer Convention via the BIOS Service-Specific Entry Routines.

BIOSCall initializes the stack frame with pointers in the format required by the current processor mode. Then, it calls the Service-Specific Entry Routine specified in DH. On return, **BIOSCall** cleans up the stack before returning to the caller.

BIOSCall Input/Output

Input:

```
MOV AX, LID                ;Service's Logical ID
MOV SI, RB_Offset         ;Data Segment DS offset to
                           ;caller's Request Block
MOV DH, Entry_Point      ;Service-Specific Routine
                           ;00h = Start Routine
                           ;01h = Interrupt Routine
                           ;02h = Time-out Routine

MOV DL, DevHlp_BIOSCall
CALL [Device_Help]
```

Output:

```
CF      = 0  If call was successful
        = 1  If error occurred
AX      = Error Code
        BIOS not present.
        Unknown BIOS command.
```

To avoid suspension in the background

BIOS functions can sometimes be suspended if the operating environment is shifted from OS/2 mode to the DOS compatibility box. To avoid this, the caller should call OS/2 **ROMCriticalSection** before calling **BIOSCommonEntry**. The method for doing this is described on the previous page.

Chapter 7

ABIOS Extensions

Overview

Introduction

In general, how a given BIOS extension is implemented depends on the nature of the BIOS-to-operating system interface.

BIOS extensions in PC XT/AT systems

In IBM PC XT/AT-compatible environments, system hardware is isolated from the operating system (typically MS- or PC-DOS) by a ROM-based BIOS, such as the Phoenix AT-compatible BIOS.

Under DOS, ROM BIOS interrupt vectors and data structures are publicly accessible. As a result, system hardware or software enhancements that require extensions to the BIOS are typically implemented by intercepting an existing software interrupt. Since BIOS data is public, BIOS extensions of this type are free to use existing BIOS data or to create data structures of their own.

continued

CBIOS extensions in PS/2-compatible systems

In MCA-based PS/2-compatibles, the CBIOS portion of the ROM BIOS isolates single-tasking operating systems (such as MS- or PC-DOS) from system hardware in the same way the BIOS isolates DOS from the hardware in PC XT/AT-compatible systems. Extensions to the CBIOS/DOS interface are implemented exactly as they are in PC XT/AT-compatible systems.

ABIOS extensions

When a multitasking operating system (such as OS/2) is interfaced with a PS/2-compatible system, that operating system interfaces with the ABIOS portion of the ROM BIOS.

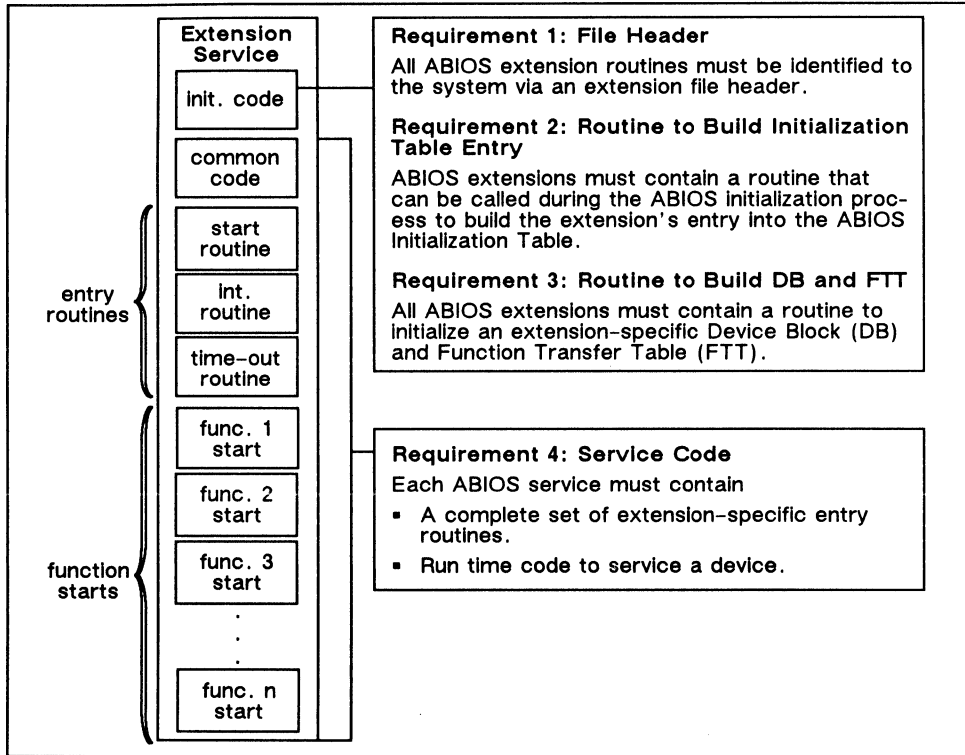
Under OS/2, however, ABIOS data structure addresses and service/function entry points are only known to the operating system. Because of this, enhancements to system software or hardware that require ABIOS extensions must insure that:

- the undefined interstage state information and work areas contained in the pre-existing service's Device Block are not overwritten,
 - the extension maintains control of all function entry points, and
 - the extension will only be initialized if its revision level is greater than the revision level of the pre-existing service.
-

continued

Summary of requirements

To ensure each of the above considerations, all BIOS extensions must meet the requirements outlined in the illustration below:

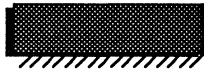
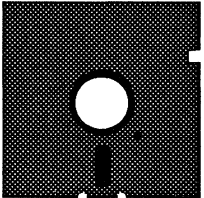


continued

Two types of BIOS implementations

As long as it meets the four requirements for creating valid BIOS extensions, an extension can be implemented either in ROM or in RAM.

The differences between the two types of implementations are described below.

Type	Description
ROM extension 	ABIOS ROM extensions: <ul style="list-style-type: none">▪ Exist in the same peripheral card ROM that contains their CBIOS counterparts.▪ Contain manufacturer-specific BIOS device services.▪ Are located and initialized as part of the overall BIOS ROM initialization process.
RAM extension 	ABIOS RAM extensions exist as files that will be located and initialized into system RAM during the overall BIOS ROM initialization process. If they are to be implemented under OS/2, extension file names must be listed in the file ABIOS.SYS. The following facts apply: <ul style="list-style-type: none">▪ ABIOS.SYS must contain a list of filespecs separated by either blanks or new lines.▪ Both ABIOS.SYS and any files listed in ABIOS.SYS must reside in the root directory of the OS/2 IPL volume.▪ The files listed in ABIOS.SYS are loaded into memory in the order in which they are listed.▪ The sector size of all RAM extension files must be a multiple of 512K.▪ All ABIOS RAM extension files must have a .BIO extension.▪ RAM extensions are loaded after ROM extensions.

continued

Overview, Continued

In this chapter

This chapter explains how to implement BIOS extensions.

Topics presented

The following topics are presented:

- Recommendations for extending BIOS
 - Requirement 1: BIOS Extension Header Formats
 - Requirement 2: Build Initialization Table Routine
 - Requirement 3: Build Device Block and Function Transfer Table Routine
 - Requirement 4: BIOS Function Code
 - Initialization: BIOS ROM Extensions
 - Initialization: BIOS RAM Extensions
 - Examples: Introduction
 - Example 1: Non-Intrusive Interception
 - Example 2: Redirection of a Nonstaged Function
 - Example 3: Redirection of a Staged Function
-

Recommendations for Extending BIOS

Background

IBM suggests in its document, *Advanced BIOS Supplement for the Personal System/2 and Personal Computer BIOS Interface Technical Reference*, that an ABIOS extension may:

- ADD a completely new device service.
 - REPLACE an existing device service.
 - PATCH revised function support into an existing device service.
 - EXTEND new function support to an existing device service.
-

Why patching and extending are not recommended

In its definition for ABIOS extensions that patch or extend device services, IBM suggests that such function-level ABIOS extensions can be implemented by changing the vectors to function starts in the Function Transfer Table used by the previous service.

There are two problems with this approach to function-level extensions:

■ **Start replacement does not accommodate interstage data**

If the start of a staged function is replaced, interrupt and time-out handling associated with this function must also be replaced (because the extension and the previous service may use interstage state information differently). There are no public vectors to an individual function's interrupt and time-out handling.

■ **Stack and register contents are not defined**

Stack and register contents passed to individual function starts are not defined by IBM and are presumably subject to change. Therefore, a start routine in one service doesn't know how to pass control to an individual function start in another service, and the function start doesn't know how to return.

continued

Recommendations for Extending BIOS, Continued

Phoenix recommendations

Phoenix Technologies Ltd. recommends that function-level extensions to the BIOS fall under the single class: MODIFY. Extensions that modify BIOS services on the function level must carefully conform to the requirements set down in this chapter. The requirements insure that:

- **Device Block private data is not overwritten**

The Device Blocks associated with all BIOS services contain a substantial amount of “private” data that is neither publicly defined nor publicly accessible.

- **Interstage state information is not lost**

Interstage state information is not defined. Therefore, if an extension performs any stage of a particular function or a particular call, it must perform all stages.

Phoenix terminology

When handled in this way, BIOS extensions fall into three categories: REPLACE, ADD, and MODIFY. As a general rule, the extent of change between the pre-existing service and the extension service should help determine which extension type is required.

Graphic examples

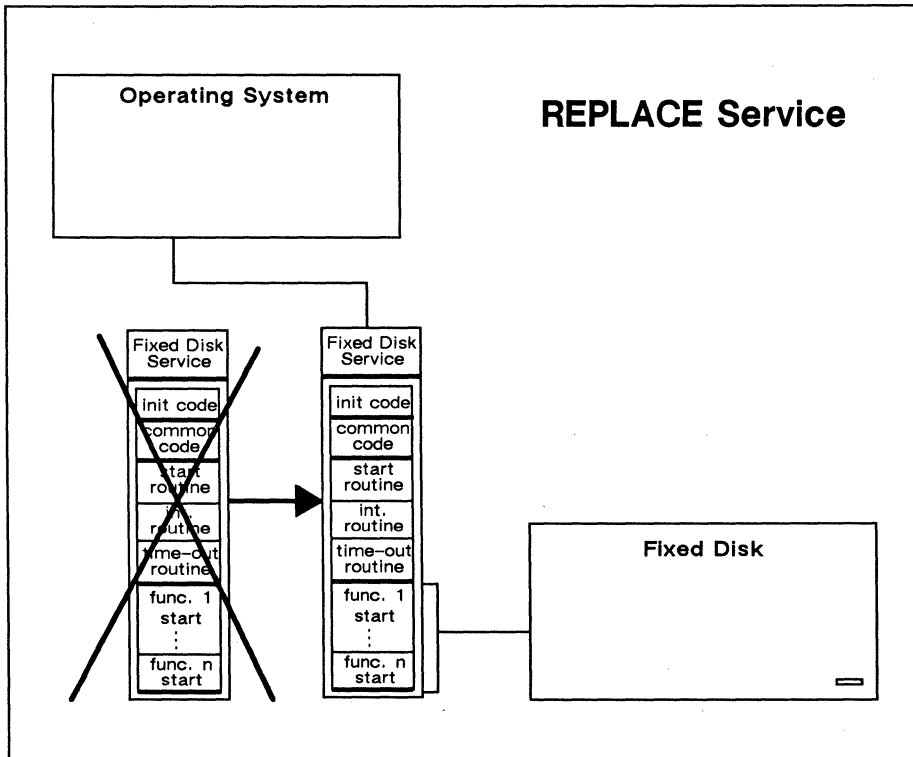
The following illustrations graphically depict how extensions that REPLACE, ADD, or MODIFY might interface with fixed disk hardware.

continued

Recommendations for Extending BIOS, Continued

ABIOS extensions that REPLACE services

ABIOS extensions that replace an existing service completely supersede that service.



The REPLACE option and ABIOS Data Structures

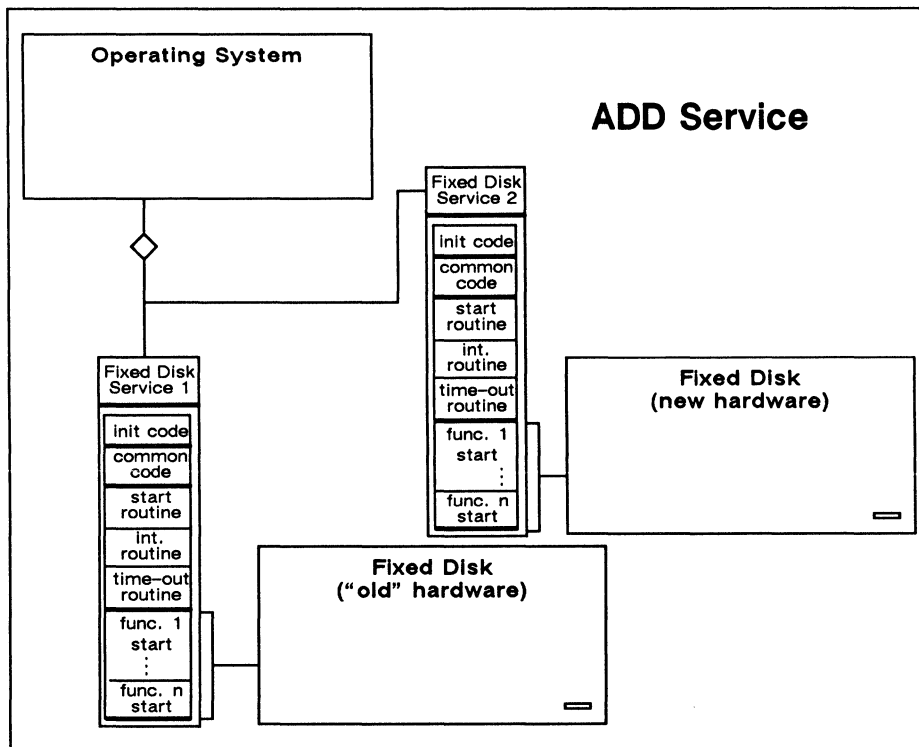
ABIOS extensions that replace existing services require an all new Device Block and Function Transfer Table. At initialization, the Common Data Area pointers to the "old" service's Device Block and Function Transfer Table are zeroed, thus nullifying all calls to the older service.

continued

Recommendations for Extending BIOS, Continued

ABIOS extensions that ADD services

ABIOS extensions that add a service are similar to those extensions that replace a service, but differ in one critical respect. ABIOS extensions that add a service do not supersede any pre-existing service.



The ADD option and ABIOS Data Structures

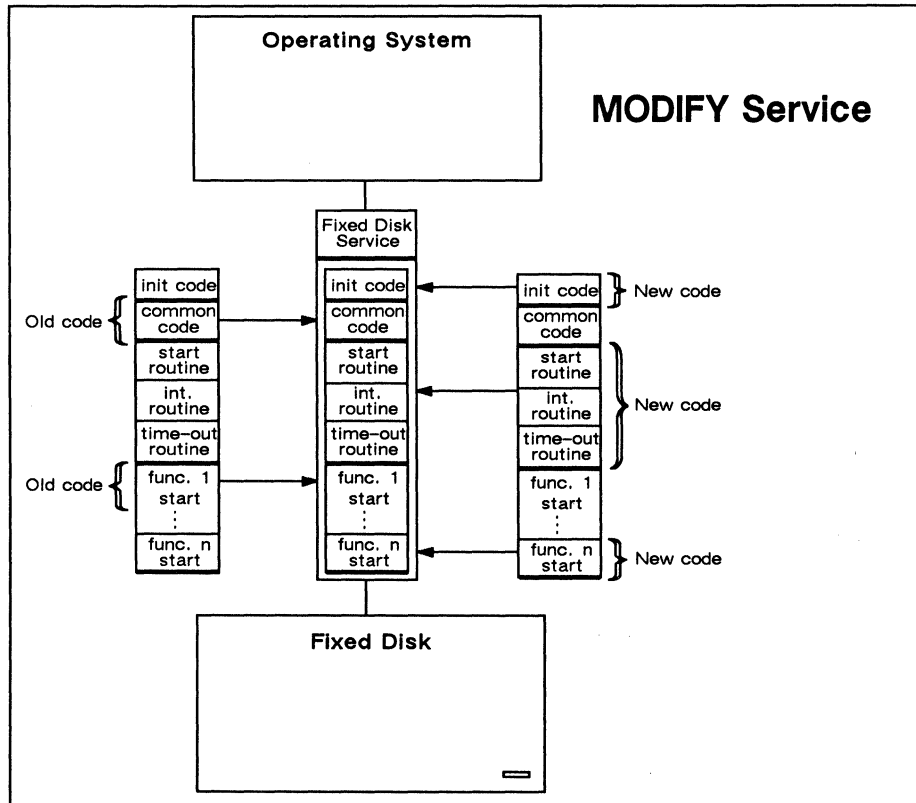
ABIOS extensions that add a new, coexistent device service require an all new Device Block and Function Transfer Table. At initialization, each device service is assigned its own Logical ID. Function calls to each service are distinguished from each other by Logical ID.

continued

Recommendations for Extending BIOS, Continued

ABIOS extensions that MODIFY services

When the degree of change between a pre-existing BIOS service and the extension is small, the BIOS service can be modified by creating an extension that shares duties with the pre-existing service.



continued

The MODIFY option and BIOS Data Structures

BIOS extensions that share functionality with an existing service require a “hybrid” Device Block and a new Function Transfer Table. At initialization, the Common Data Area pointers to the “old” device service data structures are nullified. All function calls are relayed via a new Logical ID associated with the modified service.

The new Function Transfer Table directs all functions to the new service, where a decision to revector certain functions to the “old” service may be made. The private Device Block parts reserved by the “old” service must not be written to by the new service; private data may not be shared between services.

Requirement 1: Create Proper Extension Header

Introduction

ABIOS extensions are identified to the system by an extension file header. The format of this header varies depending on whether the extension is to be implemented in ROM or RAM.

ROM extension file header

The ROM extension file header is described in the table below.

Offset	Bytes	Description
00h	2	EXTENSION ROM SIGNATURE = AA55h Extension ROMs may contain extensions to either (<i>or both</i>) the CBIOS and the ABIOS. The power-on self test (POST) searches for CBIOS extension ROMs in 2K increments over the absolute address range C0000h–DF800h. ABIOS extension ROMs are searched for over the same address range as part of ABIOS initialization. The extension ROM signature AA55h informs POST that an extension ROM is present. All extension ROMs must begin on a paragraph boundary.
02h	1	ROM LENGTH IN 512-BYTE BLOCKS Extension ROMs must be divided into 512-byte blocks. The maximum value for this field is 7Fh.
03h	3	ENTRY POINT TO CBIOS INITIALIZATION ROUTINE (FAR CALL) If the extension ROM contains a portion that will be used to extend CBIOS, then the entry point to the CBIOS extension's Initialization Routine must appear here. Note: If the extension ROM contains an ABIOS extension only, this field must contain a dummy Return Far instruction. The dummy Return Far instruction allows POST to continue its ROM extension scan.
06h	2	ABIOS EXTENSION ROM SIGNATURE — BB66h The search for ABIOS extension ROMs is performed in 2K increments over the absolute address range C0000h–DFFFFh as part of ABIOS initialization. The ABIOS Extension ROM signature indicates that the ROM contains an ABIOS extension. The ABIOS Extension Signature must be set to BB66h.
08h	1	NUMBER OF ENTRIES TO ADD TO INITIALIZATION TABLE The total number of Initialization Table Entries is accumulated as part of the ABIOS initialization process. This field must contain the number of Initialization Table Entries required by the ABIOS ROM extension. The minimum acceptable value is 1.
09h	–	ENTRY POINT FOR BUILD INITIALIZATION TABLE ROUTINE Entry point to routine to Build Initialization Table routine.

continued

Requirement 1: Create Proper Extension Header, Continued

RAM extension file header

The RAM extension file header is described in the table below.

Offset	Bytes	Description
00h	2	EXTENSION RAM PRESENT = AA55h The value AA55h indicates that a RAM extension is present.
02h	1	RAM EXTENSION LENGTH IN 512-BYTE BLOCKS Extension RAMs must be created in increments of 512-byte blocks. The maximum value for this field is 7Fh.
03h 04h 05h	1 1 1	MODEL BYTE SUBMODEL BYTE ROM REVISION LEVEL The model byte and submodel byte identify the system board with which the RAM extension is to be associated. The ROM Revision Level indicates the revision of system board ROM which the RAM extension is designed to interact.
06h	2	DEVICE ID The Device ID field (<i>along with the secondary device ID below</i>) identifies the BIOS device with which the RAM extension is to be associated.
08h	2	NUMBER OF ENTRIES TO ADD TO INITIALIZATION TABLE The total number of Initialization Table Entries is accumulated as part of the BIOS initialization process. The value contained in this field must be set to the number of Initialization Table Entries that will be required by this BIOS ROM extension. The minimum acceptable value is 1.
09h	1	ENTRY POINT FOR BUILD INITIALIZATION TABLE ROUTINE Entry point to routine to Build Initialization Table routine.
0Ch	1	SECONDARY DEVICE ID When more than one physical device is associated with a Device ID, the two physical devices can be differentiated with a Secondary Device ID. When a unique BIOS Service is required for each physical device, the two services must be differentiated via their Secondary Device ID.
0Dh	1	RAM EXTENSION REVISION LEVEL The RAM extension revision level identifies the revision level of the RAM extension.
0Eh	2	RESERVED (Must be initialized to zero.)

continued

Requirement 1: Create Proper Extension Header, Continued

Table of BIOS Device IDs and Services

The system ROM BIOS contains one device service for each of the Device IDs listed in the table below.

Device ID	Device Type/Service	Device ID	Device Type/Service
00h	BIOS Internal Calls	08h	Pointing Device
01h	Diskette	0Ch	Reserved
02h	Fixed Disk	0Dh	Reserved
03h	Video	0Eh	CMOS RAM
04h	Keyboard	0Fh	Direct Memory Access
05h	Parallel Port	10h	Programmable Option Select
06h	Serial Port	11h	Error Log
07h	System Timer	12h-15h	Reserved
08h	Real Time Clock Timer	16h	Keyboard Security
09h	System Services	17h-FFFFh	Reserved
0Ah	Nonmaskable Interrupt		

Requirement 2: Build Initialization Table Entry Routine

Introduction

ABIOS extensions are initialized and integrated as part of the general process of initializing the system ROM BIOS. BIOS extensions must contain a routine that can be called during the BIOS initialization process to build the extension's entry into the BIOS Initialization Table.

Example: Routine to Build Initialization Table Entry

The example below shows how a routine to build an BIOS extension's entry into the Initialization Table could be implemented.

Those routines associated with BIOS extensions that modify an existing service must reserve a Device Block size equal to that reserved by the existing service plus enough bytes for the modification's workspace.

```
MY-BINIT                proc far
;                        Entry: ES:DI points to where to put init
                        table entry(s)
;
;                        Exit: far ret w/AL =000h (successful),
                        80h (no units found), or other error and
                        CX = # entries added. Other registers are
                        preserved.

                        assume DS:nothing, ES:nothing
                        cld
; *                      patch CS in template
                        mov  fixseg1, CS
; *                      copy template to entry
                        mov  SI, offset MY-INITEMPL
                        mov  CX, 18h/2
                        rep movs word ptr ES:[DI], word ptr CS:[SI]
                        sub  di, 18h
```

continued

Requirement 2: Build Initialization Table Entry Routine, Continued

Example: Routine to Build Initialization Table Entry, cont'd

```
;* At this point, if this is a MODIFY, the Device Block
;* and Request Block reserved sizes should be calculated
;* from those of the previous service plus the sizes for
;* workspace required by the modification.
;*
;*
;*      return w/AL = 0, cx = 1
;*      popa
;*      mov  al, 0
;*      mov  cx, 1
;*      ret
MY_BINIT  endp

MY_INITEMPL label word          ; Init Table Template
      dw  MY_DID1                ; device ID
      dw  MY_LIDCNT              ; logical ID count
      dw  MY_DEVBLEL            ; device block size in
                                ; bytes
      dw  offset MY_INITDS      ; ptr to init-FUNCT-and-
                                ; DEVBL routine
fixseg1 dw  ?                    ; CS - fill in the blank
      dw  MY_REQBLLEN           ; default request block
                                ; length
      dw  (MY_FUNCNT +4)*4      ; function table size in
                                ; bytes
      dw  MY_DATAPCNT*6         ; bytes required for
                                ; buffer ptrs
      dw  MY_DID2                ; secondary Device ID
      dw  MY_REV+80H            ; code level
      dw  0, 0, 0                ; reserved
```

Requirement 3: Routine to Build Device Blocks and FTT

Introduction

All BIOS extensions must contain a routine to initialize an extension-specific Device Block and Function Transfer Table (FTT).

How the routine is entered

The routine to build an extension-specific Device Block and Function Transfer Table must be entered with DS:00h pointing to the Common Data Area (CDA) and DX = Logical ID, so that:

- the double word at DS:(DX*8) points to where the Device Block is to be built, and
 - the double word at DS:(DX*8+4) points to where the Function Transfer Table is to be built.
-

continued

Requirement 3: Routine to Build Device Blocks and FTT, Continued

Extension type requirements

How the routine to initialize an extension's Device Block and Function Transfer operates depends on the type of BIOS extension involved.

Extension Type	Initialization Routine
ADD Service	The extension's Device Block and Function Transfer Table can be initialized to any length required.
REPLACE Service	<p>The initialization routine must zero the Common Data Area pointers to the older service's Device Block and Function Transfer Table.</p> <p>This ensures that the operating system will not call the BIOS service being replaced.</p>
MODIFY Service	<p>The initialization routine must:</p> <ul style="list-style-type: none">▪ Copy the existing service's Device Block to where the extension's Device Block is to be built.▪ Initialize the modification's workspace in the Device Block.▪ Patch the existing Device ID in the service's Device Block to a number unrecognized by an operating system (e.g. 0FFh) so that an operating system will not call it. <p>Initializing the Device Block in this way ensures that the functions involved with the BIOS extension:</p> <ul style="list-style-type: none">▪ Do not make use of any undefined areas within the existing service's Device Block.▪ Access extension-specific data only in the extension-specific workspace added at the end of the Device Block. In this way interstage state information need not be maintained between the extension and the pre-existing BIOS service.▪ Build an extension-specific Function Transfer Table that takes over all three entry routines.

continued

Requirement 3: Routine to Build Device Block and FTT, Continued

Pseudocode: Routine to Build Device Block and Function Transfer Table for a Fixed Disk Service REPLACEMENT

```
MY_INITDS proc far
;
;           Entry:   CX = logical ID count
;
;           DX = 1st logical ID
;
;           DS:0 points to CDA
;
;           Exit:    FAR RET w/: AL = error code (0)
;*
;*           for each LID up until mine,
;*           if it's for my device (DIDs & DID2s are =),
;*           if device block rev > mine,
;*           go zero my LID's ptrs and abort
;*           endif
;*           zero that LID's ptrs
;*           endif
;*           next LID
;*           init device block(s) {
;*           copy DB template
;*           init device unique data
;*           init all unit unique data {
;*           for each UUDA,
;*           get drive #
;*           get potential hard disk params
;*           get drive type, params for type
;*           call UUDInit
;*           next UUDA
;*           }
;*           fill in offset to first data pointer
;*           fill in LID
;*           calc & fill in dev block size
;*           }
;*           build function transfer table
;*           exit
MY_INITDS endp
```

Requirement 4: BIOS Service Code

Introduction

Each BIOS extension must contain the code implementing all or part of an BIOS device service. This is the run-time code of an extension; the other requirements deal only with BIOS initialization.

Reference

Information under the Examples heading in this chapter provides concrete examples of how various kinds of extensions can be coded.

Initialization: BIOS ROM Extensions

Introduction

The process of initializing an BIOS ROM extension includes the same steps and is closely integrated with the process of initializing the system board ROM BIOS.

Limits of this discussion

The discussion below is limited to only those areas of BIOS initialization directly related to the initialization of BIOS ROM extensions. For additional BIOS initialization information, refer to Chapter 4.

continued

Initialization: BIOS ROM Extensions, Continued

ROM extension initialization procedure

The table below lists the steps involved in initializing BIOS ROM extensions.

Step	Action
1.	The operating system calls CBIOS INT 15h, AH = 04h Build System Parameters Table.
2.	The Build System Parameters Table function scans (in 2K increments) absolute addresses C0000h to DFFFFh, testing for a valid BIOS ROM extension header. Valid BIOS ROM headers contain a signature value of BB66h at offset 06h.
3.	When a valid BIOS ROM extension is found, the Build System Parameters Table function: <ul style="list-style-type: none">▪ Reads the Number of Initialization Table Entries at ROM extension header offset 08h.▪ Adds that value to the value in offset 1Eh of the System Parameters Table.▪ Continues its search at the next 2K boundary.
4.	The operating system call CBIOS INT 15h, AH = 05h Build Initialization Table.
5.	The Build Initialization Table function scans (in 2K increments) absolute addresses C0000h to DFFFFh, testing for a valid BIOS ROM extension header.
6.	Valid BIOS ROM headers contain a signature value of BB66h at offset 06h. When a valid BIOS ROM extension is found, the Build Initialization Table function performs a far call to offset 09h in the ROM extension header. This is the ROM extension's routine to Build Initialization Table Entry.
7.	Once it has built its entry into the Initialization Table, the BIOS ROM extension's Routine to Build the extension's Device Block and Function Transfer Table are called directly from OS/2 in the same manner as system board BIOS device code. The routine to build the extension's Device Block and Function Transfer Table may choose to install or not install itself. If it does not install itself, it zeros its Device Block pointer and Function Transfer Table pointer in the Common Data Area.

Initialization: BIOS RAM Extensions

Introduction

The process of initializing an BIOS RAM extension is similar to the process of initializing an BIOS ROM extension.

The difference between the two lies in the additional steps that the Build Initialization Table routine contained BIOS RAM extensions must take to insure that only the latest revision of a given BIOS service is initialized.

When RAM extensions are initialized

BIOS RAM extensions are initialized only after the system board ROM BIOS and all adapter board BIOS ROM extensions have been fully initialized.

Extension requirements reprised

As discussed in the previous pages, all BIOS RAM extensions must meet the four requirements below:

- Requirement 1: ROM Extension Header
 - Requirement 2: Build Initialization Table Routine
 - Requirement 3: Routine to Build Device Block and Function Transfer Table
 - Requirement 4: Extension Code
-

Limits of this discussion

The discussion below is limited to those areas of BIOS initialization directly related to the initialization of BIOS RAM extensions. For further information about BIOS initialization, refer to Chapter 4.

continued

Initialization: BIOS RAM Extensions, Continued

RAM extension initialization procedure

The table below lists the steps involved in initializing BIOS RAM extensions.

Step	Action
1.	The operating system calls CBIOS INT 15h, AH = 04h Build System Parameters Table.
2.	The Build System Parameters Table function scans RAM extension area, polling the Number of Initialization Table Entries field (offset 08h) of each RAM extension header to determine the total number of Initialization Table Entries.
3.	When a valid BIOS RAM extension is found, the Build System Parameters Table function: <ul style="list-style-type: none">▪ Reads the Number of Initialization Table Entries at RAM extension header offset 08h.▪ Adds that value to the value in offset 1Eh of the System Parameters Table.▪ Continues its search at the next 2K boundary.
4.	The operating system calls CBIOS INT 15h, AH = 05h Build Initialization Table.
5.	Once it has built its entry into the Initialization Table, the BIOS ROM extension's Routine to Build the extension's Device Block and Function Transfer Table are called directly from OS/2 in the same manner as system board BIOS device code.
6.	If the extension replaces or modifies an existing service, the Routine to Build DB and FTT may compare the extension's revision level against the revision level of the existing service, choosing to replace or modify only if its own revision is greater. To disable either the existing service or the extension's service, the corresponding DB and FTT pointers in the CDA must be zeroed.

Reference

To be initialized, all BIOS RAM extensions should be listed in the OS/2 file ABIOS.SYS. For more information on ABIOS.SYS, refer to the documentation accompanying OS/2.

continued

Initialization: BIOS RAM Extensions, Continued

The BIOS RAM extension area

The BIOS RAM extension area resides in system RAM and contains a chained list of all BIOS RAM extensions. The BIOS RAM extension area must start on a paragraph boundary. Individual RAM extensions within the BIOS RAM extension area are linked via the Extension Length in 512-Byte Blocks field in the RAM Extension header.

Considerations for last RAM extension in area

The last RAM extension in the BIOS RAM extension area must have its Extension Length in 512-Byte Blocks header field set to zero.

Examples of How to Modify an Existing Service

Introduction

Anything a programmer might hope to achieve with BIOS extensions that MODIFY an existing service can be achieved by building extensions that do not replace individual function-start vectors.

Well-behaved programming strategies that are useful in different situations include:

- Non-intrusive service interception
 - Redirection of a nonstaged function
 - Redirection of a staged function
-

Example 1: Non-Intrusive Interception

Description

The example below shows how an BIOS extension can replace an entire service yet redirect all activity to the previous device service transparently to that service and to the operating system.

Code example: non-intrusive interception

```
CODE segment byte public
    assume cs:CODE
start:
;   RAM extension header
    dw 0AA55h
    db EXTSIZE                ; length in half Ks
    db 0, 0, 0                ; model, submodel, ROM revision
    dw 2                      ; device ID for hard disk
    db 1                      ; # initialization table entries
    jmp near ptr BINIT        ; build init table entry
    db OFFh, OFFh            ; secondary dev ID, my revision
    dw 0                      ; reserved
;   my workspace in pseudo-Device-Block:
;   offsets into workspace:
WORK1          equ 0          ; word
WORK3          equ 2          ; byte
WORKSPACE_SIZE equ 3          ; my read-only (after init) data:
OLDFUNCTOFF   dw ?           ; offset in CDA of ptr to old LIDs
FUNCT
MYDATAPTROFF  dw ?           ; offset in CDA of my data ptr
WORKSPACEOFFSET dw ?         ; offset in device block of my
workspace
FUNCTABLE_SIZE dw ?          ; size of previous function table
```

continued

Example 1: Non-Intrusive Interception, Continued

Code example: non-intrusive interception, cont'd

```
BINIT proc far ; Build-initialization table subroutine
;* find previous init table for this device ID
;* if none, go return unhappy
;* store DID, LID count=1
;* WORKSPACEOFFSET = previous device block size
;* store device block size = previous device block size +
;* WORKSPACESIZE
;* store pointer to init-data-structures subroutine
;* store request block size = previous request block size
;* store function table size = 4*4
;* store data pointer size, DID2, revision, 0000s
;* return happy
BINIT      endp

INITDS     proc far ; Initialize-data-structures subroutine
;* get offset of our pointers in CDA
;* find previous service's ptrs in CDA
;* save offset to its function table
;* copy device block to mine, changing size
;* init workspace at end of device block
;* give old driver a weird device ID so OS doesn't call it
;* build function transfer table - vectors to
;* LOGSTART, LOGINTERRUPT,
;* and LOG Time-out routines (and no functions)
;* update Common Data Area's data pointer count
;* find where my data pointer goes
;* load my data pointer
;* MYDATAPTROFF = offset of my data pointer's selector in CDA
;* return happy
INITDS     endp
```

continued

Example 1: Non-Intrusive Interception, Continued

Code example: non-intrusive interception, cont'd

```
LOGTime-out:                ; time-out routine
;* save offset into transfer table = 8
;* goto LOGCOMMON
LOGINTERRUPT:              ; interrupt routine
;* save offset into transfer table = 4
;* goto LOGCOMMON
LOGSTART:                  ; start routine
;* save offset into transfer table = 0
;* goto LOGCOMMON

LOGCOMMON proc far
;* save registers
;* change function transfer table in stack for revector
;* get address of previous service's start, int, or time-out
;* routine from previous transfer table
;* place it in stack for return from here
;* get function code from request block
;* get pointer to workspace
;* get my data pointer
;* record function code
;* restore registers and return (to previous service's start
;* routine)
LOGCOMMON endp

EXTSIZE = (offset $ - offset start + 1FFh)/200h
;* RAM ext. size / 200h
org EXTSIZE*200h

CODE ends
end
```

Example 2: Redirection of a Nonstaged Function

Description

Redirection of a nonstaged function is relatively easy. The extension's start routine examines the function code in the request block and either services the function itself or redirects the request to the previous device service.

In the example below, a replacement video service redirects all functions to the previous video service and regains control when that service returns. It then checks to see if one of the font-loading functions was performed; if so, the font is "italicized" by shifting half of each loaded character font.

Code example: redirecting a nonstaged function — italicizing fonts

```
INITDS    proc far ; Initialize-data-structures subroutine
;* get offset of our pointers in CDA
;* find previous service's ptrs in CDA
;* save offset to its function table
;* copy device block to mine
;* give old driver a weird device ID so OS doesn't call it
;* build function transfer table - vector to MYSTART (and no
;* functions)
;* update Common Data Area's data pointer count
;* find where my data pointer goes
;* load my data pointer
;* MYDATAPTROFF = offset of my data pointer's selector in CDA
;* return happy
INITDS    endp
```

continued

Example 2: Redirection of a Nonstaged Function, Continued

Code example: redirecting a nonstaged function — italicizing fonts, cont'd

```
MYSTART      proc far ; start routine
;* Save registers.
;* Build additional stack, copying pointers from passed stack,
;* replacing transfer table vector with vector to previous
;* transfer table.
;* Get previous service start address from previous transfer
;* table.
;* Call previous service start routine.
;* Flush additional stack.
;* Get function code from request block.
;*   if function code = 0Fh or 10h,
;*       set up hardware to access font plane,
;*       get my data pointer (to font at A0000)
;*       for each character,
;*           shift top half of character definition right.
;*       next character
;*       restore hardware
;*   endif
;* restore registers and return to OS
MYSTART      endp
```

Example 3: Redirection of a Staged Function

Description

Replacing a staged function requires careful design because **all** stages of the interrupt function must be redirected, not just the function start.

The way to handle this is for all three device service entries — the start routine, the interrupt routine, and the time-out routine — to make redirection decisions based on function code.

For instance, if an extension wishes to replace only “read” functionality, all three entries into the device service (start, interrupt, and time-out routines) will examine the function code in the request block and redirect all functions except “read” to the corresponding entry in the previous device service.

In this manner, interstage state information (which is private) need not be compatible between, say, an extension’s start processing and the system ROM BIOS’s interrupt processing.

In the example on the following page, an BIOS extension for a diskette drive which has a “ready” signal rather than a change line is presented.

continued

Example 3: Redirection of a Staged Function, Continued

Code example: redirecting staged functions — diskette change line emulation

```
INITDS    proc far ; Initialize-data-structures subroutine
;* get offset of our pointers in CDA
;* find previous service's ptrs in CDA
;* save offset to its function table
;* copy device block to mine, changing size
;* init workspace at end of device block
;* give old driver a weird device ID so OS doesn't call it
;* build function transfer table - vectors to MYSTART,
;* MYINTERRUPT,
;* and previous service's time-out routine (and no functions)
;* update Common Data Area's data pointer count
;* find where my data pointer goes
;* load my data pointer
;* MYDATAPTROFF = offset of my data pointer's selector in CDA
;* return happy
INITDS    endp

MYINTERRUPT    proc far ; interrupt routine
;* save registers
;* if it's diskette interrupt but function code = 0000
;* (default),
;* sense interrupt status
;*     if interrupt is due to change in state of ready line,
;*         changedflag = yes
;*     endif
;*     restore regs and return to caller w/ return code 0000
;*     endif
;* get address of previous service's interrupt routine
;* place it in stack for return from here
;* restore registers and return (to previous service's
;* interrupt routine)
MYINTERRUPT    endp
```

continued

Example 3: Redirection of a Staged Function, Continued

Code example: redirecting staged functions — diskette change line emulation, cont'd

```
MYSTART      proc far ; start routine
;* save registers
;* get function code from request block
;* if function code = 000E,
;*   if changedflag = yes,
;*     restore regs and return to caller w/ change active
;*   else if changedflag = no,
;*     restore regs and return to caller w/ change inactive
;*   endif
;* else if function code = 0008 through 000B,
;*   if changedflag = yes,
;*     sense drive status
;*     if drive not ready,
;*       restore regs and return to caller
;*       (w/ return code 800D)
;*     else,
;*       restore regs and return to caller
;*       (w/ return code 8006)
;*       changedflag = no
;*     endif
;*   endif
;* endif
;* change function transfer table in stack for revector
;* get address of previous service's start routine
;* place it in stack for return from here
;* restore registers and return
;* (to previous service's start routine)
MYSTART      endp
```

Chapter 8

ABIOS Diskette Service

Overview

Introduction

The ABIOS Diskette Service provides access to diskette I/O operations.

Rules for using the PS/2 BIOS Diskette Services

Because the CBIOS and ABIOS interface with different operating systems, either the CBIOS or ABIOS Diskette Service may place diskette hardware in an unknown state. The CBIOS does not inform the ABIOS of the diskette hardware state when control is passed from the CBIOS to the ABIOS, and vice versa. As a result, it is possible for one part of the PS/2 ROM BIOS to put the diskette hardware in a state that will not be recognized by the other part of the BIOS.

continued

Overview, Continued

Rules for using the PS/2 BIOS Diskette Services, cont'd

Follow the rules listed below to avoid diskette problems in the PS/2 BIOS:

1. If there is an outstanding ABIOS Diskette function request, do not request a CBIOS diskette function.
 2. If there is an outstanding CBIOS Diskette function request, do not request an ABIOS Diskette function request.
 3. If using CBIOS and the last Diskette Service function call was to ABIOS, request INT 13h, AH = 00h, Reset Diskette System, before invoking other CBIOS Diskette functions.
 4. If using ABIOS and the last Diskette Service function call was to CBIOS, request ABIOS Diskette Service function 05h, Reset Diskette System, before invoking other ABIOS Diskette Service functions.
 5. After ABIOS is initialized, the first ABIOS Diskette Service function request must be function 05h, Reset Diskette System.
-

Logical IDs for diskette drives

All diskette drives may share the same Logical ID with each drive assigned a separate unit number. For example, all diskette drives may be Logical ID 11. Diskette drive 0 may be Logical ID 11, Unit 0 and diskette drive 1 may be Logical ID 11, Unit 1.

continued

Diskette Change Line signal

Some diskette drives are equipped with a Diskette Change Line signal that is set when the diskette drive door is opened. The Diskette Change Line signal has two states, set and not set:

- **set**

The diskette has been changed or the diskette drive door is open. Diskette data cannot be read or written in this state. Diskette media status cannot be determined while the Change Line is set.

- **not set**

The diskette drive door is closed, information can be read from and written to the diskette, and all status information can be read and modified.

Testing the Diskette Change Line signal status

The Diskette Change Line signal status is tested by the BIOS Diskette Service before functions 08h Read Diskette, 09h Write to Diskette, 0Ah Format Diskette, or 0Bh Verify Diskette Data are processed.

Function 03h Read Device Parameters may be requested to determine if the Diskette Change Line signal status is supported by the specified diskette drive.

continued

Summary of Diskette Service functions

Function	Description
00h	Default Interrupt Handler
01h	Return Logical ID Parameters
02h	Reserved
03h	Read Device Parameters
04h	Set Device Parameters
05h	Reset/Initialize Diskette Subsystem
06h	Reserved
07h	Disable Diskette
08h	Read Diskette
09h	Write to Diskette
0Ah	Format Diskette
0Bh	Verify Diskette Sectors
0Ch	Read Media Parameters
0Dh	Set Media Type for Format
0Eh	Read Change Line Signal Status
0Fh	Turn Diskette Motor Off
10h	Interrupt Status

In this chapter

This chapter includes information about the following topics:

- Hardware Environment
 - Error Handling
 - BIOS Diskette Service functions
-

Hardware Environment

Hardware interrupt

The BIOS Diskette Service is associated with hardware interrupt request 6 (IRQ 6).

Diskette controller

The BIOS supports an NEC 765 or equivalent diskette controller. Also, additional hardware support for data separation, interrupt, and DMA control is required.

Data transfer rates

All supported formats must use a 512-byte sector size. Data transfer rates supported are:

Transfer Rate (Kbs)*	Diskette Capacity	Drive Capacity	Drive Size
250Kbs	360K	360K	5.25
250Kbs	720K	720K	3.5
500Kbs	1.2MB	1.2MB	5.25
300Kbs	360K	1.2MB	5.25
250Kbs	720K	1.44MB	3.5
500Kbs	1.44MB	1.44MB	3.5
* Kilobits per second			

continued

Hardware Environment, Continued

Supported drive types

The BIOS supports a maximum of two diskette drives. The Phoenix BIOS supports both 5.25 inch and 3.5 inch diskette drives. The IBM BIOS supports only 3.5 inch diskette drives.

The Phoenix BIOS Diskette Service supports four types of diskette drives:

Maximum Storage Capacity	Disk Size (inches)	Maximum Tracks/Side	Maximum Sectors/Track
1.44 MB	3.5	80	18
720K	3.5	40	18
360K	5.25	40	9
1.2 MB	5.25	80	15

5.25 inch diskette compatibility

5.25 inch diskette media can be high density (1.2 MB) or double density (360K). Diskettes written to on one type of 5.25 inch drive may or may not be written to or read from using the other type. The following table outlines the possible read/write combinations of 5.25 inch diskette media and drive types.

Media Type	If diskette was formatted on...	Then it can be read on...	or...	And it can be written to by...
360K	360K drive	360K drives	1.2 MB drives	360K drives only
1.2 MB	360K drive	360K drives	1.2 MB drives	360K drives only

If a high density (1.2 MB) diskette is formatted on a 1.2 MB drive, it can only be read from and written to by 1.2 MB drives.

The DOS command, `format/4`, can be used to format a 360K diskette in a 1.2 MB drive. Both 1.2 MB and 360K drives can generally read and write these diskettes.

continued

Hardware Environment, Continued

3.5 inch diskette compatibility

The following outlines the possible read/write combinations for 3.5 inch drives and media:

- If a 720K media type diskette is formatted on a 720K drive, it can be read on either 720K or 1.44 MB drives, but it can only be written to by 720K drives.
 - If a 1.44 MB media type diskette is formatted on a 1.44 MB drive, it can only be read from and written to by 1.44 MB drives.
-

Error Handling

How errors are reported

ABIOS signals the status (Successful, Resume Stage after Interrupt, etc.) of each ABIOS request by returning a one word Return Code at offset 0Ch in the Request Block.

If Bit 15 of the Return Code field is set, the diskette function requested has an error. The caller's Return Code handler routine should test Bits 14, 13, 12, and 8 to determine the class of error that has occurred, and should then test the remaining bits to determine the precise nature of the error.

Retryable diskette errors

When Bit 8 (Retryable Error) of the Return Code is set, the request should be retried. The recommended number of retries is returned by Diskette function 03h, Read Device Parameters (in offset 2Bh in the Request Block).

Function: 00h — Default Interrupt Handler

Description

Function 00h, Default Interrupt Handler, is a single-staged request. It handles unexpected hardware interrupts by resetting the interrupt at the device level.

How function is requested

When the Logical ID associated with the BIOS Diskette Service has no outstanding Request Blocks waiting for an interrupt and an unexpected hardware interrupt occurs, the operating system must build a Request Block in the format below and call function 00h to turn off the interrupt at the diskette controller level.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length (10h)	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0000h)	
08h	Word	Reserved (initialize to 0000h)	
0Ah	Word	Reserved (initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0005h	Not My Interrupt, Resume Stage after Interrupt
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 01h — Return Logical ID Parameters

Description

This function is a single-staged request that returns the parameters for the Diskette Logical ID.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length (20h)		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0001h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word			
0Ch	Word			Return Code
0Eh	Word			
10h	Byte			Hardware Interrupt Level (06h)
11h	Byte			Arbitration Level (02h)
12h	Word			Device ID (0001h)
14h	Word			Count of Units
16h	Word			Logical ID Flags Bits 15-4 = Reserved Bit 3 = 0 No overlap across units Bit 2 = 0 Reserved Bits 1-0 = Transfer Data Pointer Mode: 10b = Physical Pointer Required
18h	Word			Request Block Length (for other functions)
1Ah	Byte			Reserved (Initialize to 0000h)
1Bh	Byte			Revision Level
1Ch	Word	Reserved (Initialize to 0000h)		
1Eh	Word	Reserved (Initialize to 0000h)		
3Ah	Byte		Diskette Controller Status Byte	

continued

Function: 01h — Return Logical ID Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 03h — Read Device Parameters

Description

This function is a single-staged request. It returns the parameters for this device in the Request Block as specified below. The information returned applies to the maximum capacity of the media for the specified drive type. The BIOS resets the diskette subsystem if there is a hardware error.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0003h)	
08h	Word	Reserved (initialize to 0000h)	
0Ah	Word	Reserved (initialize to 0000h)	
0Ch	Word		
0Eh	Word		Time-out Value
10h	Word		Sectors per track for the maximum media density supported by the drive
12h	Word		Sector Size in bytes 00h = Reserved 01h = 256 bytes/sector 02h = 512 bytes/sector (Default)
14h	Word		Device Control Flags Bits 15-4 = Reserved Bit 3 = Recalibrate status 0 Recalibrate not required 1 Recalibrate required Bit 2 = Concurrent operations 0 Not supported 1 Supported Bit 1 = Format unit information 0 Not supported 1 Supported Bit 0 = Change signal availability 0 Not available 1 Available

continued

Function: 03h — Read Device Parameters, Continued

Request Block Structure, cont'd

Offset	Size	Input:	Output:	
16h	Word		Diskette Drive Type 00h = Drive not present/invalid CMOS RAM 01h = 5.25 inch 40-Track, 2-Head, 360K 02h = Reserved 03h = Reserved 04h = 3.5 inch, 80-Track, 2-Head, 1.44MB	
18h	Word		Reserved (Initialize to 0000h)	Reserved
1Ch	DWord			Delay Before Turning Motor Off (microseconds)
20h	DWord			Motor Startup Time (microseconds)
26h	Word			Number of Cylinders on the Maximum-Density Media the Drive Supports
2Ah	Byte			Number of Heads
2Bh	Byte			Recommended Number of Software Retries
2Ch	Byte			Fill Byte for Format
2Dh	DWord			Head Settle Time (microseconds)
31h	Byte			Gap Length for Read/Write/Verify
32h	Byte			Gap Length for Format
33h	Byte			Data Length
3Ah	Byte			Disk Controller Status Byte

continued

Function: 03h — Read Device Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8000h	Device Busy, Request Refused
8001h	Diskette Not Started
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 04h — Set Device Parameters

Description

This function is a single-staged request. It is used to change the default Sector Size (512 bytes), Gap Length, and Data Length for diskette I/O functions.

If a diskette is formatted with a sector size other than 512 bytes, this function must be issued before executing functions 08h Read Diskette, 09h Write to Diskette, 0Ah Format Diskette, or 0Bh Verify Diskette Data.

The BIOS resets the diskette subsystem if there is a hardware error.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0004h)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		
10h	Word	Reserved (Initialize to 0000h)	
12h	Word	Size of sector (bytes) 00h = Reserved 01h = 256 bytes per sector 02h = 512 bytes per sector (default)	
31h	Byte	Gap Length for Read/Write/Verify	
33h	Byte	Data Length	

continued

Function: 04h — Set Device Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8000h	Device Busy, Request Refused
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Sector Size
FFFFh	Return Code Field Not Valid

Function: 05h — Reset/Initialize Diskette Subsystem

Description

This function is a discrete multistaged request. It resets the diskette controller to a known state, which generates an interrupt to which the caller must respond and forces a recalibration of the diskette drive(s) on the next access to each drive. The BIOS resets the diskette subsystem if there is a hardware error.

If using both BIOS and CBIOS Diskette Services

If a CBIOS Diskette Service function has been requested last, the caller must request BIOS Diskette function 05h before requesting any other Diskette function.

If BIOS has just been initialized, the caller must request this function to set the diskette subsystem to a known state.

continued

Function: 05h — Reset/Initialize Diskette Subsystem, Continued

Testing the Diskette Change Line signal status

The Diskette Change Line signal status is tested by the BIOS Diskette Service before functions 08h Read Diskette, 09h Write to Diskette, 0Ah Format Diskette, or 0Bh Verify Diskette Data are processed.

Function 03h Read Device Parameters may be requested to determine if the Diskette Change Line signal status is supported by the specified diskette drive.

Diskette Change Line signal

The Diskette Change Line signal has two states: set and not set.

- **Set**

The diskette has been changed or the diskette drive door is open. Diskette data cannot be read or written in this state. Diskette media status cannot be determined while the Change Line is set.

- **Not set**

The diskette drive door is closed, information can be read from and written to the diskette, and all status information can be read and modified.

Turning off the diskette motor

If it is known that the requested function is the last diskette operation and the Return Code is 0000h Successful Operation, the caller can turn off the diskette motor by requesting function 0Fh, Turn Diskette Motor Off. The recommended turn-off delay value can be found by requesting function 03h, Read Device Parameters.

Track switching

The BIOS Diskette Service does not support a switch from head 1 of a diskette cylinder to head 0 of the next cylinder. However, it allows you to cross track boundaries if you switch from head 0 to head 1 on the same cylinder.

continued

Function: 05h — Reset/Initialize Diskette Subsystem, Continued

Switching between BIOS and CBIOS

Request this function after switching from CBIOS to BIOS and before invoking any other BIOS Diskette Service functions.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0005h)	
08h	Word	Reserved (initialize to 0000h)	
0Ah	Word	Reserved (initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Return Code
10h	Word	Reserved (Initialize to 0000h)	Time-out

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
8000h	Device Busy, Request Refused
9009h	Controller Failure in Reset Operation
A120h	Controller Failure
B020h	Controller Failure
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 06h — Reserved

Function: 07h — Disable Diskette

Description

This function is a single-staged request that disables the diskette interrupt. BIOS reads the diskette controller result bytes and returns them at offsets 3Ah and 3Bh in the Request Block and then shuts down the diskette interrupt. The BIOS resets the diskette subsystem if there is a hardware error.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0007h)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		
10h	Word	Reserved (Initialize to 0000h)	
3Ah	Word		Diskette Result Byte 0 (Status Register 0), where: Bits 7-6 = Interrupt Code 00b = Command ends normally 01b = Commands ends abnormally 10b = Invalid command 11b = Ready Line state changed Bit 5 = 1 Seek End Bit 4 = 1 Equipment Check Bit 3 = 1 Not Ready Bit 2 = 1 Head Address Bit 1 = 1 Unit 1 selected Bit 0 = 1 Unit 0 selected
3Bh	Word		Diskette Result Byte 1 (Present Cylinder Number)

continued

Function: 07h — Disable Diskette, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
8000h	Device Busy, Request Refused
9009h	Controller Failure in Reset Operation
9120h	All Diskette Controller Register Bytes Not Read
9180h	Either Data is Not Ready Or the Transfer Direction is Incorrect
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 08h — Read Diskette

Description

This function is a discrete multistaged request that reads data from the specified cylinder, head, and sector number on the specified diskette drive to the memory location specified in Data Pointer 2, if the Diskette Change Line signal status is not set (the diskette door is closed).

If a diskette is formatted with a sector size other than 512 bytes, function 04h Set Device Parameters must be issued, resetting the sector size, before executing this function.

The BIOS resets the diskette subsystem if there is a hardware error.

Turning off the diskette motor

If it is known that the requested function is the last diskette operation and the Return Code is 0000h Successful Operation, the caller can turn off the diskette motor by requesting function 0Fh, Turn Diskette Motor Off. The recommended turn-off delay value can be found by requesting function 03h, Read Device Parameters.

Track switching

The BIOS Diskette Service does not support a switch from head 1 of a diskette cylinder to head 0 of the next cylinder. However, it allows you to cross track boundaries if you switch from head 0 to head 1 on the same cylinder.

continued

Function: 08h — Read Diskette, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0008h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	Word	Reserved (Initialize to 0000h)		
12h	DWord	Reserved		
16h	Word	Reserved (Initialize to 0000h)		
18h	Word	Reserved (Initialize to 0000h)		
1Ah	DWord	Data pointer 2 (32-bit physical address required)		
1Eh	Word	Reserved (Initialize to 0000h)		
20h	DWord		Time to wait before continuing request (microseconds) — Valid only if Return Code is 0002h.	
24h	Word	Number of sectors to read	Number of sectors read	
26h	Word	Cylinder number (0-based)		
2Ah	Byte	Head number (0-based)		
31h	Word	Sector number		

continued

Function: 08h — Read Diskette, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation If the Number of Sectors to Read field is zero, no action is performed and this Code is generated.
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
8000h	Device Busy, Request Refused
8003h	Write Attempted on Write-Protected Diskette
8006h	Media Changed Generated if the BIOS can reset the Diskette Change Line signal to not set; data may be transmitted.
800Dh	Media Not Present Generated if the BIOS cannot reset the Diskette Change Line signal to not set; no data is transferred.
9009h	Controller Failure in Reset Operation
9102h	Address Mark Not Found
9104h	Requested Sector Not Found
9108h	DMA Overrun on Operation
9110h	Bad CRC on Diskette Read
9120h	Controller Failure
9140h	Seek Operation Failed
9180h	Error
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit
C004h	Invalid Request Block Length
C00Ch	Unsupported Media Type/Unestablished Media
FFFFh	Return Code Field Not Valid

Function: 09h — Write to Diskette

Description

This function is a discrete multistaged request that writes the specified number of sectors of data from the memory location specified by Data Pointer 2 to the specified cylinder, head, and sector number on the specified diskette drive if the diskette Change Line signal status is not set (the diskette door is closed).

If a diskette used in the system's diskette drive(s) was formatted with a sector size other than 512 bytes, function 04h Set Device Parameters must be issued, resetting the sector size, before executing function 09h Write to Diskette.

The BIOS resets the diskette subsystem if there is a hardware error.

Turning off the diskette motor

If it is known that the requested function is the last diskette operation for awhile and the Return Code is 0000h Successful Operation, the caller can turn off the diskette motor by requesting function 0Fh, Turn Diskette Motor Off. The recommended turn-off delay value can be found by requesting function 03h, Read Device Parameters.

Track switching

The BIOS Diskette Service does not support a switch from head 1 of a diskette cylinder to head 0 of the next cylinder. However, it allows you to cross track boundaries if you switch from head 0 to head 1 on the same cylinder.

continued

Function: 09h — Write to Diskette, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0009h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	Word	Reserved (Initialize to 0000h)		
12h	DWord	Reserved		
16h	Word	Reserved (Initialize to 0000h)		
18h	Word	Reserved (Initialize to 0000h)		
1Ah	DWord	Data pointer 2 (32-bit physical address required)		
1Eh	Word	Reserved (Initialize to 0000h)	Time to wait before continuing request (microseconds) — Valid only if Return Code is 0002h.	
20h	DWord			
24h	Word	Number of sectors to write	Number of sectors written	
26h	Word	Cylinder number (0-based)		
2Ah	Byte	Head number (0-based)		
31h	Word	Sector number		

continued

Function: 09h — Write to Diskette, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation If the Number of Sectors to Read field is zero, no action is performed and this Code is generated.
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
8000h	Device Busy, Request Refused
8003h	Write Attempted on Write-Protected Diskette
8006h	Media Changed Generated if the BIOS can reset the Diskette Change Line signal to not set; data may be transmitted.
800Dh	Media Not Present Generated if the BIOS cannot reset the Diskette Change Line signal to not set; no data is transferred.
9009h	Controller Failure in Reset Operation
9102h	Address Mark Not Found
9104h	Requested Sector Not Found
9108h	DMA Overrun on Operation
9110h	Bad CRC on Diskette Read
9120h	Controller Failure
9140h	Seek Operation Failed
9180h	Error
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit
C004h	Invalid Request Block Length
C00Ch	Unsupported Media Type/Unestablished Media
FFFFh	Return Code Field Not Valid

Function: 0Ah — Format Diskette

Description

This function is a discrete multistaged request that formats the media in the specified drive if the Diskette Change Line signal status is not set (the diskette drive door is closed).

If a diskette is formatted with a sector size other than 512 bytes, function 04h Set Device Parameters must be issued, resetting the sector size, before executing function 0Ah Format Diskette.

The BIOS resets the diskette subsystem if there is a hardware error.

Formatting a diskette

To format a diskette:

1. Construct a table with one entry (the Address field) for each sector on the track. The table entries (Address fields) must have the format described below.
 2. Place the table in a buffer and place the address of the buffer in Data Pointer 2.
 3. Request function 0Dh Set Media Type for Format. The field ID for each sector is written sequentially from the buffer to each sector in the specified track.
-

Address field format

Each Address field consists of the following four bytes, in order:

- Track number (C)
- Head number (H)
- Sector number (R)
- Sector size (N)

Each sector on the track must be represented by one entry, consisting of one byte each containing the data items above, in order, in the buffer pointed to by Data Pointer 2.

continued

Function: 0Ah — Format Diskette, Continued

Return Code Processing

If the Return Code is 8006h Media Changed or 800Dh Media Not Present, request function 0Dh Set Media Type for Format before invoking function 0Ah Format Diskette again.

Turning off the diskette motor

If it is known that the requested function is the last diskette operation and the Return Code is 0000h Successful Operation, the caller can turn off the diskette motor by requesting function 0Fh, Turn Diskette Motor Off. The recommended turn-off delay value can be found by requesting function 03h, Read Device Parameters.

Track switching

The BIOS Diskette Service does not support a switch from head 1 of a diskette cylinder to head 0 of the next cylinder. However, it allows you to cross track boundaries if you switch from head 0 to head 1 on the same cylinder.

continued

Function: 0Ah — Format Diskette, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Ah)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	Word	Reserved (Initialize to 0000h)		
12h	DWord	Reserved		
16h	Word	Reserved (Initialize to 0000h)		
18h	Word	Reserved (Initialize to 0000h)		
1Ah	DWord	Data pointer 2 (32-bit physical)		
1Eh	Word	Reserved (Initialize to 0000h)		
20h	DWord		Time to wait before continuing request (microseconds). Valid only if Return Code is 0002h.	
24h	Word	Subfunction number		
26h	Word	Cylinder number (0-based)		
2Ah	Byte	Head number (0-based)		

continued

Function: 0Ah — Format Diskette, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation If the Number of Sectors to Read field is zero, no action is performed and this Code is generated.
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
8000h	Device Busy, Request Refused
8003h	Write Attempted on Write-Protected Diskette
8006h	Media Changed Generated if the BIOS can reset the Diskette Change Line signal to not set; data may be transmitted.
800Dh	Media Not Present Generated if the BIOS cannot reset the Diskette Change Line signal to not set; no data is transferred.
9009h	Controller Failure in Reset Operation
9102h	Address Mark Not Found
9104h	Requested Sector Not Found
9108h	DMA Overrun on Operation
9110h	Bad CRC on Diskette Read
9120h	Controller Failure
9140h	Seek Operation Failed
9180h	Error
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit
C004h	Invalid Request Block Length
C00Ch	Unsupported Media Type/Unestablished Media
FFFFh	Return Code Field Not Valid

Function: 0Bh — Verify Diskette Sectors

Description

This function is a discrete multistaged request that verifies the data residing on the specified number of sectors on the specified cylinder number, head number, and sector number on the drive if the the diskette drive door is closed. The data is not read, but is determined to be accurate.

If a diskette used in the system's diskette drive(s) was formatted with a sector size other than 512 bytes, function 04h Set Device Parameters must be issued, resetting the sector size, before executing this function.

The BIOS resets the diskette subsystem if there is a hardware error.

Turning off the diskette motor

If it is known that the requested function is the last diskette operation and the Return Code is 0000h Successful Operation, the caller can turn off the diskette motor by requesting function 0Fh, Turn Diskette Motor Off. The recommended turn-off delay value can be found by requesting function 03h, Read Device Parameters.

Track switching

The BIOS Diskette Service does not support a switch from head 1 of a diskette cylinder to head 0 of the next cylinder. However, this service will allow you to cross track boundaries if you switch from head 0 to head 1 on the same cylinder.

continued

Function: 0Bh — Verify Diskette Sectors, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Bh)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
16h	Word	Reserved (Initialize to 0000h)		
1Eh	Word	Reserved (Initialize to 0000h)		
20h	DWord		Time to wait before continuing request (microseconds). Valid only if Return Code is 0002h.	
24h	Word	Number of sectors to verify	Number of sectors verified	
26h	Word	Cylinder number (0-based)		
2Ah	Byte	Head number (0-based)		
31h	Word	Sector number		

continued

Function: 0Bh — Verify Diskette Sectors, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation If the Number of Sectors to Read field is zero, no action is performed and this Code is generated.
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
8000h	Device Busy, Request Refused
8003h	Write Attempted on Write-Protected Diskette
8006h	Media Changed Generated if the BIOS can reset the Diskette Change Line signal to not set; data may be transmitted.
800Dh	Media Not Present Generated if the BIOS cannot reset the Diskette Change Line signal to not set; no data is transferred.
9009h	Controller Failure in Reset Operation
9102h	Address Mark Not Found
9104h	Requested Sector Not Found
9108h	DMA Overrun on Operation
9110h	Bad CRC on Diskette Read
9120h	Controller Failure
9140h	Seek Operation Failed
9180h	Error
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit
C004h	Invalid Request Block Length
C00Ch	Unsupported Media Type/Unestablished Media
FFFFh	Return Code Field Not Valid

Function: 0Ch — Read Media Parameters

Description

This function is a discrete multistaged request. It returns the parameters of the media used in the preceding diskette function, if the Diskette Change Line signal status is not set.

This function is intended to be used after requests for either function 08h Read Diskette, 09h Verify Diskette, or 0Ah Format Diskette.

The BIOS resets the diskette subsystem if there is a hardware error.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Ch)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		
0Eh	Word			Return Code
10h	Word			Time-out
12h	Word			Number of sectors per track
			Size of sector (bytes)	
			00h = Reserved	
			01h = 256 bytes per sector	
			02h = 512 bytes per sector	
			03h-FFFFh = Reserved	
16h	Word	Reserved (Initialize to 0000h)		
26h	Word		Number of cylinders	
2Ah	Byte		Number of heads	
31h	Byte		Gap Length for Read/Write/Verify	
32h	Byte		Gap Length for format	
33h	Byte		Data Length	

continued

Function: 0Ch — Read Media Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation If the Number of Sectors to Read field is zero, no action is performed and this Code is generated.
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
8000h	Device Busy, Request Refused
8003h	Write Attempted on Write-Protected Diskette
8006h	Media Changed Generated if the BIOS can reset the Diskette Change Line signal to not set.
800Dh	Media Not Present Generated if the BIOS cannot reset the Diskette Change Line signal to not set.
9009h	Controller Failure in Reset Operation
9102h	Address Mark Not Found
9104h	Requested Sector Not Found
9108h	DMA Overrun on Operation
9110h	Bad CRC on Diskette Read
9120h	Controller Failure
9140h	Seek Operation Failed
9180h	Error
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit
C004h	Invalid Request Block Length
C00Ch	Unsupported Media Type/Unestablished Media
FFFFh	Return Code Field Not Valid

Function: 0Dh — Set Media Type for Format

Description

This function is a single-staged request that sets the media information used in the format function. This information includes the number of tracks and the number of sectors per track. These parameters are used until changed by function 04h Set Device Parameters, or until the drive door is opened (changing the Change Line signal status to set).

The BIOS resets the diskette subsystem if there is a hardware error.

Warning: If the sector size is changed by this function, the caller must restore the sector size to its original value by invoking function 04h Set Device Parameters immediately after using function 0Dh.

BIOS diskette processing

If the diskette has been changed since the last time function 0Dh was requested and a diskette is in the drive, the BIOS sets the specified parameters and resets the Diskette Change Line signal to not set.

Note: request this function before invoking function 0Ah Format Diskette.

continued

Function: 0Dh — Set Media Type for Format, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Dh)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	Word	Number of sectors per track		
12h	Word	Size of sector (bytes) 01h = 256 bytes per sector 02h = 512 bytes per sector		
16h	Word	Reserved (Initialize to 0000h)		
20h	DWord		Time to wait before continuing request (microseconds). Valid only if Return Code is 0002h.	
26h	Word	Number of tracks to format		
2Ch	Byte	Fill byte for format		
32h	Byte	Gap Length for format		

continued

Function: 0Dh — Set Media Type for Format, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation If the Number of Sectors to Read field is zero, no action is performed and this Code is generated.
8000h	Device Busy, Request Refused
8006h	Media Changed Generated if the BIOS can reset the Diskette Change Line signal to not set; data may be transmitted.
800Dh	Media Not Present Generated if the BIOS cannot reset the Diskette Change Line signal to not set (the diskette drive door is open); no data is transferred.
800Fh	Invalid Value in CMOS RAM
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Diskette Parameter
C00Ch	Unsupported Media Type/Unestablished Media Invalid input in the Number of Tracks to Format or Number of Sectors Per Track fields causes this code to be set.
FFFFh	Return Code Field Not Valid

Function: 0Eh — Read Change Line Signal Status

Description

This function is a single-staged request. It determines if the drive door of the specified diskette drive has been opened since the last time the change line was cleared. The Change Line Signal Status field is valid only if the specified drive supports the Diskette Change Line signal. Function 03h, Read Device Parameters specifies if the Change Line signal status is supported.

The BIOS resets the diskette subsystem if there is a hardware error.

This function does not clear the Diskette Change Line signal.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (000Eh)	
08h	Word	Reserved (initialize to 0000h)	
0Ah	Word	Reserved (initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Return Code
10h	Byte		Time-out
			Change Signal Status, where: 00h = Change signal not set 06h = Change signal set
16h	Word	Reserved (initialize to 0000h)	

continued

Function: 0Eh — Read Change Line Signal Status, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8000h	Device Busy, Request Refused
800Eh	Change Line Signal Not Available
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Fh — Turn Diskette Motor Off

Description

This function is a single-staged request that turns off the diskette drive motor for the specified diskette drive. The caller should request this function when the 0000h Successful Operation Return Code is set for those functions that may turn the motor on:

- 05h Reset/Initialize Diskette
- 08h Read Diskette
- 09h Write to Diskette
- 0Ah Format Diskette
- 0Bh Verify Diskette Data

The BIOS resets the diskette subsystem if there is a hardware error.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (000Fh)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Return Code
10h	Word		Time-out
16h	Byte	Reserved (Initialize to 0000h)	

continued

Function: 0Fh — Turn Diskette Motor Off, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8000h	Device Busy, Request Refused
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 10h — Interrupt Status

Description

This function is a single-staged request that returns the diskette interrupt pending status. This function does not reset the interrupt condition.

The BIOS resets the diskette subsystem if there is a hardware error.

Track switching

The BIOS Diskette Service does not support a switch from head 1 of a diskette cylinder to head 0 of the next cylinder. However, it allows you to cross track boundaries if you switch from head 0 to head 1 on the same cylinder.

continued

Function 10h — Interrupt Status, Continued

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0010h)	
08h	Word	Reserved (initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Interrupt Pending Status 00h = No interrupt 01h = Interrupt pending
10h	Byte		
16h	Word	Reserved (initialize to 0000h)	

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8000h	Device Busy, Request Refused
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Chapter 9

ABIOS Fixed Disk Service

Overview

Description

The ABIOS Fixed Disk Service provides access to fixed disk I/O through a series of function requests. All ABIOS Fixed Disk Service functions disable interrupts upon completion.

DMA interface

The ABIOS Fixed Disk Service accesses the ABIOS DMA Service. If the ABIOS Fixed Disk Service is used, the ABIOS DMA Service must also be initialized.

continued

Rules for using the PS/2 BIOS fixed disk services

Because the CBIOS and ABIOS interface with different operating systems, either the CBIOS or ABIOS Fixed Disk Service may place fixed disk hardware in an unknown state. The CBIOS does not inform the ABIOS of the fixed disk hardware state when control is passed from the CBIOS to the ABIOS, and vice versa. As a result, it is possible for one part of the PS/2 ROM BIOS to put the fixed disk hardware in a state that will not be recognized by the other part of the BIOS.

Follow the rules listed below to avoid fixed disk problems in the PS/2 BIOS:

1. If there is an outstanding ABIOS Fixed Disk function request, do not request a CBIOS Fixed Disk function.
2. If there is an outstanding CBIOS Fixed Disk function request, do not request an ABIOS Fixed Disk function.
3. If using CBIOS and the last Fixed Disk Service function call was to ABIOS, request INT 13h, AH = 0Dh, Alternate Reset Fixed Disk System, before invoking other CBIOS Fixed Disk functions.
4. If using ABIOS and the last Fixed Disk Service function call was to CBIOS, request ABIOS Fixed Disk Service function 05h, Reset Fixed Disk System, before invoking other ABIOS Fixed Disk Service functions.
5. After ABIOS is initialized, the first ABIOS Fixed Disk Service function request must be function 05h, Reset Fixed Disk System.

Fixed disk drive access

In the ABIOS, fixed disk drives are specified by Logical ID. All fixed disk drives may share the same Logical ID, with each drive assigned a separate unit number. For example, all fixed disk drives may be Logical ID 11; fixed disk drive 0 may be Logical ID 11, Unit 0 and fixed disk drive 1 may be Logical ID 11, Unit 1.

continued

Fixed Disk Relative Block addressing

The BIOS Fixed Disk Service functions 08h, 09h, 0Ah, and 0Bh require Relative Block Addresses as input in the Request Block that invokes these functions. An RBA is constructed of three elements. The derivation of these elements is as follows:

1. Multiply the Number of Sectors Per Track times the Head Number times the Cylinder Number to arrive at the first element of the RBA.
2. Multiply the Number of Sectors Per Track times the Head Number for the second element of the RBA.
3. The third element is the Sector ID minus one.

Add all three elements together to arrive at the RBA.

Relative Block Addresses begin at zero. In relative block addressing the first data block is at location 0 on the fixed disk (Cylinder 0, Head 0, Sector 1).

Relative Block Address input

Function 03h Read Device Parameters of the BIOS Fixed Disk Service returns actual physical values for the number of sectors per track, number of heads, and number of cylinders. When multiplied together, these three values indicate the number of Relative Block Addresses on the fixed disk. The largest Relative Block Address is one less than that number.

continued

Summary of BIOS Fixed Disk Service functions

Function	Description
00h	Default Interrupt Handler
01h	Return Logical ID Parameters
02h	Reserved
03h	Read Device Parameters
04h	Reserved
05h	Reset/Initialize Fixed Disk
06h-07h	Reserved
08h	Read Fixed Disk
09h	Write to Fixed Disk Drive
0Ah	Write and Verify Fixed Disk
0Bh	Verify Fixed Disk Data
0Ch	Fixed Disk Interrupt Status

In this chapter

This chapter includes information about the following topics:

- Hardware Environment
 - Fixed Disk Service Parameters Table
 - Error Handling
 - BIOS Fixed Disk Service functions
-

Hardware Environment

Fixed disk controller

The BIOS supports up to two fixed disks. The BIOS supports a PS/2-compatible ST506 fixed disk adapter. The ST506 adapter is single-tasking and must complete one operation before starting another, even though the next operation may be for the other fixed disk. The hardware interrupt request for fixed disk is 14.

RLL and ESDI fixed disk controllers may also be supported by the BIOS.

Data Transfer characteristics

The standard data transfer rate for the fixed disks is 500 kilobits per second (Kbs). The largest contiguous block of data that can be transferred at any one time is 255 sectors. The time-out value for all fixed disk functions is 22 seconds.

Fixed Disk Service Parameters Table

Table of fixed disk definitions

The defined entries into the Fixed Disk Parameters table are listed below. Wherever possible, the manufacturer name and model number associated with a given drive type are listed in the column, "Manufacturer."

Type	Manufacturer	Cyl.	Heads	Write Precomp	Landing Zone	Sectors/Track	Defect Map
1	IBM 10 MB	306	4	128	305	17	No
2	IBM 20 MB Seagate ST-225 CDC Wren II 9415-5-25 Miniscribe 8438F	615	4	300	615	17	No
3	IBM 30 MB	615	6	300	615	17	No
4	IBM 62 MB	940	8	512	940	17	No
5	IBM 46 MB	940	6	512	940	17	No
6	IBM 20 MB Miniscribe MS 8425 Seagate ST4026 Tandon TM 262 Tandon TM 702AT	615	4	0FFFFh*	615	17	No
7	IBM 30 MB	462	8	256	511	17	No
8	IBM 30 MB Seagate ST-4038 CDC Wren II 9415-5-38 Tandon TM 703AT	733	5	0FFFFh*	733	17	No
9	IBM 112 MB	900	15	0FFFFh*	901	17	No
10	IBM 20 MB	820	3	0FFFFh*	820	17	No
11	IBM 35 MB	855	5	0FFFFh*	855	17	No
12	IBM 49 MB	855	7	0FFFFh*	855	17	No
13	IBM 20 MB	306	8	128	319	17	No
<p>* If a table entry contains 0FFFFh for Write Precompensation, then there is no write precompensation for this disk. If the Write Precompensation is zero, then there is write precompensation for all cylinders.</p>							

continued

Fixed Disk Service Parameters Table, Continued

Table of fixed disk definitions, cont'd

Type	Manufacturer	Cyl.	Heads	Write Precomp	Landing Zone	Sectors/Track	Defect Map
14	IBM 42 MB	733	7	0FFFFh*	733	17	No
15	Not used						
16	IBM 20 MB	612	4	0*	663	17	No
17	IBM 40 MB	977	5	300	977	17	No
18	IBM 56 MB	977	7	0FFFFh*	977	17	No
19	IBM 59 MB	1024	7	512	1023	17	No
20	IBM 30 MB	733	5	300	732	17	No
21	IBM 42 MB	733	7	300	732	17	No
22	IBM 30 MB	733	5	300	733	17	No
23	IBM 10 MB	306	4	0*	336	17	No
24	IBM 20 MB	612	4	305	663	17	No
25	IBM 10 MB	306	4	0FFFFh*	340	17	No
26	IBM 20 MB	612	4	0FFFFh*	670	17	No
27	IBM 40.5 MB	698	7	300	732	17	Yes
28	IBM 40.5 MB	976	5	488	977	17	Yes
29	IBM 10 MB	306	4	0FFFFh*	340	17	No
30	IBM 20 MB	611	4	306	663	17	Yes
31	IBM 42.5 MB	732	7	300	732	17	Yes
32	IBM 42.5 MB	1023	5	0FFFFh*	1023	17	Yes
<p>* If a table entry contains 0FFFFh for Write Precompensation, then there is no write precompensation for this disk. If the Write Precompensation is zero, then there is write precompensation for all cylinders.</p>							

Error Handling

How errors are reported

ABIOS signals the status (Successful, Resume Stage after Interrupt, etc.) of each ABIOS request by returning a one word Return Code at offset 0Ch in the Request Block.

If Bit 15 of the Return Code field is set, the fixed disk function requested has an error. The caller's Return Code handler routine should then test Bits 14, 13, 12, and 8 to determine the class of error that has occurred. The return code handler routine should then test the remaining bits to determine the precise nature of the error.

Retryable fixed disk errors

When Bit 8 (Retryable Error) of the Return Code is set, the request should be retried. The recommended number of retries is returned by Diskette function 03h, Read Device Parameters (in offset 1Dh in the Request Block).

Function: 00h — Default Interrupt Handler

Description

This function is a single-staged request that handles unexpected hardware interrupts by resetting the interrupt at the device level. It is invoked through the interrupt routine.

When invoked

This function is invoked by calling the interrupt routine with a function code of 0000h. It is only invoked if a given Logical ID has no outstanding Request Blocks waiting for an interrupt.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0000h)	
08h	Word	Reserved (initialize to 0000h)	
0Ah	Word	Reserved (initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0005h	Not My Interrupt, Resume Stage after Interrupt
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 01h — Return Logical ID Parameters

Description

This function is a single-staged request that returns the parameters for the specified Logical ID.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length (20h)		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0001h)		
08h	Word	Reserved (initialize to 0000h)		
0Ah	Word	Reserved (initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Hardware Interrupt Level (14h, 0Eh)
10h	Byte			Arbitration Level (00h through 0Eh)
11h	Byte			Device ID (0002h)
12h	Word		Count of Units	
14h	Word		Logical ID Flags Bits 15-4 = Reserved Bit 3 = 0 No overlap across units Bit 2 = 0 Reserved Bits 1-0 = Transfer Data Pointer Mode 10b = Physical Pointer Required	
16h	Word		Request Block Length (for other functions)	
18h	Word		Secondary Device ID	
1Ah	Byte		Revision Level	
1Bh	Byte			
1Ch	Word		Reserved (initialize to 0000h)	
1Eh	Word	Reserved (initialize to 0000h)		

continued

Function: 01h — Return Logical ID Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 02h — Reserved

Function: 03h — Read Device Parameters

Description

This function is a single-staged request that returns the parameters for this device in the Request Block as specified below. The information returned applies to the maximum capacity of the media for the specified drive type. All BIOS Fixed Disk Service functions disable interrupts upon completion.

Software retries

This function returns a recommendation for the Number of Retries (at offset 1Dh) performed by the caller when a Retryable error (Bit 8 in the Return Code field) is set.

Fixed disk hardware error conditions

The BIOS resets the Fixed Disk subsystem if there is a fixed disk hardware error.

Request Block structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0003h)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		
10h	Word		Sectors per track associated with requested unit
12h	Word		Size of Sector (bytes) 00h-01h = Reserved 02h = 512-byte sectors 03h-FFFFh = Reserved
16h	Word		Drive Type, where: 00h = Drive Type 1 01h = Drive Type 2, etc, up to 21h = Drive Type 33

continued

Function: 03h — Read Device Parameters, Continued

Request Block structure, cont'd

Offset	Size	Input:	Output:
14h	Word		Device Control Flags Bits 15-13 = Reserved Bits 12-11 = Format support 00 No format support 01 Format track support 10 Format unit support 11 Track/unit support Bit 10 = ST506 drive 0 No ST506 1 ST506 Bit 9 = Concurrent unit requests 0 Not concurrent 1 Concurrent Bit 8 = Ejecting capability 0 No ejecting 1 Ejecting Bit 7 = Media organization 0 Random 1 Sequential Bit 6 = Locking capability 0 No locking 1 Locking Bit 5 = Read capability 0 Not readable 1 Readable Bit 4 = Caching support 0 No caching 1 Caching Bit 3 = Write frequency 0 Write once 1 Write many Bit 2 = 1 Change signal supported Bits 1-0 = Reserved
18h	DWord		Number of cylinders associated with requested unit
1Ch	Byte		Number of heads associated with requested unit
1Dh	Byte		Suggested number of software retries
20h	DWord		Number of Block Addresses associated with requested unit
24h	DWord	Reserved (Initialize to 0000h)	Reserved
28h	Word		Reserved
2Ch	Word		Maximum number of blocks to transfer for one call

continued

Function: 03h — Read Device Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 04h — Reserved

Function: 05h — Reset/Initialize Fixed Disk

Description

This function is a discrete multistaged request that resets the fixed disk subsystem (both hardware and software) to a known state. Setting the fixed disk drives to a known state involves resetting the fixed disk controller, which generates an interrupt to which the caller must respond, and recalibrating the drives on the next fixed disk access. All ABIOs Fixed Disk Service functions disable interrupts upon completion.

continued

Function: 05h — Reset/Initialize Fixed Disk, Continued

Switching between BIOS and CBIOS

Invoke this function after switching from CBIOS to BIOS and before invoking any other BIOS Fixed Disk Service functions.

Software retries

Function 03h Read Device Parameters returns the Number of Retries (at offset 1Dh in the Request Block) to attempt for a Fixed Disk Service function when a Retryable error (Bit 8 in the Return Code field) is set.

Fixed disk hardware error conditions

The BIOS resets the Fixed Disk subsystem if there is a fixed disk hardware error.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0005h)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out
10h	Word	Reserved (Initialize to 0000h)	Time to Wait Before Continuing request (microseconds). Valid only if Return Code is 0002h.
28h	DWord		

continued

Function: 05h — Reset/Initialize Fixed Disk, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation: If the Number of Blocks to Read field is zero, no processing occurs, and this code is generated.
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
0005h	Not My Interrupt, Resume Stage after Interrupt
8000h	Device Busy, Request Refused
800Fh	DMA Arbitration Level Out of Range
9001h	Bad Command
9002h	Address Mark Not Found
9004h	Record Not Found
9005h	Reset Failed
9007h	Controller Parameter Activity Failed
900Ah	Defective Sector
900Bh	Bad Track
900Dh	Invalid Sector on Format
900Eh	CAM Detected During Read or Verify
9010h	Uncorrectable ECC or CRC Error
9020h	Bad Controller
9021h	Equipment Check
9040h	Bad Seek
9080h	Device Did Not Respond
90AAh	Drive Not Ready
90BBh	Undefined Error
90CCh	Write Fault
90FFh	Incomplete Sense Operation
9105h	Reset Failed
9107h	Controller Parameter Activity Failed

continued

Function: 05h — Reset/Initialize Fixed Disk, Continued

Return Codes, cont'd

Code	Description
9120h	Bad Controller
9121h	Equipment Check
9140h	Bad Seek
9180h	Device Did Not Respond
91AAh	Drive Not Ready
91BBh	Undefined Error
91CCh	Write Fault
91FFh	Incomplete Sense Operation
A000h	Time-out Occurred — No Other Error
A001h	Bad Command
A002h	Address Mark Not Found
A004h	Record Not Found
A005h	Reset Failed
A007h	Parameter Activity Failed
A00Ah	Defective Sector
A00Bh	Bad Track
A00Dh	Invalid Sector on Format
A00Eh	CAM Detected During Read or Verify
A010h	Uncorrectable ECC or CRC Error
A011h	ECC Corrected Data Error
A020h	Bad Controller
A021h	Equipment Check
A040h	Bad Seek
A080h	Device Did Not Respond
A0AAh	Drive Not Ready
A0BBh	Undefined Error
A0CCh	Write Fault
A0FFh	Incomplete Sense Operation
A100h	Time-out Occurred — No Other Error

continued

Function: 05h — Reset/Initialize Fixed Disk, Continued

Return Codes, cont'd

Code	Description
A105h	Reset Failed
A107h	Controller parameter Activity Failed
A120h	Bad Controller
A121h	Equipment Check
A140h	Bad Seek
A180h	Device Did Not Respond
A1AAh	Drive Not Ready
A1BBh	Undefined Error
A1CCh	Write Fault
A1FFh	Incomplete Sense Operation
B001h	Bad Command
B020h	Bad Controller
B021h	Equipment Check
B080h	Device Did Not Respond
B0BBh	Undefined Error
B0FFh	Sense Failed
B101h	Bad Command
B120h	Bad Controller
B121h	Equipment Check
B180h	Device Did Not Respond
B1BBh	Undefined Error
B1FFh	Sense Failed
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Fixed Disk Parameter: If Number the of Blocks to Read is greater than 255, no processing occurs, and this code is returned.
FFFFh	Return Code Field Not Valid

Functions: 06h – 07h — Reserved

Function: 08h — Read Fixed Disk

Description

This function, a multistaged request, reads the amount of data specified in the Number of Blocks to Read field from the specified location on the fixed disk to the location specified in Data Pointer 2. All ABIOS Fixed Disk Service functions disable interrupts upon completion.

Fixed disk hardware error conditions

The ABIOS resets the Fixed Disk subsystem if there is a fixed disk hardware error.

Software retries

Function 03h Read Device Parameters returns the Number of Retries (at offset 1Dh in the Request Block) to attempt for a Fixed Disk Service function when a Retryable error (Bit 8 in the Return Code field) is set.

continued

Function: 08h — Read Fixed Disk, Continued

Request Block

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0008h)	
08h	Word	Reserved (initialize to 0000h)	
0Ah	Word	Reserved (initialize to 0000h)	
0Ch	Word	Return Code	Return Code
0Eh	Word		Time-out
10h	Word	Reserved (initialize to 0000h)	
12h	DWord	Reserved	
16h	Word	Reserved (initialize to 0000h)	
18h	Word	Reserved (initialize to 0000h)	
1Ah	DWord	Data Pointer 2 (32-bit physical data pointer required)	
1Eh	Word	Reserved (initialize to 0000h)	
20h	DWord	Relative Block Address	
24h	DWord	Reserved (initialize to 0000h)	
28h	DWord		Time to wait before continuing request (microseconds). Valid only if Return Code is 0002h.
2Ch	Word	Number of blocks to read (amount of data to be transferred)	Number of blocks read (not updated if Return Code is C005h). Number of blocks transferred if successful or partially successful. Only valid if request is complete.
2Eh	Byte	Caching Bits 7-1 = Reserved (set to 0) Bit 0 = Caching 0 = Caching is OK for this request 1 = Do not cache	
2Fh	Word		Soft Error Occurred: If adapter detected and corrected an error, the recovered error code is displayed. 0000h = no error.

Return Codes

All Return Codes valid for the BIOS Fixed Disk Service may be returned by the BIOS in response to this function. See the Return Codes list for Function 05h — Reset/Initialize Fixed Disk.

Function: 09h — Write to Fixed Disk

Description

This function, a discrete multistaged request, writes the amount of data specified in the Number of Blocks to Write field from the location specified in Data Pointer 2 to the Relative Block Address on the fixed disk. All ABIOs Fixed Disk Service functions disable interrupts upon completion.

Fixed disk hardware error conditions

The ABIOs resets the Fixed Disk subsystem if there is a fixed disk hardware error.

Software retries

Function 03h Read Device Parameters returns the Number of Retries (at offset 1Dh in the Request Block) to attempt for a Fixed Disk Service function when a Retryable error (Bit 8 in the Return Code field) is set.

continued

Function: 09h – Write to Fixed Disk, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0009h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	Word	Reserved (Initialize to 0000h)		
12h	DWord	Reserved		
16h	Word	Reserved (Initialize to 0000h)		
18h	Word	Reserved (Initialize to 0000h)		
1Ah	DWord	Data Pointer 2 (32-bit physical)		
1Eh	Word	Reserved (Initialize to 0000h)		
20h	DWord	Relative Block Address		
24h	DWord	Reserved (Initialize to 0000h)		
28h	DWord		Time to Wait Before Continuing Request (microseconds). Valid only if Return Code is 0002h.	
2Ch	Word	Number of Blocks to Write: If zero, no processing occurs; Return Code is 0000h; if > 255, Return Code is C005h, no processing occurs.	Number of blocks written (not updated if Return Code is C005h). Number of blocks transferred if successful or partially successful. Only valid if request is complete.	
2Eh	Byte	Caching Bits 7-1 = Reserved (set to 0) Bit 0 = Caching 0 = Caching is OK for this request 1 = Do not cache		
2Fh	Word		Soft Error Occurred: if adapter detected and corrected an error, the recovered error code is displayed. 0000h = no error.	

Return Codes

All Return Codes valid for the BIOS Fixed Disk Service may be returned by the BIOS in response to this function. See the Return Codes list for Function 05h – Reset/Initialize Fixed Disk.

Function: 0Ah — Write and Verify Fixed Disk

Description

This function, a multistaged request, writes data to the fixed disk (just as function 09h does), then immediately verifies that same data (as function 0Bh does).

Software retries

Function 03h Read Device Parameters returns the Number of Retries (at offset 1Dh in the Request Block) to attempt for a Fixed Disk Service function when a Retryable error (Bit 8 in the Return Code field) is set.

Fixed disk hardware error conditions

The BIOS resets the Fixed Disk subsystem if there is a fixed disk hardware error.

continued

Function: 0Ah — Write and Verify Fixed Disk, Continued

Request Block structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Ah)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	Word	Reserved (Initialize to 0000h)		
12h	DWord	Reserved		
16h	Word	Reserved (Initialize to 0000h)		
18h	Word	Reserved (Initialize to 0000h)		
1Ah	DWord	Data Pointer 2 (32-bit physical)		
1Eh	Word	Reserved (Initialize to 0000h)		
20h	DWord	Relative Block Address		
24h	DWord	Reserved (Initialize to 0000h)		
28h	DWord			
			Time to Wait Before Continuing Request (microseconds). Valid only if Return Code is 0002h.	
2Ch	Word	No. of Blocks to Write/Verify. — If the input is zero, no processing occurs. If the input is > 255, no processing occurs and the Return Code is set to C005h.	Number of blocks written and verified (not updated if Return Code is C005h). Number of blocks written/verified if successful or partially successful. Only valid if request is complete.	
2Eh	Byte	Caching Bits 7-1 = Reserved (set to 0) Bit 0 = Caching 0 = Caching OK 1 = Do not cache		
2Fh	Word		Soft Error Occurred: If adapter detected and corrected an error, the recovered error code is displayed. 0000h = no error.	
31h	Word	Reserved for Subfunction		

Return Codes

All Return Codes valid for the BIOS Fixed Disk Service may be returned by the BIOS in response to this function. See the Return Codes list for Function 05h – Reset/Initialize Fixed Disk.

Function: 0Bh — Verify Fixed Disk Data

Description

This function, a discrete multistaged request, verifies the readability of the data on the fixed disk. Data is read but is not transferred. All BIOS Fixed Disk Service functions disable interrupts upon completion.

Software retries

Function 03h Read Device Parameters returns the Number of Retries (at offset 1Dh in the Request Block) to attempt for a Fixed Disk Service function when a Retryable error (Bit 8 in the Return Code field) is set.

Fixed disk hardware error conditions

The BIOS resets the Fixed Disk subsystem if there is a fixed disk hardware error.

continued

Function: 0Bh — Verify Fixed Disk Data, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Bh)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
16h	Word	Reserved (Initialize to 0000h)		
18h	Word	Reserved (Initialize to 0000h)		
1Eh	Word	Reserved (Initialize to 0000h)		
20h	DWord	Relative Block Address		
24h	DWord	Reserved (Initialize to 0000h)		
28h	DWord		Time to wait before continuing request (microseconds). Valid only if Return Code is 0002h.	
2Ch	Word	Number of Blocks to Verify — If > 255, no processing occurs. If zero, Return Code is C005h.	Number of Blocks Written (contains number of blocks verified if Return Code is 0000h).	
2Eh	Byte	Caching Bits 7-1 = Reserved Bit 0 = Caching allowed		
2Fh	Word		Soft Error Occurred: If adapter detected and corrected an error, the recovered error code is displayed. 0000h = no error.	

Return Codes

All Return Codes valid for the BIOS Fixed Disk Service may be returned by the BIOS in response to this function. See the Return Codes list for Function 05h — Reset/Initialize Fixed Disk.

Function: 0Ch — Fixed Disk Interrupt Status

Description

This function, a single-staged request, returns the fixed disk controller interrupt pending status. All BIOS Fixed Disk Service functions disable interrupts upon completion.

The interrupt condition is not reset by the BIOS.

Logical ID

The Interrupt Status field applies to the fixed disk controller Logical ID and not to any individual diskette drive (Unit) entered in the Request Block by the caller.

Interrupt Status field

A parameter error in the input to the Request Block will cause the Interrupt Status field to be invalid. The Interrupt Status field indicates if any interrupts are currently pending from the interrupt controller.

Fixed disk hardware error conditions

The BIOS resets the Fixed Disk subsystem if there is a fixed disk hardware error.

Software retries

Function 03h Read Device Parameters returns the Number of Retries (at offset 1Dh in the Request Block) to attempt for a Fixed Disk Service function when a Retryable error (Bit 8 in the Return Code field) is set.

continued

Function: 0Ch — Fixed Disk Interrupt Status, Continued

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (000Ch)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Interrupt Status 00h = Interrupt not pending 01h = Interrupt pending 02h-FFh = Reserved
10h	Byte		
16h	Word	Reserved (Initialize to 0000h)	

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Fixed Disk Parameter
FFFFh	Return Code Field Not Valid

Chapter 10

ABIOS Keyboard Service

Overview

Description

The ABIOS Keyboard Service provides an interface between the operating system and IBM PS/2-compatible keyboards. The functions available provide the user with a means of accessing the keyboard which is independent of hardware specifics. To maintain maximum compatibility across different keyboard controllers, direct hardware programming of a controller should be avoided.

continued

Overview, Continued

Summary of Keyboard Service functions

Function	Description
00h	Default Interrupt Handler
01h	Return Logical ID Parameters
02h	Reserved
03h	Read Keyboard ID Bytes
04h	Reserved
05h	Reset/Initialize Keyboard
06h	Enable Keyboard
07h	Disable Keyboard
08h	Continuous Keyboard Read
09h-0Ah	Reserved
0Bh	Read Keyboard LED Status
0Ch	Set Keyboard LED Status
0Dh	Set Typematic Rate and Delay
0Eh	Read Keyboard Scan Code Mode
0Fh	Set Keyboard Scan Code Mode
10h	Write Command(s) to Keyboard Controller
11h	Write Command(s)/Data to Keyboard

In this chapter

This chapter includes information about the following topics:

- Hardware Environment
 - 101-key Keyboard Layout
 - Scan Codes
 - System Scan Codes
 - Error Handling
 - BIOS Keyboard Service functions
-

Hardware Environment

Introduction

The BIOS supports an intelligent keyboard subsystem based on the Intel 8042 or equivalent keyboard controller.

The hardware interrupt level associated with the BIOS Keyboard Service is IRQ 1.

The 8042 controller chip

The Intel 8042 peripheral controller (or compatible) is a single-chip micro-computer that can be programmed to allow bidirectional communication between the master microprocessor and up to two auxiliary serial input devices. The 8042 chip, generally, is mounted on the system motherboard. 8042 programs reside as firmware in the 8042 chip itself.

Device support

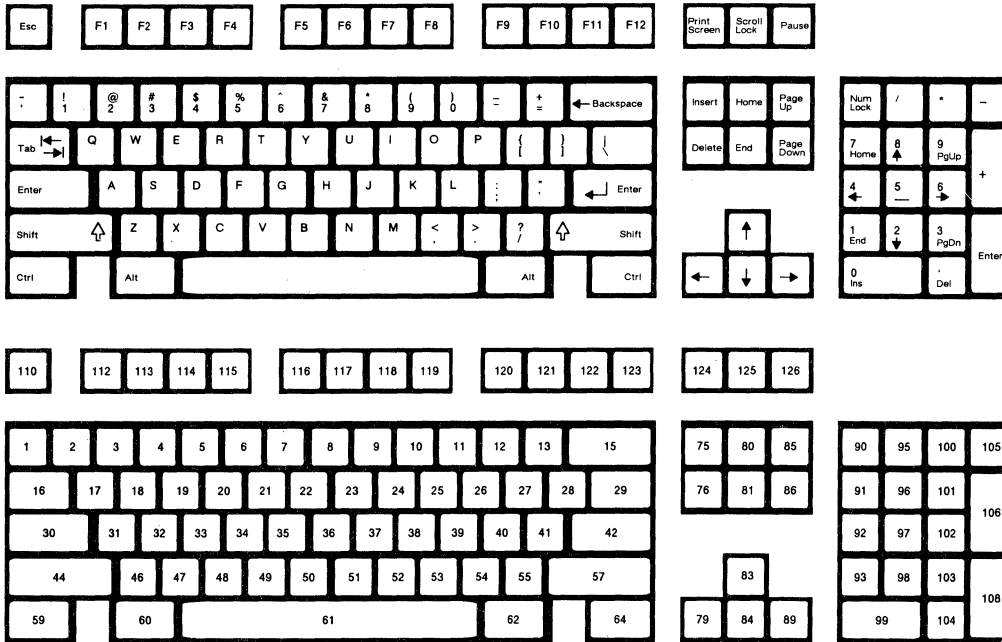
The kind of devices a given 8042 chip supports are dependent on how the 8042 is programmed.

On IBM PS/2-compatible systems, the 8042 is programmed to allow bidirectional communication between the system and the keyboard, as well as between the system and one other auxiliary serial device, such as a mouse, joystick, or trackball.

101-Key Keyboard Layout

Keyboard layout

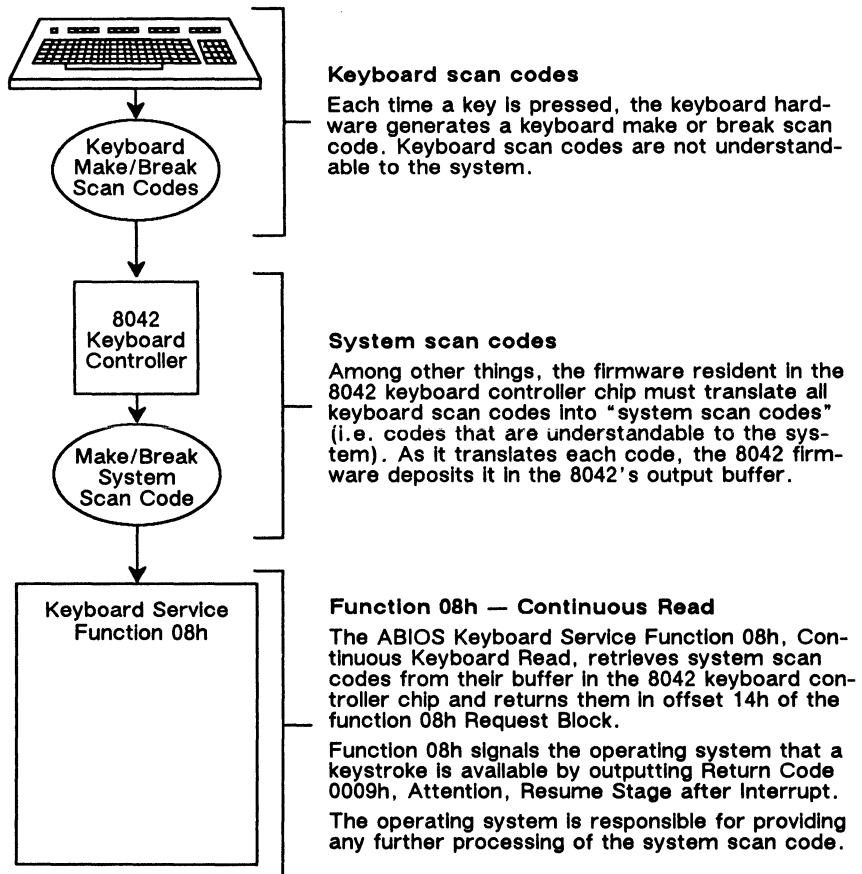
The figure below depicts the key arrangement and key number system applied to the typical 101-key keyboard.



Scan Codes

Introduction

The keyboard hardware generates two kinds of keyboard codes: keyboard scan codes and system scan codes. The illustration below defines each type of code and shows their relationship to the BIOS Keyboard Service.



System Scan Codes

The following system scan codes, including multi-byte codes, can be used for the Write Password, Write Invocation and Write Match Byte functions.

Typewriter/Function Keys

Key #	U.S. Keyboard Legend	System Scan Codes (hex)
1	~	0E
2	1!	16
3	2@	1E
4	3#	26
5	4\$	25
6	5%	2E
7	6	36
8	7&	3D
9	8*	3E
10	9(46
11	0)	45
12	-_	4E
13	=+	55
15	Backspace	66
16	Tab	0D
17	Q	15
18	W	1D
19	E	24
20	R	2D
21	T	2C
22	Y	35
23	U	3C
24	I	43
25	O	44

Key #	U.S. Keyboard Legend	System Scan Codes (hex)
26	P	4D
27	[{	54
28] }	5B
29	101-key only	5D
30	Caps Lock	58
31	A	1C
32	S	1B
33	D	23
34	F	2B
35	G	34
36	H	33
37	J	3B
38	K	42
39	L	4B
40	; :	4C
41	' "	52
42	102-key only	5D
43	Enter	5A
44	L Shift	12
45	102-key only	61
46	Z	1A
47	X	22
48	C	21
49	V	2A

continued

System Scan Codes, Continued

Typewriter/Function Keys, cont'd

Key #	U.S. Keyboard Legend	System Scan Codes (hex)
50	B	32
51	N	31
52	M	3A
53	, <	41
54	. >	49
55	/ ?	4A
57	R Shift	59
58	L Ctrl	14
60	L Alt	11
61	Space	29
62	R Alt	E0-11
64	R Ctrl	E0-14
90	Num Lock	77
91	7 Home	6C
92	4 Left	6B
93	1 End	69
96	8 Up	75
97	5	73
98	2 Down	72
99	0 Ins	70
100	*	7C

Key #	U.S. Keyboard Legend	System Scan Codes (hex)
101	9 PgUp	7D
102	6 Right	74
103	3 Page Down	7A
104	. Del	71
105	-	7B
106	+	79
108	Enter	E0-5A
110	Esc	76
112	F1	05
113	F2	06
114	F3	04
115	F4	0C
116	F5	03
117	F6	0B
118	F7	83
119	F8	0A
120	F9	01
121	F10	09
122	F11	78
123	F12	07
125	Scroll Lock	7E

continued

System Scan Codes, Continued

Other keys

The rest of the keys send a series of codes that depend on the state of the shift keys (Ctrl, Alt, and Shift) and the Num Lock key (On or Off). Since the base scan code is the same as that for another key, an additional code (hex E0) is added to the base code so that it is unique. The following four tables summarize the scan codes for these other keys.

Cursor/Control Keys

Key #	U.S. Keyboard Legend	Base Case or Shift + Num Lock	Shift Case*	Num Lock on
75	Insert	E0-70	E0 F0 12 E0 70	E0 12 E0 70
76	Delete	E0-71	E0 F0 12 E0 71	E0 12 E0 71
79	Left	E0-6B	E0 F0 12 E0 6B	E0 12 E0 6B
80	Home	E0-6C	E0 F0 12 E0 6C	E0 12 E0 6C
81	End	E0-69	E0 F0 12 E0 69	E0 12 E0 69
83	Up	E0-75	E0 F0 12 E0 75	E0 12 E0 75
84	Down	E0-72	E0 F0 12 E0 72	E0 12 E0 72
85	Page Up	E0-7D	E0 F0 12 E0 7D	E0 12 E0 7D
86	Page Down	E0-7A	E0 F0 12 E0 7A	E0 12 E0 7A
89	Right	E0-74	E0 F0 12 E0 74	E0 12 E0 74
* With the Left Shift key down, the F0 12 shift code is added to the other scan codes sent. With the Right Shift key down, F0 59 is added. When both keys are down, both sets of codes are sent with the rest of the scan code.				

continued

System Scan Codes, Continued

"/" Key on Numeric Keypad

Key #	U.S. Keyboard Legend	System Scan Codes (hex)	Shift Case*
95	/	E0 4A	E0 F0 12 E0 4A

* With the Left Shift key down, the F0 12 shift code is added to the other scan codes sent. With the Right Shift key down, F0 59 is added. When both keys are down, both sets of codes are sent with the rest of the scan code.

Print Screen/Sys Req Key

Key #	U.S. Keyboard Legend	System Scan Codes (hex)	Ctrl Case Shift Case	Alt Case
124	Print Screen	E0 12 E0 7C	E0 7C	84

Pause/Break Key

Key #	U.S. Keyboard Legend	System Scan Codes (hex)	Ctrl Key Pressed
126	Pause	E1 14 77 E1 F0 14 F0 77	E0 7E E0 F0 7E

Error Handling

Description

The BIOS Keyboard Service assumes that the operating system will handle:

- **Hardware errors**

The BIOS does not reset the keyboard after a keyboard hardware error. It is the operating system's responsibility to execute function 05h Reset/Initialize Keyboard every time there is a hardware error.

- **Time-out error retries**

The BIOS Keyboard Service resets the keyboard after time-out errors.

How errors are reported

BIOS signals the status (Successful, Resume Stage after Interrupt, etc) of each BIOS request by returning a one word Return Code at offset 0Ch in the Request Block.

If Bit 15 of the Return Code field is set, the keyboard function requested has an error. The caller's Return Code handler routine should then test Bits 14, 13, 12, and 8 to determine the class of error that has occurred. The return code handler routine should then test the remaining bits to determine the precise nature of the error.

Function: 00h — Default Interrupt Handler

Description

This single-staged function handles unexpected hardware interrupts by resetting the interrupt at the device level.

How and When to invoke

This function is invoked by calling the interrupt routine with a function code of 0000h. It is only invoked if a given Logical ID has no outstanding Request Blocks waiting for an interrupt.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0000h)	
08h	Word	Reserved (initialize to 0000h)	
0Ah	Word	Reserved (initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0005h	Not My Interrupt, Resume Stage after Interrupt
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Request Code Field Not Valid

Function: 01h — Return Logical ID Parameters

Description

This single-staged function returns the parameters for the specified Logical ID.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length (20h)	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0001h)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	Return Code
0Eh	Word		
10h	Byte		Hardware Interrupt Level (01h)
11h	Byte		Arbitration Level (FFh)
12h	Word		Device ID (0004h)
14h	Word		Count of Units
16h	Word		Logical ID flags Bits 15-4 = Reserved Bit 3 = 0 No overlap across units = 1 Overlap across units supported Bit 2 = 0 Reserved Bits 1-0 = Transfer Data Pointer Mode 00 = No Pointers Required
18h	Word		Request Block Length (for other functions)
1Ah	Byte	Reserved (Initialize to 0000h)	Secondary Device ID
1Bh	Byte		Revision Level
1Ch	Word	Reserved (Initialize to 0000h)	
1Eh	Word	Reserved (Initialize to 0000h)	

continued

Function: 01h — Return Logical ID Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 02h — Reserved

Function 03h — Read Keyboard ID Bytes

Description

This function generates a multistaged request that returns the keyboard identification code which indicates the keyboard type.

Keyboard ID code

The Keyboard ID is a two-byte code. Function 03h returns the low byte of the keyboard ID in offset 14h of the function 03h Request Block. The high byte of the keyboard ID is returned in offset 15h.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0003h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	DWord			Time, in microseconds, to Wait Before Continuing Request (Valid only if Return Code = 0002h)
14h	Byte			Keyboard ID low byte
15h	Byte		Keyboard ID high byte	
16h	Word	Reserved (Initialize to 0000h)		

continued

Function: 03h — Read Keyboard ID Bytes, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
0005h	Not My Interrupt, Resume Stage after Interrupt
0009h	Attention, Resume Stage after Interrupt
8000h	Device Busy, Request Refused
8003h	Security Enabled, Keyboard Inhibited — Request Refused
9000h	Keyboard Controller Perpetually Busy
9001h	Keyboard Failed Reset
9002h	Resend Error
9003h	Keyboard Parity Error
9004h	General Hardware Time-out
9006h	Undefined Mode Returned by Keyboard
9100h	Keyboard Controller Perpetually Busy
9101h	Keyboard Failed Reset
9102h	Resend Error
9103h	Keyboard Parity Error
9104h	General Hardware Time-out
B001h	Keyboard Error
B101h	Keyboard Error The caller should invoke function 05h Reset/Initialization after a keyboard hardware error.
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Keyboard Parameter
FFFFh	Return Code is Not Valid

Function: 04h — Reserved

Function: 05h — Reset/Initialize Keyboard

Description

This function generates a multistaged request that resets the keyboard hardware and turns off the Caps Lock, Num Lock, Scroll Lock LEDs.

Keyboard Reset

The BIOS does not reset the keyboard after a keyboard hardware error.

Note: It is the operating system's responsibility to execute function 05h Reset/Initialize Keyboard every time there is a hardware error.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0005h)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	Return Code
0Eh	Word		Time-out
10h	DWord		Time, in microseconds, to wait before continuing request (Valid only if Return Code = 0002h)
16h	Word	Reserved (Initialize to 0000h)	

continued

Function: 05h — Reset/Initialize Keyboard, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
0005h	Not My Interrupt, Resume Stage after Interrupt
0009h	Attention, Resume Stage after Interrupt
8000h	Device Busy, Request Refused
8003h	Security Enabled, Keyboard Inhibited — Request Refused
9000h	Keyboard Controller Perpetually Busy
9001h	Keyboard Failed Reset
9002h	Resend Error
9003h	Keyboard Parity Error
9004h	General Hardware Time-out
9006h	Undefined Mode Returned by Keyboard
9100h	Keyboard Controller Perpetually Busy
9101h	Keyboard Failed Reset
9102h	Resend Error
9103h	Keyboard Parity Error
9104h	General Hardware Time-out
B001h	Keyboard Error
B101h	Keyboard Error The operating system should invoke function 05h Reset/Initialization after a keyboard hardware error.
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Keyboard Parameter
FFFFh	Return Code is Not Valid

Function: 06h — Enable Keyboard

Description

This function generates a multistaged request that enables the keyboard, allowing data from the keyboard to be passed to the system.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0006h)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Return Code
10h	DWord		Time-out
16h	Word	Reserved (Initialize to 0000h)	Time to wait before continuing request (microseconds)

continued

Function: 06h — Enable Keyboard, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0002h	Resume Stage after Time Delay
8000h	Device Busy, Request Refused
8003h	Security Enabled, Keyboard Inhibited — Request Refused
9000h	Keyboard Controller Perpetually Busy
9100h	Keyboard Controller Perpetually Busy
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Keyboard Parameter
FFFFh	Return Code is Not Valid

Function: 07h — Disable Keyboard

Description

This function generates a multistaged request that disables the keyboard, inhibiting the flow of data from the keyboard to the system.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0007h)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Return Code
10h	DWord		Time-out
			Time to wait before continuing request (microseconds)
16h	Word	Reserved (Initialize to 0000h)	

continued

Function: 07h — Disable Keyboard, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0002h	Resume Stage after Time Delay
8000h	Device Busy, Request Refused
8003h	Security Enabled, Keyboard Inhibited — Request Refused
9000h	Keyboard Controller Perpetually Busy
9100h	Keyboard Controller Perpetually Busy
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Keyboard Parameter
FFFFh	Return Code is Not Valid

Function: 08h — Continuous Keyboard Read

Description

This function generates a continuous multistaged request that retrieves system scan codes as they are generated by the 8042 keyboard controller chip from I/O port 0060h and signals the operating system that a keystroke is available for processing.

When to invoke function 08h

Since no keystrokes can be processed unless function 08h has been successfully invoked, function 08h should be invoked immediately after BIOS initialization.

What is a raw system scan code

The keyboard hardware generates two kinds of keyboard codes: keyboard scan codes and system scan codes.

■ Keyboard scan codes

Each time a key is pressed, the keyboard hardware generates a keyboard make or break scan code. Keyboard scan codes are not understandable to the system.

■ System scan codes

The firmware resident in the 8042 keyboard controller chip translates all keyboard scan codes into "system scan codes" (i.e. codes that are understandable to the system). As it translates each code, the 8042 firmware deposits it in the 8042's output buffer.

Function 08h retrieves system scan codes from their buffer in the 8042 keyboard controller chip and returns them in offset 14h of the Request Block. Function 08h signals to the operating system that a keystroke is available by sending Return Code 0009h Attention, Resume Stage after Interrupt.

continued

Function: 08h — Continuous Keyboard Read, Continued

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0008h)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out
14h	Byte		Keyboard Scan Code
16h	Word	Reserved (Initialize to 0000h)	

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0005h	Not My Interrupt, Resume Stage after Interrupt
0009h	Attention, Resume Stage after Interrupt
8000h	Device Busy, Request Refused
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Keyboard Parameter
FFFFh	Return Code Field Not Valid

Functions: 09h – 0Ah – Reserved

Function: 0Bh – Read Keyboard LED Status

Description

This single-staged request returns the status of the keyboard Caps Lock, Scroll Lock, and Num Lock LEDs.

Function 0Bh limitations

The data returned by function 0Bh reflects the state of the Keyboard LED status byte after the last successful call of either function 05h, Reset/Initialize Keyboard or function 0Ch Write Keyboard LED Status.

When function 11h, Write Keyboard Command, is used to set the Keyboard LEDs, the value returned by function 0Bh may not be reliable.

continued

Function 0Bh — Read Keyboard LED Status, Continued

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (000Bh)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		
14h	Byte		
16h	Word	Reserved (Initialize to 0000h)	
			Return Code
			Time-out
			Keyboard Indicator LED Status Bits 7-3 = 1 Reserved Bit 2 = 1 Caps Lock on Bit 1 = 1 Num Lock on Bit 0 = 1 Scroll Lock on

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8000h	Device Busy, Request Refused
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Keyboard Parameter
FFFFh	Return Code Field Not Valid

Function: 0Ch — Set Keyboard LED Status

Description

This function generates a multistaged request that turns on or off the keyboard Caps Lock, Scroll Lock, and/or Num Lock LEDs.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Ch)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	DWord			Time to wait before continuing request (microseconds)
14h	Byte	Program the Keyboard LED Status Indicators Bits 7-3= 0 Reserved Bit 2 = 1 Caps Lock on Bit 1 = 1 Num Lock on Bit 0 = 1 Scroll Lock on		
16h	Word	Reserved (Initialize to 0000h)		

continued

Function: 0Ch — Set Keyboard LED Status, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
0005h	Not My Interrupt, Resume Stage after Interrupt
8000h	Device Busy, Request Refused
8003h	Security Enabled, Keyboard Inhibited — Request Refused
9000h	Keyboard Controller Perpetually Busy
9002h	Resend Error
9100h	Keyboard Controller Perpetually Busy
9101h	Keyboard Failed Reset
9102h	Resend Error
B001h	Keyboard Error
B101h	Keyboard Error The operating system should invoke function 05h Reset/Initialize Keyboard after a keyboard hardware error.
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Keyboard Parameter
FFFFh	Return Code is Not Valid

Function: 0Dh — Set Typematic Rate and Delay

Description

This function generates a single-staged or discrete multistaged request that changes or sets the typematic rate and the keystroke delay for all keys on the keyboard.

Typematic rate

The typematic rate is the maximum number of make codes per second that the keyboard can support. The rate in characters per second can be set at any of 32 values, ranging from 2 to 30 characters per second.

Keystroke delay

The keystroke delay established by this function is the delay between the period of time that elapses between a keystroke and the scan code generated by the key stroke being sent to the keyboard controller. This delay can be set at 250, 500, 750, or 1000 milliseconds.

continued

Function: 0Dh — Set Typematic Rate and Delay, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Dh)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	DWord			Time to wait before continuing request (microseconds)
14h	Byte	Typematic Rate Setting Bits 7-5= Reserved (must be 0) Bits 4-0= Rate setting in characters per second (values in binary) 00000 = 30.0 10000 = 7.5 00001 = 26.7 10001 = 6.7 00010 = 24.0 10010 = 6.0 00011 = 21.8 10011 = 5.5 00100 = 20.0 10100 = 5.0 00101 = 18.5 10101 = 4.6 00110 = 17.1 10110 = 4.3 00111 = 16.0 10111 = 4.0 01000 = 15.0 11000 = 3.7 01001 = 13.3 11001 = 3.3 01010 = 12.0 11010 = 3.0 01011 = 10.9 11011 = 2.7 01100 = 10.0 11100 = 2.5 01101 = 9.2 11101 = 2.3 01110 = 8.6 11110 = 2.1 01111 = 8.0 11111 = 2.0		
15h	Byte	Typematic Delay Setting Bits 7-2 = Reserved (must be 0) Bits 1-0 = Delay value in milliseconds (values in binary) 00 = 250 01 = 500 10 = 750 11 = 1000		
16h	Word	Reserved (Initialize to 0000h)		

continued

Function: 0Dh — Set Typematic Rate and Delay, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
0005h	Not My Interrupt, Resume Stage after Interrupt
8000h	Device Busy, Request Refused
8003h	Security Enabled, Keyboard Inhibited — Request Refused
9000h	Keyboard Controller Perpetually Busy
9002h	Resend Error
9100h	Keyboard Controller Perpetually Busy
9101h	Keyboard Failed Reset
9102h	Resend Error
B001h	Keyboard Error
B101h	Keyboard Error The operating system should invoke function 05h Reset/Initialize Keyboard after a keyboard hardware error.
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Keyboard Parameter
FFFFh	Return Code is Not Valid

Function 0Eh — Read Keyboard Scan Code Mode

Description

This function generates a discrete multistaged request that reads the keyboard scan code mode from the 8042 status port.

What is a keyboard scan code mode?

Each time a key is pressed, the keyboard hardware generates a keyboard make or break scan code. The keyboard hardware is capable of generating any of three sets of keyboard scan codes (Modes 1, 2, or 3).

Keyboard scan codes are not understandable to the system. The firmware resident in the 8042 keyboard controller chip translates all keyboard scan codes into system scan codes (i.e. codes that are understandable to the system).

Function 0Eh and the Phoenix 8042 AK/MCF

Systems equipped with the Phoenix 8042 Advanced Keyboard/Mouse Controller Firmware (AK/MCF) support keyboard scan code mode 2 exclusively. Function 0Eh returns the value corresponding to keyboard scan code mode 2 only.

continued

Function 0Eh — Read Keyboard Scan Code Mode, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Eh)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	DWord			Time to Wait Before Continuing Request (microseconds)
14h	Byte			Current Keyboard Scan Code 00h = Reserved 01h = Set to 1 02h = Set to 2 03h = Set to 3 04h-Fh = Reserved
16h	Word	Reserved (Initialize to 0000h)		

continued

Function 0Eh — Read Keyboard Scan Code Mode, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
0005h	Not My Interrupt, Resume Stage after Interrupt
8000h	Device Busy, Request Refused
8003h	Security Enabled, Keyboard Inhibited — Request Refused
9000h	Keyboard Controller Perpetually Busy
9002h	Resend Error
9003h	Keyboard Parity Error
9004h	Hardware Time-out
9006h	Undefined Mode from Keyboard
9100h	Keyboard Controller Perpetually Busy
9101h	Keyboard Failed Reset
9102h	Resend Error
9103h	Keyboard Parity Error
9104h	General Hardware Time-out
9106h	Undefined Mode from Keyboard
B001h	Keyboard Error
B101h	Keyboard Error The operating system should invoke function 05h Reset/Initialize Keyboard after a keyboard hardware error.
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Keyboard Parameter
FFFFh	Return Code Is Not Valid

continued

Function: 0Fh — Set Keyboard Scan Code Mode, Continued

Description

This function generates a multistaged request that sets the keyboard scan code mode.

What is a keyboard scan code mode

Each time a key is pressed, the keyboard hardware itself generates a keyboard make or break scan code. The keyboard hardware is capable of generating any of three sets of keyboard scan codes (Modes 1, 2, or 3).

Keyboard scan codes are not understandable to the system. The firmware resident in the 8042 keyboard controller chip translates all keyboard scan codes into system scan codes (i.e. codes that are understandable to the system).

Function 0Fh and the Phoenix 8042 AK/MCF

Systems equipped with the Phoenix 8042 Advanced Keyboard/Mouse Controller Firmware (AK/MCF) support keyboard scan code mode 2 exclusively. Function 0Fh returns the value corresponding to keyboard scan code mode 2 only.

continued

Function: 0Fh — Set Keyboard Scan Code Mode, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Fh)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	DWord			Time to Wait Before Continuing Request (microseconds)
14h	Byte	Change Current Keyboard Scan Code 00h = Reserved 01h = Set to 1 02h = Set to 2 03h = Set to 3		
16h	Word	Reserved (Initialize to 0000h)		

continued

Function: 0Fh — Set Keyboard Scan Code Mode, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
0005h	Not My Interrupt, Resume Stage after Interrupt
8000h	Device Busy, Request Refused
8003h	Security Enabled, Keyboard Inhibited — Request Refused
9000h	Keyboard Controller Perpetually Busy
9002h	Resend Error
9100h	Keyboard Controller Perpetually Busy
9101h	Keyboard Failed Reset
9102h	Resend Error
B001h	Keyboard Error
B101h	Keyboard Error The operating system should invoke function 05h Reset/Initialize Keyboard after a keyboard hardware error.
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Input Parameter
FFFFh	Return Code is Not Valid

Function 10h — Write Command(s) to Keyboard Controller

Description

This function generates a discrete multistaged request that writes a command and associated data from the system to the 8042 controller. The command consists of a single byte. The location and length of the command must be specified at function start-up. This function permits access to 8042 RAM, keyboard self-tests, interface tests, and other 8042 functions that are otherwise not available in BIOS.

Changes to 8042

The caller must not change the state of the 8042 in intermediate stages of a multistaged request for this function.

Input field considerations

The Data String Length (offset 1Ch) must be initialized to a nonzero value. If the Data String Length field is initialized to 00h, then function 10h does not perform any action and sets the Return Code field to 0000h, Successful Operation.

BIOS keyboard command processing

The first byte of the string passed to BIOS is assumed to be the command byte and is sent to I/O port 0064h. All subsequent bytes are sent to I/O port 0061h. BIOS issues a Return Code of 0001h, Resume Stage after Interrupt, between each byte sent. At this point, BIOS does not respond to the keyboard controller. It reads the 8042 status register, which indicates when commands have been accepted. Return Code 0000h is issued when all bytes have been read.

continued

Function 10h — Write Command(s) to Keyboard Controller, Continued

System-to-8042 commands

Command	Description
20h	READ THE 8042 COMMAND BYTE. This command instructs the 8042 to send the contents of location 20h, the command byte, to the system.
21h-3Fh	READ THE 8042 RAM. This command instructs the 8042 to send the contents of the RAM location, defined by bits 5-0 of the command to the system.
60h	WRITE THE 8042 COMMAND BYTE. This command instructs the 8042 to write the data byte following the command to the location of the command byte (20h).
61h-7Fh	WRITE THE 8042 RAM. This command instructs the 8042 to write the data byte following the command to the RAM location defined by bits 0-5 of the command.
A4h	TEST PASSWORD INSTALLED. This command instructs the 8042 to check whether there is a password currently installed. If there is no password, the contents of the first location where the password would be stored will be zero. If there is a password installed, FAh is placed in the 8042 output buffer, if not, F1h is placed in the buffer.
A5h	LOAD SECURITY. This command instructs the 8042 to read password data from the 8042 input buffer and store it until a null (0) is detected. The null is stored as the last byte of the password.
A6h	ENABLE SECURITY. This command instructs the 8042 to check the installed password against the incoming keystrokes for a match.
A7h	DISABLE AUXILIARY DEVICE INTERFACE. This command instructs the 8042 to set bit 5 of the command byte. This disables the auxiliary device by driving the auxiliary clock low.
A8h	ENABLE AUXILIARY DEVICE INTERFACE. This command instructs the 8042 to clear bit 5 of the command byte. This enables the auxiliary device by driving the auxiliary clock high.
A9h	AUXILIARY INTERFACE TEST. This command instructs the 8042 to test the auxiliary device clock and data lines. The result of the test is placed in the output buffer as follows: 00 — No error detected. 01 — Auxiliary device clock line is stuck low. 02 — Auxiliary device clock line is stuck high. 03 — Auxiliary device data line is stuck low. 04 — Auxiliary device data line is stuck high.
AAh	SELF TEST. This command instructs the keyboard 8042 to perform internal diagnostics tests. A 55h is placed in the output buffer if no errors are detected.

continued

Function 10h — Write Command(s) to Keyboard Controller, Continued

System-to-8042 commands, cont'd

Command	Description
ABh	<p>KEYBOARD INTERFACE TEST. This command instructs the 8042 to test the keyboard clock and data lines. The result of the test is placed in the output buffer as follows:</p> <ul style="list-style-type: none"> 00 — No error detected. 01 — Keyboard device clock line is stuck low. 02 — Keyboard device clock line is stuck high. 03 — Keyboard device data line is stuck low. 04 — Keyboard device data line is stuck high.
ACh	Reserved
ADh	<p>DISABLE KEYBOARD INTERFACE. This command instructs the 8042 to set bit 4 of the command byte. This disables the keyboard by driving the keyboard clock low.</p>
A Eh	<p>ENABLE KEYBOARD INTERFACE. This command instructs the 8042 to clear bit 4 of the command byte. This enables the keyboard by driving the keyboard clock line high.</p>
C0h	<p>READ INPUT PORT. This command instructs the 8042 to read the 8042 input port (port 1) and place the data in the 8042 output buffer.</p>
C1h	<p>POLL INPUT PORT LOW. This command instructs the 8042 to continuously read input port 1 bits 0-3 into the status register, bits 4-7, until IBF goes high, when the next command is executed.</p>
C2h	<p>POLL INPUT PORT HIGH. This command instructs the 8042 to continuously read input port 1 bits 4-7 into the status register, bits 4-7, until IBF goes high, when the next command is executed.</p>
D0h	<p>READ OUTPUT PORT. This command instructs the 8042 to read the 8042 output port (port 2) and place the data in the 8042 output buffer.</p>
D1h	<p>WRITE OUTPUT PORT. This command instructs the 8042 that the next byte of data received should be sent to the 8042 output port.</p>
D2h	<p>WRITE KEYBOARD OUTPUT BUFFER. This command instructs the 8042 that the next byte of data received should be sent to the output buffer. The controller will generate an interrupt to the system if the interrupt bit is enabled in the command byte.</p>
D3h	<p>WRITE AUXILIARY DEVICE OUTPUT BUFFER. This command instructs the 8042 that the next byte of data received should be sent to the output buffer. The 8042 will generate an interrupt to the system if the interrupt bit is enabled in the command byte.</p>

continued

Function 10h – Write Command(s) to Keyboard Controller,

Continued

System-to-8042 commands, cont'd

Command	Description
D4h	WRITE TO AUXILIARY DEVICE. This command instructs the 8042 that the next byte of data received should be transmitted to the auxiliary device.
E0h	READ TEST INPUTS. This command instructs the 8042 to read the 8042 T0 and T1 inputs and place them in the output buffer. Bit 0 represents T0 and bit 1 represents T1.
F0-FF	PULSE OUTPUT PORT. This command instructs the 8042 to pulse bits 0 through 3 of the output port (port 2) low for 6 microseconds. Bits 0 through 3 of the command indicate the bits to be pulsed. If the bit is a 0 then the corresponding bit on port 2 should be pulsed. If the bit is a 1 then the corresponding bit should not be pulsed.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0010h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	DWord			Time to wait before continuing request (microseconds)
14h	Word	Reserved (Initialize to 0000h)		
16h	DWord	Pointer to the data area		
1Ch	Byte	Data String Length Note: No action occurs if this field is 0.		
28h	Word	Reserved (Initialize to 0000h)		

continued

Function 10h — Write Command(s) to Keyboard Controller, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
0005h	Not My Interrupt, Resume Stage after Interrupt
8000h	Device Busy, Request Refused
8003h	Security Enabled, Keyboard Inhibited — Request Refused
9000h	Keyboard Controller Perpetually Busy
9002h	Resend Error
9100h	Keyboard Controller Perpetually Busy
9101h	Keyboard Failed Reset
9102h	Resend Error
B001h	Keyboard Error
B101h	Keyboard Error The operating system should invoke function 05h Reset/Initialize Keyboard after a keyboard hardware error.
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Keyboard Parameter
FFFFh	Return Code Is Not Valid

Function 11h — Write Command(s) and Data to Keyboard

Description

This function generates a discrete multistaged request that writes a “list” of a command and data from the system to the keyboard hardware itself. The command/data list may consist of any combination of a command and data, however, only one command can be sent per request. The location and length of the command/data list must be specified at function startup. BIOS delays between writing each byte.

Receipt of ACK

BIOS expects an Acknowledge from the 8042 keyboard controller at I/O port 0060h after each byte is received, but does not send the ACK byte to the caller.

Changes to 8042

The caller must not change the state of the 8042 in intermediate stages of a multistaged request for this function.

BIOS keyboard command processing

The first byte of the string passed to BIOS is assumed to be the command byte and is sent to I/O port 0060h. All subsequent bytes are also sent to I/O port 0060h. BIOS issues a Return Code of 0001h, Resume Stage after Interrupt, between each byte sent. From the I/O ports, each byte is sent to the keyboard controller. At this point, BIOS does not respond to the keyboard controller, since it reads the 8042 status register which indicates when commands have been accepted. Return Code 0000h is issued when all bytes have been read.

Some commands from the system to the 8042 controller require that a data byte also be written to I/O port 64h. Function 11h does not support such commands.

continued

Function 11h — Write Command(s) and Data to Keyboard, Continued

Table of system-to-keyboard commands

Command	Description
EDh	SET/RESET LED TOGGLE STATUS INDICATORS. The Num Lock, Caps Lock, and Scroll Lock LED indicators can be turned on or off by a command from the system. The EDh command byte is written to port 0060h and the keyboard responds with FAh (ACK). The system then writes the option byte to port 0060h. A value of 1 means turn the LED on. The option byte takes the format below: Bit 7-3 = Reserved (must be 00000b) Bit 2 = Caps Lock LED Bit 1 = Num Lock LED Bit 0 = Scroll Lock LED
EEh	ECHO. The system uses this command to test the keyboard. The keyboard issues an EEh (Echo) in reply to this command.
EFh	INVALID COMMAND. The keyboard does not acknowledge this command.
F0h	CHOOSE ALTERNATE SCAN CODE SET. The 8042 APCF supports scan code set 2 only.
F1h	INVALID COMMAND. The keyboard does not acknowledge this command.
F2h	READ KEYBOARD ID BYTES. The keyboard acknowledges the command and sends the two keyboard ID bytes.
F3h	SET TYPEMATIC REPEAT RATE AND DELAY PERIOD. The system may set typematic rate and delay.
F4h	ENABLE. Commands the keyboard to clear its output buffer and begin scanning.
F5h	DEFAULT DISABLE. Resets all conditions within the keyboard to their power-on default state and disables scanning. The keyboard responds with ACK, clears its output buffers, and waits for the next instruction from the system.
F6h	SET DEFAULT Resets all conditions within the keyboard to their power-on default state and enables scanning. The keyboard responds with ACK and clears all output buffers.
F7h-FDh	RESERVED
FEh	RESEND. If the system detects an error in transmission, it issues the resend command. The keyboard responds by sending the previous output.
FFh	RESET. System issues this command to invoke the keyboard's internal self-test.

continued

Function 11h – Write Command(s) and Data to Keyboard, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0011h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	DWord			Time to Wait Before Continuing Request (microseconds)
14h	Word	Reserved (Initialize to 0000h)		
16h	DWord	Logical Pointer to the Data Area		
1Ch	Byte	Data String Length Note: No action occurs if this field is 0.		
28h	Word	Reserved (Initialize to 0000h)		

continued

Function 11h — Write Command(s)/Data to Keyboard, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
0005h	Not My Interrupt, Resume Stage after Interrupt
8000h	Device Busy, Request Refused
8003h	Security Enabled, Keyboard Inhibited — Request Refused
9000h	Keyboard Controller Perpetually Busy
9002h	Resend Error
9100h	Keyboard Controller Perpetually Busy
9102h	Resend Error
B001h	Keyboard Error
B101h	Keyboard Error The operating system should invoke function 05h Reset/Initialize Keyboard after a keyboard hardware error.
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Keyboard Parameter
FFFFh	Return Code is Not Valid

Chapter 11

ABIOS Video Service

Overview

Description

The ABIOS Video Service provides I/O support for IBM PS/2-compatible video hardware, specifically, a video graphics array (VGA) adapter. VGA video is built into the motherboard of most PS/2 and compatible computers.

The ABIOS Video Service provides I/O support for both color and monochrome analog video monitors.

ABIOS Video Service provides a significant advantage in that most routines can be called by user programs regardless of the video mode being used (monochrome, CGA, EGA, or VGA). The ABIOS Video Service makes available all the features and functions of the VGA BIOS. The Video Services determine the current display type and perform the necessary address translation.

continued

Overview, Continued

Summary of Video Service functions

Function	Description
00h	Default Interrupt Handler
01h	Return Logical ID Parameters
02h	Reserved
03h	Read Device Parameters
04h	Reserved
05h	Set Video Mode
06h-0Ah	Reserved
0Bh	Return ROM Fonts Information
0Ch	Save Video Environment
0Dh	Restore Video Environment
0Eh	Select Character Generator Block
0Fh	Load Text Mode Font
10h	Enhanced Load Text Mode Font
11h	Read Palette Register
12h	Write Palette Register
13h	Read DAC Color Register
14h	Write DAC Color Register
15h	Read Block of Color Registers
16h	Write Block of Color Registers

In this chapter

This chapter includes information about the following topics:

- Hardware Environment
 - Video Modes
 - Mode/Monitor Support
 - ROM-Resident Fonts
 - Error Handling
 - Video Service Functions
-

Hardware Environment

Introduction

The BIOS Video Service supports IBM VGA-compatible hardware, including:

- a VGA-compatible chip or chip set that includes a:
 - CRT Controller
 - sequencer
 - graphics controller, and an
 - attribute controller
 - DAC chip — INMOS G171 or compatible DAC (digital-to-analog converter)
 - video RAM — 256K of dynamic read/write RAM configured as four 64K maps
 - monochrome or color direct drive analog monitor
 - monochrome or color multiple sync frequency monitors
-

VGA-compatible chip (or chip set)

The VGA chip (or chip set) provides all CRT control signals. It consists of four components, summarized in the following table:

Component	Function
CRT Controller	Generates horizontal and vertical CRT sync timings, cursor and underline timings, video buffer addressing, and refresh addressing.
Sequencer	Arbitrates system access to display RAM and fonts. The sequencer allows up to eight fonts with two fonts displayable at any one time.
Graphics Controller	Handles read/write operation on four parallel bit planes. Outputs data to Attribute Controller.
Attribute Controller	Converts incoming text mode attribute data or graphics mode pixel data into 8-bit indices into the Digital-to-Analog Converter (DAC) color registers (see below).

For a complete description of VGA-compatible components, I/O ports, and registers, refer to the hardware documentation accompanying your particular VGA-compatible chip or chip set.

continued

Hardware Environment, Continued

Digital-to-Analog Converter (DAC)

The video DAC contains 256 individual color registers which can be accessed by the BIOS as either four 64-color registers or sixteen 16-color registers.

Each DAC color register contains one 18-bit RGB analog value. Six bits of each register are allocated to each primary color. Thus, the color represented in each DAC color register may be any of 256K possible colors (i.e. $2^3 \times 6 = 256K$).

Video RAM

The BIOS video service requires at least 256K of read/write video RAM formatted into four banks (or maps) of 64K.

To maintain compatibility, display memory for each of the MDA, CGA, and EGA-compatible modes is mapped exactly as it was in the original display adapter. The display memory organization for the new VGA modes is outlined in this chapter under the Video Modes heading.

Analog monitor support

To display all modes, the Video Service requires either a monochrome or a color direct drive analog monitor with a 31.5 KHz horizontal scan frequency.

The display's vertical gain is adjusted automatically by the VGA-compatible circuitry. Thus, video modes with 350, 400, and 480 horizontal scan lines can be displayed without requiring manual adjustment.

Multiscan monitor support

In addition to 31.5 KHz direct drive analog monitors, the Video Service also supports multiscan rate monitors capable of operating in analog modes (e.g. NEC Multisync monitor). Monitors of this type require an adapter cable that matches the signal assignments and monitor ID circuitry of the DAC external video controller.

Video Modes

Introduction

The BIOS video service supports 17 video modes, providing backward compatibility with MDA (Monochrome Display Adapter), CGA (Color Graphics Adapter), and EGA (Enhanced Graphics Adapter) modes — as well as compatibility with all new VGA modes.

Table of video modes

Mode	Emul.	Res.	Type	Max. Colors	Scheme	Char. Box	Max. Pgs.	Buff. Start
0, 1	CGA*	320x200	Text	16/256K	40x25	8x8	8	B8000h
0, 1	EGA*	320x350	Text	16/256K	40x25	8x14	8	B8000h
0, 1	VGA+	360x400	Text	16/256K	40x25	9x16	8	B8000h
2, 3	CGA*	640x200	Text	16/256K	80x25	8x8	8	B8000h
2, 3	EGA*	640x350	Text	16/256K	80x25	8x14	8	B8000h
2, 3!	VGA+	720x400	Text	16/256K	80x25	9x16	8	B8000h
4, 5	CGA	320x200	Graphics	4/256K	40x25	8x8	1	B8000h
6	CGA	640x200	Graphics	2/256K	80x25	8x8	1	B8000h
7	MDA*	720x350	Text	MDA Mono	80x25	9x14	8	B0000h
7!	VGA*	720x400	Text	VGA Mono	80x25	9x16	8	B0000h
D	EGA	320x200	Graphics	16/256K	40x25	8x8	8	A0000h
E	EGA	640x200	Graphics	16/256K	80x25	8x8	4	A0000h
F	EGA	640x350	Graphics	Mono	80x25	8x14	2	A0000h
10	EGA	640x350	Graphics	16/256K	80x25	8x14	2	A0000h
11	VGA	640x480	Graphics	2/256K	80x30	8x16	1	A0000h
12	VGA	640x480	Graphics	16/256K	80x30	8x16	1	A0000h
13	VGA	320x200	Graphics	256/256K	40x25	8x8	1	A0000h
<p>“!” Indicates power-on default mode 3! = color monitor is attached, 7! = monochrome monitor is attached.</p> <p>“*” Indicates that scan lines must be specified before mode set. (See AH = 12h BL = 30h Select Scan Line for Alphanumeric Codes for details.)</p> <p>“+” Indicates default mode</p>								

continued

Video mode facts

- **All modes can be displayed on color or monochrome**
All video modes can be displayed on either color or monochrome monitors.
 - **MDA, CGA, and EGA modes are emulated**
To insure compatibility with older software, the BIOS Video Service emulates all MDA, CGA, and EGA modes.
 - **Modes 00h, 02h, 04h = Modes 01h, 03h, 05h**
On the CGA adapter, modes 00h, 02h, and 04h have color burst turned off, and modes 01h, 03h, and 05h have color burst turned on. The BIOS video service does not support color burst; modes 00h, 02h, and 04h are identical to modes 01h, 03h, and 05h, respectively.
 - **Text mode resolution determines default font/text scheme**
In text modes, the number of scan lines to display must be specified via function 05h Set Video Mode. Setting text mode scan lines determines both the default ROM-resident font BIOS will load and the column-by-row text scheme in which the font will be displayed.
 - **Text mode default font and text scheme can be overridden**
BIOS Video Service function 0Fh Load Text Mode Font allows the caller to override the default font loaded when a text mode is requested. Function 10h Enhanced Load Text Mode Font allows the caller to override both the default font and column-by-row text scheme associated with a given text mode.
 - **Color modes are programmable**
Unless specified otherwise, the default colors associated with a given video mode are programmed by BIOS at mode set. BIOS Video Service functions 11h-16h can be used to override the default colors associated with any given color mode. See the Programming Colors section in this chapter for more information.
 - **200 scan-line modes are double-scanned**
Each line of video is painted on the screen twice, one beneath the other, before the next new scan line is painted.
 - **No cursor in graphics modes**
A cursor is not displayed in graphics modes.
-

Mode/Monitor Support

Introduction

Both monochrome and color 31.5 KHz direct drive analog monitors, as well as multiscan monitors, can display all of the VGA video modes.

When a monochrome monitor is attached, colors are displayed as shades of gray. When a color monitor is attached, gray scale summing must be explicitly enabled or disabled via function 05h Set Video Mode.

Some mode/monitor facts

If...	Then...
a monochrome analog monitor is attached...	colors are displayed as shades of gray, with the maximum number of shades equal to the maximum number of colors displayable in the respective mode. In mode 0Eh, for example, 16 shades of gray can be displayed. Mode 13h is an exception to this rule. Only 64 (and not 256) shades of gray can be displayed in mode 13h.
a color analog monitor is attached.....	the colors displayed are selected from a BIOS initialized palette of 64 color registers. The number of colors displayed in each mode is listed under the Video Mode heading in this chapter. In mode 0Eh, for example, 16 different colors can be displayed. The BIOS initializes the 64 colors in this palette to analog equivalents of the 64-color digital EGA palette. The remaining 192 color registers are undefined. Mode 13h is an exception to this rule. Mode 13h is capable of displaying all 256 colors stored in the DAC. The BIOS initializes these colors to IBM compatible defaults at mode set.
a 200 scan line mode is selected....	all 200 scan line modes are double-scanned. That is, each horizontal line is scanned twice, for a total of 400 scan lines. This is true for both color and monochrome analog monitors.
power-on default video modes are in effect	the default power on mode is mode 3+ (720x400, 16-color) when a color analog monitor is attached, or mode 7+ (720x400, monochrome) is the default power-on mode when a monochrome analog monitor is attached.

continued

Palette register/DAC relationship

The Attribute Controller component of the VGA hardware contains an internal Palette Register. The Palette Register is composed of sixteen 8-bit registers. The value contained in each palette register is combined with other Attribute Controller information to create an 8-bit index into the DAC.

The video DAC contains 256 individual color registers. Each DAC color register contains one 18-bit RGB analog value. Six bits of each register are allocated to each primary color. Thus, the color represented in each DAC color register may be any of 256K possible colors (i.e. $2^3 \times 6 = 256K$).

BIOS initializes the Palette Registers and the DAC Color Registers upon each mode set:

Type	Description
EGA and 16-color VGA	<p>DAC The BIOS initializes access to the 256 DAC color registers in blocks, or pages, of 64 colors each. The first 64 color page is initialized to analog equivalents of the 64 EGA colors. The remaining 192 colors remain undefined.</p> <p>Attribute Controller To maintain color compatibility with existing EGA software, the BIOS initializes the Attribute Controller to the 16 commonly accepted default EGA colors.</p>
CGA	<p>DAC The BIOS initializes access to the 256 DAC color registers as four blocks of 64 colors each in CGA modes. The first 64 color block is initialized as four 16 color subblocks. Each of these 16 color subblocks is initialized to the 16 colors displayable in CGA mode. The remaining 192 DAC registers are undefined.</p> <p>Attribute Controller To maintain color compatibility with existing CGA software, the BIOS initializes the Attribute Controller to the 16 color CGA palette. When applications software requests a CGA color, the Attribute Controller maps the requests through to its analog equivalent in the DAC controller.</p>
13h	<p>DAC The BIOS initializes all 256 color registers to their IBM PS/2 default equivalents.</p> <p>Attribute Controller The attribute controller receives 8-bit color values directly from application programs. It maps those values directly into the DAC color registers.</p>

continued

Mode/Monitor Support, Continued

Overriding default color values

ABIOS Video Service functions 11h–16h provide ways of overriding default Palette Register values. For more information, see the descriptions of these functions later in this chapter.

ROM-Resident Fonts

ROM-resident fonts

The BIOS contains three ROM BIOS-resident character generators (fonts):

- 8x8 dot
- 8x14 dot
- 8x16 dot

The BIOS is also capable of generating a 9x16 dot font in 400 line text modes. The BIOS generates this 9x16 font by running the 8x16 character set through a ROM-resident table of corrections.

continued

ROM-Resident Fonts, Continued

Default fonts are loaded during mode set

The default font associated with a given video mode is loaded by function 05h Set Video Mode. Which default font is loaded is directly related to the resolution produced by the mode requested.

If the resolution is....	Then the default font is...
320x200	8x8
320x350	8x14
360x400	9x16
640x200	8x8
640x350	8x14
640x480	8x16
720x350	9x14
720x400	9x16

Text modes and character blocks

In text modes, the ASCII character number is written to video memory map 0; the corresponding attribute byte is written to memory map 1. Memory map 2 is used for storing font data.

Each font occupies one 8K block, making for a total of 8 possible fonts. Character block numbers are zero-based. Character block 0 contains the first 8K font; character block 1 contains the second 8K character block; and so on through character block 7.

The maximum number of characters associated with any one font is 100h (i.e. 256 characters). The video hardware is capable of displaying characters selected from up to two character blocks (i.e. up to 512 characters).

Overriding default fonts

ABIOS Video Service functions 0Fh Load Text Mode Font and function 10h Enhanced Load Text Mode Font provide a way to load nondefault or user-defined fonts into character blocks for display in a given text mode.

Error Handling

How errors are reported

ABIOS signals the status (Successful, Resume Stage after Interrupt, etc.) of each ABIOS request by returning a one word Return Code at offset 0Ch in the Request Block.

If Bit 15 of the Return Code field is set, the Video Service function requested has an error. The caller's Return Code handler routine should then test Bits 14, 13, 12, and 8 to determine the class of error that has occurred. The return code handler routine should then test the remaining bits to determine the precise nature of the error.

Function: 00h — Default Interrupt Handler

Description

This single-staged function handles unexpected hardware interrupts by resetting the interrupting condition at the device level.

In the BIOS Video Service, function 00h resets the appropriate video hardware registers.

When to invoke

This function is only invoked if a given Logical ID has no outstanding Request Blocks waiting for an interrupt.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0000h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper BIOS performance.

Code	Description
0000h	Successful Operation
0005h	Not My Interrupt, Resume Stage after Interrupt
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 01h — Return Logical ID Parameters

Description

This function is a single-staged request that returns the parameters for the specified Logical ID.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length (20h)		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0001h)		
08h	Word	Reserved (initialize to 0000h)		
0Ah	Word	Reserved (initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	Byte			Hardware Interrupt Level (06h)
11h	Byte			Arbitration Level (09h)
12h	Word		Device ID (0003h)	
14h	Word		Count of Units	
16h	Word		Logical ID flags Bits 15-4 = Reserved Bit 3 = 0 No overlap across units = 1 Overlap across units supported Bit 2 = 0 Reserved Bits 1-0 = Transfer Data Pointer Mode 00 = No Pointers Required	
18h	Word		Request Block Length (for other functions)	
1Ah	Byte		Reserved (initialize to 0000h)	Secondary Device ID
1Bh	Byte			Revision Level
1Ch	Word		Reserved (initialize to 0000h)	
1Eh	Word	Reserved (initialize to 0000h)		

continued

Function: 01h — Return Logical ID Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper BIOS performance.

Code	Description
0000h	Successful Operation
0005h	Not My Interrupt, Resume Stage after Interrupt
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 02h — Reserved

Function: 03h — Read Device Parameters

Description

This single-staged function returns parameter data concerning the current video state. It also returns buffer size information that must be referenced before invoking Function 0Bh Return ROM Fonts Information and Function 0Ch Save Video Environment.

Video state parameters

The video parameters returned by function 03h include:

- current video mode,
 - number of scan lines associated with mode,
 - monitor type (color or monochrome),
 - character height, and
 - character blocks specifiers.
-

Character block specifier field

When the current video mode is a text mode, the Character Block Specifier field at offset 24h in the Request Block identifies which two character blocks are being used.

Bits 11–8 hold the binary value (000b–111b) of character block A, which contains the first displayable character set. Bits 3–0 hold the binary value of character block B, which contains the second displayable character set. The remaining bits in this field are reserved.

continued

Function: 03h — Read Device Parameters, Continued

Interpreting the character block specifier field

The values returned in the function 03h Request Block Character Block Specifier field are interpreted as described in the table below:

Selector Field Value	Meaning
When the value of character block specifier A is different from character block specifier B	<ul style="list-style-type: none">▪ Two character sets are available for display.▪ Bit 3 of the attribute byte identifies which character block is to be displayed:<ul style="list-style-type: none">1 = Use character block selector A0 = Use character block selector B
When the value in character block specifier A is equal to the value in character block specifier B	<ul style="list-style-type: none">▪ One character set is available for display.▪ Bit 3 of the attribute byte determines the foreground intensity of the character displayed:<ul style="list-style-type: none">1 = Foreground Intensity on0 = Foreground Intensity off

If the mode returned is a graphics mode, the value returned in the Character Block Specifier field is undefined.

continued

Function: 03h — Read Device Parameters, Continued

Buffer size parameters

Function 03h also returns buffer size information applicable to the two Video Service functions listed below:

- **Buffer size: Function 0Bh Return ROM Fonts Information**

Function 0Bh Return ROM Fonts Information outputs information concerning each of the ROM fonts to a buffer specified on input. The value returned by function 03h at Request Block offset 2Ah provides the size of the buffer required by function 0Bh.

Function 03h must always be invoked before function 0Bh can be executed successfully. For more information on Function 0Bh, see the description of this function later in this chapter.

- **Buffer size: Function 0Ch Save Video Environment**

Function 0Ch Save Video Environment outputs the requested aspects of the current video state to a buffer specified on input. The values returned by function 03h at Request Block offsets 2Eh, 30h, 32h, and 34h are referenced by function 0Ch when determining the buffer size it will require.

Function 03h must always be invoked before function 0Ch can be executed successfully. For more information on Function 0Ch, see the description of this function later in this chapter.

continued

Function: 03h — Read Device Parameters, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0003h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
1Ch	Byte			Scan lines per screen 00h = 200 lines 01h = 350 lines 02h = 400 lines 03h = 480 lines 04h-0FFh = Reserved
1Eh	Word			Video Mode
20h	Word		Type of monitor attached Bits 15-1 = Reserved Bit 0 = Color or monochrome 0 = Color 1 = Monochrome	
22h	Word		Character height (bytes/character)	
24h	Word		Character block specifier (Modes 0, 1, 2, 3, or 7 only) Bits 15-12 = Reserved Bits 11-8 = Character block select A Bits 7-4 = Reserved Bits 3-0 = Character block select B	
28h	Word		Reserved (Initialize to 0000h)	
2Ah	Word			Size of data buffer required for function 0Bh Return ROM Fonts Information
2Eh	Word			Size of save/restore buffer header (bytes)
30h	Word			Size of save/restore hardware states (bytes)
32h	Word	Size of save/restore Device Block state (bytes)		
34h	Word	Size of save/restore Digital-to-Analog Converter state (bytes)		

continued

Function: 03h — Read Device Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper BIOS performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Video Parameter
FFFFh	Return Code Field Not Valid

Function: 04h — Reserved

Function: 05h — Set Video Mode

Description

This single-staged function initializes the video hardware to the requested mode.

Table of video modes

Mode	Emul.	Res.	Type	Max. Colors	Scheme	Char. Box	Max. Pgs.	Buff. Start
0, 1	CGA*	320x200	Text	16/256K	40x25	8x8	8	B8000h
0, 1	EGA*	320x350	Text	16/256K	40x25	8x14	8	B8000h
0, 1	VGA+	360x400	Text	16/256K	40x25	9x16	8	B8000h
2, 3	CGA*	640x200	Text	16/256K	80x25	8x8	8	B8000h
2, 3	EGA*	640x350	Text	16/256K	80x25	8x14	8	B8000h
2, 3!	VGA+	720x400	Text	16/256K	80x25	9x16	8	B8000h
4, 5	CGA	320x200	Graphics	4/256K	40x25	8x8	1	B8000h
6	CGA	640x200	Graphics	2/256K	80x25	8x8	1	B8000h
7	MDA*	720x350	Text	MDA Mono	80x25	9x14	8	B0000h
7!	VGA*	720x400	Text	VGA Mono	80x25	9x16	8	B0000h
D	EGA	320x200	Graphics	16/256K	40x25	8x8	8	A0000h
E	EGA	640x200	Graphics	16/256K	80x25	8x8	4	A0000h
F	EGA	640x350	Graphics	Mono	80x25	8x14	2	A0000h
10	EGA	640x350	Graphics	16/256K	80x25	8x14	2	A0000h
11	VGA	640x480	Graphics	2/256K	80x30	8x16	1	A0000h
12	VGA	640x480	Graphics	16/256K	80x30	8x16	1	A0000h
13	VGA	320x200	Graphics	256/256K	40x25	8x8	1	A0000h
<p>"!" Indicates power-on default mode 3! = color monitor is attached, 7! = monochrome monitor is attached.</p> <p>"**" Indicates that scan lines must be specified before mode set. (See AH = 12h BL = 30h Select Scan Line for Alphanumeric Codes for details.)</p> <p>"+" Indicates default mode</p>								

continued

Function: 05h — Set Video Mode, Continued

Video mode facts

- **All modes can be displayed in color and monochrome**

All 17 video modes can be displayed on either color or monochrome monitors.

- **MDA, CGA, and EGA modes are emulated**

To insure compatibility with older software, the BIOS Video Service emulates all MDA, CGA, and EGA modes.

- **Modes 00h, 02h, 04h = Modes 01h, 03h, 05h**

On the CGA adapter, modes 00h, 02h, and 04h have color burst turned off, and modes 01h, 03h, and 05h have color burst turned on. The BIOS video service does not support color burst; modes 00h, 02h, and 04h are identical to modes 01h, 03h, and 05h, respectively.

- **Text mode resolution determines default font/text scheme**

In text modes, the number of scan lines to display must be specified via function 05h Set Video Mode. Setting text mode scan lines determines both the default ROM-resident font BIOS will load and the column-by-row text scheme in which the font will be displayed.

- **Text mode default font and text scheme can be overridden**

BIOS Video Service function 0Fh Load Text Mode Font allows the caller to override the default font loaded when a text mode is requested. Function 10h Enhanced Load Text Mode Font allows the caller to override the default font and the column-by-row text scheme associated with a text mode.

- **Color modes are programmable**

The default colors associated with a given video mode are programmed by BIOS at mode set. BIOS Video Service functions 11h–16h can be used to override the default colors associated with any color mode. See the Programming Colors section in this chapter for more information.

- **200 scan-line modes are double-scanned**

Each line of video is painted on the screen twice, one beneath the other, before the next new scan line is painted.

continued

Function: 05h — Set Video Mode, Continued

Video mode facts, cont'd

- **No dots or characters written to screen by BIOS**

BIOS does not actually write dots or characters to video RAM so that they appear on the screen. The caller and/or other programs must do this.

- **No cursor in graphics modes**

A cursor is not displayed in graphics modes.

Function 05h input field considerations

The Video Mode field (offset 1Eh) must be defined to a valid video mode number on entry into function 05h. The entry values of the remaining input fields depends on the monitor attached and the mode being requested

Device Control Flag field (offset 1Ah)

Located at offset 1Ah in the function 05h Request Block, the Device Control Flag contains three flags. The Device Control Flag field must always be defined upon entry into function 05h.

- **Bit 2 — Gray scale summing flag**

When a monochrome monitor is attached, gray scale summing is handled automatically by the video hardware. When a color monitor is attached, gray scale summing is enabled or disabled by setting this bit.

- **Bit 1 — Initialize DAC to default values flag**

The video DAC can be initialized to IBM-compatible default values upon each mode set. Bit 1 enables (on) or disables (off) this feature of the BIOS Video Service.

- **Bit 0 — Clear video buffer flag**

Video buffer memory can be cleared upon mode set. This clears the video screen. Bit 0 enables (on) or disables (off) this feature.

continued

Function: 05h — Set Video Mode, Continued

Text mode scan lines (offset 1Ch)

The number of scan lines must be defined each time a text mode (modes 00h, 01h, 02h, 03h, 07h) is requested. Besides indicating the number of scan lines to display, the value in this field determines which ROM-resident font the requested text mode will use.

The PS/2 BIOS contains three ROM BIOS-resident fonts:

- 8x8 dot, used in 200 scan-line text and graphics modes
- 8x14 dot, used in 350 scan-line text and graphics modes
- 8x16 dot, used in 480 scan-line graphics modes

When 400 scan lines are indicated in the Text Mode Scan Lines field, the BIOS generates a 9x16 dot font. Because there is no ROM-resident 9x16 font, the BIOS generates the 9x16 font by running the 8x16 font through a ROM-resident table of corrections.

Note: This field is not defined for graphics modes.

Character block to load (offset 26h)

In text modes, the ASCII character number is written to video memory map 0; the corresponding attribute byte is written to memory map 1. Memory map 2 is divided into eight 8K character blocks. Each character block may contain one text mode character font. Character block numbers are zero-based.

Bits 0–7 of the Character Block to Load field define which character blocks will be loaded with the *default* font associated with the text mode being requested. Setting Bit 0 to 1 instructs A BIOS to load the default font into Character Block 0. Setting Bit 0 to 0 tells A BIOS to preserve the present contents of Character Block 0. Setting Bit 1 to 1 instructs A BIOS to load the default font into Character Block 1, and so on.

If all eight bits of the Character Block to Load field are set to zero, then A BIOS does not load the *default* font associated with the text mode requested. Setting all bits to one causes A BIOS to load the default font into each Character Block.

continued

Function: 05h — Set Video Mode, Continued

Overriding default fonts

BIOS Video Service functions 0Fh Load Text Mode Font and function 10h Enhanced Load Text Mode Font provide a way to load nondefault or user-defined fonts into character blocks for display in a given text mode.

Character Block to Select (offset 24h)

In text modes, the video hardware is capable of displaying characters selected from up to two character blocks. The maximum number of characters associated with any one font is 256; therefore up to 512 characters can be displayed.

When the current video mode is a text mode, the Character Block Specifier field identifies which two character blocks are being used.

Bits 11–8 hold the binary value (000b–111b) of character block A, which contains the first displayable character set. Bits 3–0 hold the binary value of character block B, which contains the second displayable character set. The remaining bits in this field are reserved.

Selector Field Value	Meaning
When the value of character block specifier A is different from character block specifier B	<ul style="list-style-type: none">Two character sets are available for display.Bit 3 of the attribute byte identifies which character block is to be displayed:<ul style="list-style-type: none">1 = Use character block selector A0 = Use character block selector B
When the value in character block specifier A is equal to the value in character block specifier B	<ul style="list-style-type: none">One character set is available for display.Bit 3 of the attribute byte determines the foreground intensity of the character displayed:<ul style="list-style-type: none">1 = Foreground intensity on0 = Foreground intensity off

Function 0Eh Select Character Generator Block can be used to set the Character Block Specifier field independently from function 05h.

continued

Function: 05h — Set Video Mode, Continued

Request block structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0005h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
1Ah	Word	Device Control Flags Bits 15-3 = Reserved Bit 2 = Gray Scale Summing 0 Disabled 1 Enabled Bit 1 = Initialize DAC to default values 0 No 1 Yes Bit 0 = Clear video buffer 0 No 1 Yes		
1Ch	Byte	Text Mode Scan Lines 00h = 200 lines (modes hex 0,1,2,3) 01h = 350 lines (modes hex 0,1,2,3,7) 02h = 400 lines (modes hex 0,1,2,3,7) 03h-FFh = Reserved		
1Eh	Word	Video Mode		
24h	Word	Character Block to Display (modes hex 0,1,2,3,7) Bits 15-12 = Reserved Bits 11-8 = Character Block A Bits 7-4 = Reserved Bits 3-0 = Character block B		
26h	Word	Character block to load with de- fault ROM font (modes hex 0,1,2,3,7) Bit n = Block n flag 0 = Do not load font 1 = Load default font		
28h	Word	Reserved (Initialize to 0000h)		

continued

Function: 05h — Set Video Mode, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper BIOS performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Video Parameter
FFFFh	Return Code Field Not Valid

Functions: 06h – 0Ah — Reserved

Function: 0Bh — Return ROM Fonts Information

Description

This single-staged function outputs information concerning each ROM-resident font to a ROM Font buffer area specified on function entry.

How to determine buffer size

Function 03h Read Device Parameters returns the size of the buffer required by the Return ROM Fonts Information function. Function 03h returns this value to offset 2Ah of the function 03h Request Block.

Before function 0Bh can be issued, function 03h must always be invoked, to determine the proper buffer size.

Buffer format

Each entry into the ROM Font Information buffer occupies 12 bytes. The format of each font entry is defined in the table below:

Size	Description
Word	Reserved
DWord	Pointer to ROM-resident font
Word	Reserved
Byte	Character size: number of columns
Byte	Character size: number of rows
Byte	Total/Partial font, where: 00h = total font 01h = Partial font 02h = FFh = Reserved
Byte	Partial Font If a partial font is indicated in the byte above, this field tells which default font that partial font is drawn from.

continued

Function: 0Bh — Return ROM Fonts Information, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Bh)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Word	Reserved (Initialize to 0000h)		
12h	DWord	Pointer to buffer that stores ROM font information		
16h	Word	Reserved (Initialize to 0000h)		

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper BIOS performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Video Parameter
FFFFh	Return Code Field Not Valid

Function: 0Ch — Save Video Environment

Description

This single-staged function stores any of three video states to a buffer specified upon function entry. The three states are:

- Hardware state
 - Device Block state
 - Digital-to-Analog Converter (DAC) state
-

Buffer size values are returned by function 03h

The values returned by function 03h at Request Block offsets 2Eh, 30h, 32h, and 34h must be referenced in order to determine the buffer size to specify upon entry into function 0Ch.

Offset	Description
2Eh	Size of Save Environment buffer header
30h	Buffer size required to save hardware state
32h	Buffer size required to save Device Block state
34h	Buffer size required to save DAC state

To insure a proper buffer size is calculated, function 03h must always be issued before function 0Ch can be executed successfully.

Calculating video environment buffer size

The caller may elect to save any or all three video states.

The size of the save environment buffer that must be specified on entry into function 0Ch is calculated as follows:

Buffer header size + Buffer size of each state to be saved.

continued

Function: 0Ch — Save Video Environment, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Ch)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Word	Reserved (Initialize to 0000h)		
12h	DWord	Pointer to environment save area		
16h	Word	Reserved (Initialize to 0000h)		
2Ch	Word	Video states to be saved Bits 15-3 = Reserved (set to 0) Bit 2 = DAC state 1 = Save state Bit 1 = Device Block state 1 = Save state Bit 0 = Hardware state 1 = Save state		

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper BIOS performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Video Parameter
FFFFh	Return Code Field Not Valid

Function: 0Dh — Restore Video Environment

Description

This single-staged function restores the video environment located at the buffer area pointed to on function entry.

For more information on the structure and contents of the video environment buffer, see "Function: 0Ch — Save Video Environment."

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Dh)		
08h	Word	Reserved (initialize to 0000h)		
0Ah	Word	Reserved (initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Word	Reserved (initialize to 0000h)		
12h	DWord	Pointer to the video environment to be restored		
16h	Word	Reserved (initialize to 0000h)		
1Ah	Word	Device control flag Bits 15-1 = Reserved Bit 0 = Clear video buffer 0 = No 1 = Yes		
2Ch	Word	Video states to be restored Bits 15-3 = Reserved (set to 0) Bit 2 = DAC state 1 = Restore state Bit 1 = Device Block state 1 = Restore state Bit 0 = Hardware state 1 = Restore state		

continued

Function: 0Dh — Restore Video Environment, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper BIOS performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Video Parameter
FFFFh	Return Code Field Not Valid

Function: 0Eh — Select Character Generator Block

Description

This single-staged function allows the caller, when in text modes, to select either of two character blocks.

The maximum number of characters associated with any one character block is 256; therefore up to 512 characters can be displayed.

When the current video mode is a text mode, the Character Block Specifier field identifies which two character blocks are being used.

Bits 11–8 hold the binary value (000b–111b) of character block A, which contains the first displayable character set. Bits 3–0 hold the binary value of character block B, which contains the second displayable character set. The remaining bits in this field are reserved.

Selector Field Value	Meaning
When the value of character block specifier A is different from character block specifier B	<ul style="list-style-type: none">Two character sets are available for display.Bit 3 of the attribute byte identifies which character block is to be displayed:<ul style="list-style-type: none">1 = Use character block selector A0 = Use character block selector B
When the value in character block specifier A is equal to the value in character block specifier B	<ul style="list-style-type: none">One character set is available for display.Bit 3 of the attribute byte determines the foreground intensity of the character displayed:<ul style="list-style-type: none">1 = Foreground intensity on0 = Foreground intensity off

Function 05h Set Video Mode can also be used to set the Character Block Specifier field.

continued

Function: 0Eh — Select Character Generator Block, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Eh)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
16h	Word	Reserved (Initialize to 0000h)		
24h	Word	Character Block to Display Bits 15-12 = Reserved Bits 11-8 = Character block A Bits 7-4 = Reserved Bits 3-0 = Character block B		

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper BIOS performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Video Parameter
FFFFh	Return Code Field Not Valid

Function: 0Fh — Load Text Mode Font

Description

This single-staged function loads the font indicated on function entry into the specified Character Block in Memory Map 2.

Scan lines per character, number of character rows, buffer length, and cursor size are not recalculated by this function. The font loaded here must occupy the *default* character box size associated with the text mode in effect.

When to use function 0Fh

When in text mode, use function 0Fh to load a nondefault or user-defined font. For best results, insure that the bytes-per-character of the font being loaded is equal to the bytes-per-character of the default font.

Function 0Fh input field considerations

When a ROM-resident font is loaded, the character height is known and the full 100h character set is loaded. Therefore, the caller need not specify the Count of Characters, Character Offset, or Character Height fields on function entry.

When a user-defined font is loaded, all input fields must be specified on function entry.

continued

Function: 0Fh — Load Text Mode Font, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Fh)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Word	Reserved (Initialize to 0000h)		
12h	DWord	Pointer to the user font		
16h	Word	Reserved (Initialize to 0000h)		
18h	Word	Count of characters		
	Byte	Valid values = 01h to 100h		
1Dh	Byte	Font Type to Load 00h = User-defined 01h = 8x8 ROM font 02h = 8x14 ROM font 03h = 8x16 ROM font 04h-FFh = Reserved		
22h	Word	Character Height (bytes/character)		
24h	Word	Character Block to Load 00h-07h = Valid load values 08h-FFFFh = Reserved		
28h	Word	Character Offset		

continued

Function: 0Fh — Load Text Mode Font, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper BIOS performance.

Code	Description
0000h	Successful Operation When a user-defined font is loaded, the Count of Characters field must not be zero. If set to zero, no font is loaded and function 0Fh returns with Return Code 0000h Successful Operation.
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Video Parameter When a user-defined font is loaded, the Count of Characters and the Character offset fields must not exceed the maximum numbers of characters per character set (<i>100h characters</i>). If greater than 100h, function 0Fh returns with a Return Code field of C005h Invalid Video Parameter.
FFFFh	Return Code Field Not Valid

Function: 10h — Enhanced Load Text Mode Font

Description

This single-staged function loads the font indicated on function entry into the specified Character Block in Memory Map 2. The font loaded here need not occupy the *default* character box size associated with the text mode in effect.

Scan lines per character, number of character rows, buffer length, and cursor size are recalculated by this function.

When to use function 10h

Use function 10h to load a font that has a different box size than the default font associated with the current text mode. Cursor size and number of rows on screen will be recalculated automatically.

For example, the default font associated with mode 07h (720x400, VGA text mode) is 9x16. The default text scheme is 80 columns by 25 rows.

Since function 10h recalculates scan lines per character, number of character rows, buffer length, and cursor size, loading the ROM-resident 8x8 font in this mode yields a screen that displays 80 columns by 50 rows, with a properly scaled cursor.

Function 0Fh input field considerations

When a ROM-resident font is loaded, the character height is known and the full 100h character set is loaded. Therefore, the caller need not specify the Count of Characters, Character Offset, or Character Height fields on function entry.

When a user-defined font is loaded, all input fields must be specified on function entry.

continued

Function: 10h — Enhanced Load Text Mode Font, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0010h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Word	Reserved (Initialize to 0000h)		
12h	DWord	Pointer to the user font		
16h	Word	Reserved (Initialize to 0000h)		
18h	Word	Count of characters Valid values = 01h-100h		
1Dh	Byte	Font Type to Load 00h = User-defined 01h = 8x8 ROM font 02h = 8x14 ROM font 03h = 8x16 ROM font 04h-FFh = Reserved		
22h	Word	Character Height (bytes/character)		
24h	Word	Character Block to Load 00h-07h = Valid values 08h-FFFFh = Reserved		
28h	Word	Character offset		

continued

Function: 10h — Enhanced Load Text Mode Font, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper BIOS performance.

Code	Description
0000h	Successful Operation When a user-defined font is loaded, the Count of Characters field must not be zero. If set to zero, no font is loaded and function 0Fh returns with Return Code 0000h Successful Operation.
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Video Parameter When a user-defined font is loaded, the Count of Characters and the Character offset fields must not exceed the maximum numbers of characters per character set (<i>100h characters</i>). If greater than 100h, function 0Fh returns with a Return Code field of C005h Invalid Video Parameter.
FFFFh	Return Code Field Not Valid

Function: 11h — Read Palette Register

Description

This single-staged function reads the value of the palette register specified on function entry.

Attribute controller, palette registers, and the DAC

The Attribute Controller component of the VGA hardware contains an internal Palette Register. The Palette Register is composed of sixteen 8-bit registers. Bits 0–5 can be programmed to any value up to 3Fh (i.e. 64). Bits 7–6 are reserved. The value contained in each palette register is combined with other Attribute Controller information to create an 8-bit index into the DAC.

Palette registers are initialized at mode set

The value programmed into each Palette Register is determined by BIOS at mode set. When a 16-color EGA or VGA mode is selected, BIOS initializes the Palette Register to the 16 default EGA colors. When a 16-color CGA mode is selected, BIOS initializes the Palette Registers to the 16-color CGA palette. In mode 13h, BIOS initializes the entire Attribute Controller so that the 8 bit per pixel value used in this mode maps directly into the DAC.

continued

Function: 11h – Read Palette Register, Continued

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0011h)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		
16h	Word	Reserved (Initialize to 0000h)	
32h	Word	Palette Register to Read 00h-0Fh = Valid values 10h-FFFFh = Reserved	
34h	Word		Palette Value Read

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper BIOS performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Video Parameter
FFFFh	Return Code Field Not Valid

Function: 12h — Write Palette Register

Description

This single-staged function writes a particular value to the palette register specified on function entry. Valid values are 00h–3Fh.

This function is valid for all modes except mode 13h. Writing to a palette register while in mode 13h may cause unpredictable results. In mode 13h, the VGA hardware requires that the Palette Registers remain as programmed on mode set.

Attribute controller, palette registers, and the DAC

The Attribute Controller component of the VGA hardware contains an internal Palette Register. The Palette Register is composed of sixteen 8-bit registers. Bits 0–5 can be programmed to any value up to 3Fh (i.e. 64). Bits 7–6 are reserved. The value contained in each palette register is combined with other Attribute Controller information to create an 8-bit index into the DAC.

Palette registers are initialized at mode set

The value programmed into each Palette Register is determined by BIOS at mode set. When a 16-color EGA or VGA mode is selected, BIOS initializes the Palette Register to the 16 default EGA colors. When a 16-color CGA mode is selected, BIOS initializes the Palette Registers to the 16-color CGA palette. In mode 13h, BIOS initializes the entire Attribute Controller so that the 8-bit per pixel value used in this mode maps directly into the DAC.

Reference

For more information on color palettes, the DAC, and color mode register initialization, refer to Hardware Environment in this chapter.

continued

Function: 12h – Write Palette Register, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0012h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
16h	Word	Reserved (Initialize to 0000h)		
32h	Word	Write value to the palette register 00h-0Fh = Valid write values 10h-FFFFh = Reserved		
34h	Word	Load the palette value 00h-3Fh = Valid 40h-FFFFh = Reserved		

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper BIOS performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Video Parameter
FFFFh	Return Code Field Not Valid

Function: 13h — Read DAC Color Register

Description

This single-staged function returns the 18-bit Red-Green-Blue (RGB) value contained in the DAC color register specified on function entry.

Digital-to-Analog Converter (DAC) color registers

The video DAC contains 256 individual color registers.

Each DAC color register contains one 18-bit RGB analog value. Six bits of each register are allocated to each primary color. Thus, the color represented in each DAC color register may be any of 256K possible colors (i.e. $2^3 \times 6 = 256K$).

How DAC color registers are initialized

The BIOS initializes the 8-bit index values contained in the Attribute Controller and the 18-bit analog color values contained in the DAC color registers each time a video mode is set.

When a CGA, EGA, or 16-color VGA mode is requested, BIOS initializes the first 64 DAC color registers to analog equivalents of the 64 EGA colors. The remaining 192 registers are undefined. When mode 13h (256 color mode) is set, BIOS initializes all 256 color registers to their IBM PS/2-compatible values.

continued

Function: 13h — Read DAC Color Register, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0013h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
16h	Word	Reserved (Initialize to 0000h)		
2Ah	Word	DAC Color Register to Read 00h-FFh = Valid values 100h-FFFFh = Reserved		
2Ch	Word		Read value	
2Eh	Word		Green value	
30h	Word		Blue value	

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper BIOS performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Video Parameter
FFFFh	Return Code Field Not Valid

Function: 14h — Write DAC Color Register

Description

This single-staged function outputs the 18-bit RGB value contained to the DAC color register specified on function entry.

Digital-to-Analog Converter (DAC) color registers

The video DAC contains 256 individual color registers.

Each DAC color register contains one 18-bit RGB analog value. Six bits of each register are allocated to each primary color. Thus, the color represented in each DAC color register may be any of 256K possible colors (i.e. $2^3 \times 6 = 256K$).

How DAC color registers are initialized

The BIOS initializes the 8-bit index values contained in the Attribute Controller and the 18-bit analog color values contained in the DAC color registers each time a video mode is set.

When a CGA, EGA, or 16-color VGA mode is requested, BIOS initializes the first 64 DAC color registers to analog equivalents of the 64 EGA colors. The remaining 192 registers are undefined. When mode 13h (256 color mode) is set, BIOS initializes all 256 color registers to their IBM PS/2-compatible values.

continued

Function: 14h — Write DAC Color Register, Continued

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0014h)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		
16h	Word	Reserved (Initialize to 0000h)	
1Ah	Word	Device control flags Bits 15-3 = Reserved Bit 2 = Gray Scale Summing 0 Disable 1 Enable Bits 1-0 = Reserved	
2Ah	Word	Color Register to Write 00h-FFh = Valid write values 100h-FFFFh = Reserved	Return Code
2Ch	Word	Red Value 00h-3Fh = Valid value 40h-FFFFh = Reserved	
2Eh	Word	Green Value 00h-3Fh = Valid value 40h-FFFFh = Reserved	
30h	Word	Blue Value 00h-3Fh = Valid value 40h-FFFFh = Reserved	

continued

Function: 14h — Write DAC Color Register, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper BIOS performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Video Parameter
FFFFh	Return Code Field Not Valid

Function: 15h — Read Block of Color Registers

Description

This single-staged function reads a block of 18-bit RGB color register values from the Digital-to-Analog-Converter (DAC), beginning with the color register specified on function entry.

A doubleword pointer to the buffer area where the block is to be stored must also be specified on function entry.

Function 16h input field considerations

- **Number of registers to load must be greater than zero**

If the value input into the Number of Registers to Load field is zero, no action is performed and function 16h returns with Return Code 0000h Successful Operation

- **Block length must not exceed 100h**

If the value in the First Color Register to Write field plus the value in the Number of Color Registers to Write field exceeds 100h, then no action is performed and function 16h returns with Return Code C005h Invalid Input Parameter.

Color register data format

The block of DAC color register values returned by function 15h is stored in a system RAM buffer as a series of three-byte sequences as follows:

(Red value, green value, blue value) (red value, green value, blue value)....

The standard range for red, green, or blue values is 00h to 3Fh. The values 40h to FFFFh are reserved.

continued

Function: 15h — Read Block of Color Registers, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0015h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Word	Reserved (Initialize to 0000h)		
12h	DWord	Pointer to read the color registers into the save area		
16h	Word	Reserved (Initialize to 0000h)		
18h	Word	Number of color registers		
2Ah	Word	First color register 00h-FFh = Valid read values 100h-FFFFh = Reserved		

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper BIOS performance.

Code	Description
0000h	Successful Operation If the value input into the Number of Registers to Load field is zero, no action is performed and function 16h issues Return Code 0000h Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Video Parameter If the value in the First Color Register to Write field plus the value in the Number of Color Registers to Write field exceeds 100h, no action is performed and function 16h returns with Return Code C005h Invalid Input Parameter.
FFFFh	Return Code Field Not Valid

Function: 16h — Write Block of DAC Color Registers

Description

This single-staged function loads a block of 18-bit RGB values to the DAC color registers, beginning with the color register specified on function entry.

The start address of the block of color register values to be loaded is indicated with a doubleword pointer that is also specified on function entry.

RGB data format

The block of RGB color register values to be loaded by function 16h must be stored in system RAM as a series of three-byte sequences as follows:

(Red value, green value, blue value) (red value, green value, blue value)....

The standard range for red, green, or blue values is 00h to 3Fh. The values 40h to FFFFh are reserved.

Function 16h input field considerations

■ Device Control Flag and gray scale summing

When a monochrome monitor is attached, gray scale summing is handled automatically by the video hardware. When a color monitor is attached, gray scale summing is enabled or disabled by setting the value of bit 2, Gray Scale Summing, of the Device Control Flag field.

■ Number of register to load must be greater than zero

If the value input into the Number of Registers to Load field is zero, no action is performed and function 16h returns with Return Code 0000h Successful Operation

■ Block length must not exceed 100h

If the value of the First Color Register to Write field plus the value of the Number of Color Registers to Write field exceeds 100h, then no action is performed and function 16h returns with Return Code C005h Invalid Input Parameter.

continued

Function: 16h — Write Block of DAC Color Registers, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0016h)		
08h	Word	Reserved (initialize to 0000h)		
0Ah	Word	Reserved (initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Word	Reserved (Initialize to 0000h)		
12h	DWord	Pointer to write the color registers into the save area		
16h	Word	Reserved (Initialize to 0000h)		
18h	Word	Number of color registers		
1Ah	Word	Device control flags Bits 15-3 = Reserved Bit 2 = Gray Scale Summing 0 Disable 1 Enable Bits 1-0 = Reserved		
2Ah	Word	First color register 00h-FFh = Valid write values 100h-FFFFh = Reserved		

continued

Function: 16h — Write Block of DAC Color Registers, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper BIOS performance.

Code	Description
0000h	Successful Operation If the value input into the Number of Registers to Load field is zero, no action is performed and function 16h issues Return Code 0000h Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Video Parameter If the values in the First Color Register to Write and the Number of Color Registers to Write field exceed 100h, no action is performed and function 16h returns with Return Code C005h Invalid Input Parameter.
FFFFh	Return Code Field Not Valid

Chapter 12

ABIOS Serial Communications Service

Overview

Introduction

The ABIOS Serial Communications Service provides access to serial I/O device adapters. The functions available provide the caller with a means of accessing the system serial ports without directly programming hardware controller registers. To maintain maximum compatibility across different serial port controllers, direct hardware programming of a controller should be avoided.

continued

Overview, Continued

Summary of BIOS Serial Communications Service functions

Function	Description
00h	Default Interrupt Handler
01h	Return Logical ID Parameters
03h	Read Device Parameters
04h	Reserved
05h	Reset/Initialize Serial Port
06h-0Ah	Reserved
0Bh	Set Modem Control
0Ch	Set Line Control
0Dh	Set Baud Rate
0Eh	Transmit
0Fh	Receive
10h	Transmit and Receive
11h	Modem Status
12h	Cancel
13h	Return Status Line
14h	Return Modem Status
15h	Enable FIFO Control

In this chapter

This chapter includes information about the following topics:

- Hardware Environment
 - Error Handling
 - BIOS Serial Communications Service functions
-

Hardware Environment

Serial port

The BIOS supports a National Semiconductor 16550 serial port controller or equivalent logic. The serial ports can be addressed as Serial 1–8. See (40:10h) to find out how many serial ports are available. Serial 1 and 3 interrupts are on IRQ 4; Serial 2 and 4 interrupts are on IRQ 3. The serial port base addresses are shown below. BIOS initializes the serial ports in the same order that they reside in the ROM BIOS data area, so the serial port Logical IDs will be in the same order as in the BIOS Data area (40:10h). Additional serial ports and Logical IDs may be initialized.

Serial port addresses/interrupt levels

Serial Port Number	Base Address	Interrupt Level
1	03F8h	4
2	02F8h	3
3	3220h	3
4	3228h	3
5	4220h	3
6	4228h	3
7	5220h	3
8	5228h	3

NS 16550 characteristics

The NS 16550, which is functionally compatible with the NS 16450 and the NS 8250, supports:

- Characters of 5, 6, 7, or 8 bits,
 - 1, 1.5, or 2 stop bits, and
 - even, odd, or no parity modes.
-

continued

NS 16550 Serial Communications Controller

The NS 16550 does serial to parallel conversions on data received from a peripheral device or a modem, and parallel to serial conversion on data received from the system processor. The system processor can read the status of the NS 16550 at any time during its operation. The information furnished includes the type and condition of transfer operations in progress, as well as any error conditions (parity, overrun, framing, or break interrupt) present. The NS 16550 provides complete modem control, and has a user programmable processor-interrupt system.

Programmable baud rate generator

The serial port controller can operate at speeds of from 110 to 19,200 bps.

NS 16550 Serial Controller Registers

The NS 16550 has 12 accessible registers:

- Receiver Buffer Register (Read Only)
- Transmitter Holding Register (Write Only)
- Interrupt Enable Register (Read/Write)
- Interrupt Identification Register (Read Only)
- FIFO (First in/First out) Control Register (Write Only)
- Line Control Register (Read/Write)
- Modem Control Register (Read/Write)
- Line Status Register (Read Only)
- Modem Status Register (Read Only)
- Scratch Register (Read/Write)
- Divisor Latch (LSB) (Read/Write)
- Divisor Latch (MSB) (Read/Write)

Information on the operation of these registers is contained in the National Semiconductor NS 16550 Data Sheet. However, to avoid any incompatibility problems introduced by direct hardware programming, use the access to the serial controller provided through the BIOS services.

continued

Hardware Environment, Continued

I/O port addresses

The I/O port table in Section 2 provides a complete list of all serial I/O port addresses. These I/O addresses correspond to the serial port registers and provide a standard means of access for the caller.

Error Handling

How errors are reported

ABIOS signals the status (Successful, Resume Stage after Interrupt, etc.) of each ABIOS request by returning a one word Return Code at offset 0Ch in the Request Block.

If Bit 15 of the Return Code field is set, the Serial Communications Service function requested has an error. The caller's Return Code handler routine should test Bits 14, 13, 12, and 8 to determine the class of error that has occurred and then test the remaining bits to determine the precise nature of the error.

Function: 00h — Default Interrupt Handler

Description

This function is a single-staged or multistaged request that handles unexpected hardware interrupts by resetting the interrupt at the device level.

When to invoke

This function is invoked by calling the interrupt routine with a function code of 0000h. It is only invoked if a given Logical ID has no outstanding Request Blocks waiting for an interrupt.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0000h)		
08h	Word	Reserved (initialize to 0000h)		
0Ah	Word	Reserved (initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0005h	Not My Interrupt, Resume Stage after Interrupt
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 01h — Return Logical ID Parameters

Description

This function is a single-staged request that returns the parameters for the specified Logical ID.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length (20h)		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0001h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Byte			Interrupt Level (03h or 04h)
11h	Byte			Arbitration Level (FFh)
12h	Word		Device ID (0006h)	
14h	Word		Count of Units	
16h	Word		Logical ID Flags Bits 15–4 = Reserved Bit 3 = 0 No overlap across units (only one unit is supported). Bit 2 = 0 Reserved Bits 1–0 = Transfer Data Pointer Mode Return Codes 01 = Logical Pointer Required	
18h	Word		Request Block Length (for other functions)	
1Ah	Byte	Reserved (Initialize to 0000h)	Secondary Device ID	
1Bh	Byte		Revision Level	
1Ch	Word	Reserved (Initialize to 0000h)		
1Eh	Word	Reserved (Initialize to 0000h)		

continued

Function: 01h — Return Logical ID Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 03h — Read Device Parameters

Description

This function is a single-staged request that reads the parameters of the serial port(s). This function does not interact with any hardware and has no effect on any other outstanding requests.

Before using this function

Invoke function 05h Reset/Initialize Serial Device to synchronize the serial port parameters before invoking this function.

How the serial port parameters are set up

The serial port parameters are maintained in the Serial Port Device Block (Device ID 0006h). These parameters always reflect the state of the serial port hardware. The parameters are updated every time an ABIOS (or CBIOS) Serial Communications Service function is invoked.

Warning: If the serial ports are directly programmed or programmed using CBIOS INT 14h functions after ABIOS initialization, the serial parameters in the ABIOS Device Block will not be accurate.

Initial serial port values

The serial port(s) is initialized to the following values during ABIOS initialization:

- Baud rate of 1200
 - No parity
 - One stop bit
 - Seven Bits Per Character
 - No break
-

continued

Function: 03h — Read Device Parameters, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0003h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
18h	Word	Reserved (Initialize to 0000h)		
28h	Byte			Modem Control Bits 7-5 = Reserved (set to 0) Bit 4 = 1 Loop Bit 3 = 1 Out2 Bit 2 = 1 Out1 Bit 1 = 1 Request To Send Bit 0 = 1 Data Terminal Ready
29h	Byte		Asynchronous Interrupt Status Bits 7-6 = Reserved (set to 0) Bit 5 = 1 Modem Status Interrupt Bit 4 = Reserved Bit 3 = 1 Transmit Interrupt Bit 2 = 1 Receive Interrupt Bits 1,0 = Reserved	
2Ah	Byte		FIFO Trigger Level	
44h	Byte		Transmission Baud Rate	
			00h = 110 01h = 150 02h = 300 03h = 600 04h = 1200 05h = 2400 06h = 4800 07h = 9600 08h = 19200	

continued

Function: 03h — Read Device Parameters, Continued

Request Block Structure, cont'd

Offset	Size	Input:	Output:
45h	Byte		Parity Type 00h = None 01h = Odd 02h = Even 03h = Stick parity odd 04h = Stick parity even
46h	Byte		Number of Stop Bits Note: The following is true if the number of Bits Per Characters either 1, 2, or 3. 00h = 1 01h = 2 Note: The following is true if the number of Bits Per Character is 0. 00h = 1 01h = 1-1/2
47h	Byte		Number of Bits Per Character 00h = 5 01h = 6 02h = 7 03h = 8
48h	Byte		Break Status 00h = Disabled 01h = Enabled

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 04h — Reserved

Function: 05h — Reset/Initialize Serial Port

Description

This function is a single-staged request that initializes the serial port(s) according to the specified values.

Reset process

When this interrupt is invoked, the BIOS:

- cancels all outstanding serial communications Request Blocks,
- disables all serial port interrupts, including those generated by functions 0Eh Transmit, 0Fh Receive, 10h Transmit and Receive, and 11h Modem Status,
- clears any pending data at the serial ports, and
- synchronizes the Device Block parameters so they match the current hardware serial port values (this must be done before function 03h Read Device Parameters is invoked).

The caller is responsible for deallocating the appropriate Request Blocks and ensuring the appropriate condition of the interrupt controller (by sending EOIs).

continued

Function: 05h — Reset/Initialize Serial Port, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0005h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
18h	Word	Reserved (Initialize to 0000h)		
28h	Byte	Modem Control Bits 7-5 = Reserved Bit 4 = 1 Loop Bit 3 = 1 Out2 Note: This bit must be 0 if Bit 4 is 0. Bit 2 = 1 Out1 Note: This bit must be 0 if Bit 4 is 0. Bit 1 = 1 Send Bit 0 = 1 Data Terminal Ready		
29h	Byte	FIFO Trigger Level, where: 00h = 1 Byte 01h = 4 Bytes 02h = 8 Bytes 03h = 14 Bytes		
2Ah	Byte	FIFO Control, where: 00h = Do not change or FIFO not supported 01h = Enable and Reset FIFO 02h-FFh=Enable FIFO		
44h	Byte	Transmission Baud Rate 00h = 110 01h = 150 02h = 300 03h = 600 04h = 1200 05h = 2400 06h = 4800 07h = 9600 08h = 19200		

continued

Function: 05h — Reset/Initialize Serial Port, Continued

Request Block Structure, cont'd

Offset	Size	Input:	Output:
45h	Byte	Parity Type 00h = None 01h = Odd 02h = Even 03h = Stick parity odd 04h = Stick parity even	
46h	Byte	Number of Stop Bits Note: The following is true for a 6 bit, 7 bit, or 8 bit word length. 00h = 1 01h = 2 Note: The following is true for a 5 bit word length. 00h = 1 01h = 1-1/2	
47h	Byte	Number of Bits Per Character 00h = 5 01h = 6 02h = 7 03h = 8	
48h	Byte	Break Status 00h = Disabled 01h = Enabled	
49h	Byte		
4Ah	Byte		Line Status Bit 7 = Reserved Bit 6 = 1 Transmitter Empty Bit 5 = 1 Transmitter Holding Register Empty Bit 4 = 1 Break Interrupt Bit 3 = 1 Framing Error Bit 2 = 1 Parity Error Bit 1 = 1 Overrun Error Bit 0 = 1 Data Ready Modem Status Bit 7 = 1 Data Carrier Detect Bit 6 = 1 Ring Indicator Bit 5 = 1 Data Set Ready Bit 4 = 1 Clear To Send Bit 3 = 1 Delta Data Carrier Detect Bit 2 = 1 Trailing Edge Ring Indicator Bit 1 = 1 Delta Data Set Ready Bit 0 = 1 Delta Clear To Send

continued

Function: 05h — Reset/Initialize Serial Port, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Functions: 06h – 0Ah — Reserved

Function: 0Bh — Set Modem Control

Description

This function is a single-staged request that sets the parameters for control of the modem as specified in the Request Block.

This function does not affect any Request Blocks already in process that have a Return Code of 0001h Resume Stage after Interrupt or 0005h Not My Interrupt, Resume Stage after Interrupt.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Bh)		
08h	Word	Reserved (initialize to 0000h)		
0Ah	Word	Reserved (initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
18h	Word	Reserved		
28h	Byte	Modem Control Bits 7-5 = Reserved (set to 0) Bit 4 = 1 Loop Bit 3 = 1 Out2 Note: This bit must be 0 if Bit 4 is 0. Bit 2 = Out1 Note: This bit must be 0 if Bit 4 is 0. Bit 1 = 1 Request-To-Send Bit 0 = 1 Data Terminal Ready		

continued

Function: 0Bh — Set Modem Control, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Ch — Set Line Control

Description

This function is a single-staged request that sets the parameters for the line as specified in the Request Block.

This function does not affect any Request Blocks already in process that have a Return Code of 0001h Resume Stage after Interrupt or 0005h Not My Interrupt, Resume Stage after Interrupt.

continued

Function: 0Ch — Set Line Control, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Ch)		
08h	Word	Reserved (initialize to 0000h)		
0Ah	Word	Reserved (initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
18h	Word	Reserved (initialize to 0000h)		
45h	Byte	Parity Type 00h = None 01h = Odd 02h = Even 03h = Stick parity odd 04h = Stick parity even		
46h	Byte	Number of Stop Bits Note: The following is true for a 6 bit, 7 bit, or 8 bit word length. 00h = 1 Stop bit 01h = 2 Stop bits Note: The following is true for a 5 bit word length. 00h = 1 Stop bit 01h = 1-1/2 Stop bits		
47h	Byte	Number of Bits Per Character 00h = 5 bits 01h = 6 bits 02h = 7 bits 03h = 8 bits		
48h	Byte	Break Status 00h = Disabled 01h = Enabled		

continued

Function: 0Ch — Set Line Control, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Dh — Set Baud Rate

Description

This function is a single-staged request that sets the baud rate as specified in the Request Block.

This function does not affect any Request Blocks already in process that have a Return Code of 0001h Resume Stage after Interrupt or 0005h Not My Interrupt, Resume Stage after Interrupt.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (000Dh)	
08h	Word	Reserved (initialize to 0000h)	
0Ah	Word	Reserved (initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		
18h	Word	Reserved (initialize to 0000h)	
44h	Byte	Transmission Baud Rate 00h = 110 01h = 150 02h = 300 03h = 600 04h = 1200 05h = 2400 06h = 4800 07h = 9600 08h = 19200 09h-0FFh = Reserved	

continued

Function: 0Dh — Set Baud Rate, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Eh — Transmit

Description

This function is a discrete multistaged request that enables the transmit interrupt. No data is transmitted until the actual transmit interrupt occurs.

Pointer field definitions

The Transmit Tail Pointer must point to the first byte of the data block to be transmitted. The value in this field should be relative to the beginning of the Transmit Buffer. The value in this field must always be between zero and the value in the Transmit Buffer Length field minus one.

The Transmit Head Pointer should point to a location one byte past the last byte to be sent. The value in this field should be relative to the beginning of the Transmit Buffer. The value in this field must always be between zero and the value in the Transmit Buffer Length field minus one.

continued

Function: 0Eh — Transmit, Continued

Updating the Transmit Tail Pointer field

When a transmit interrupt occurs, the interrupt routine increments the value of the Transmit Tail Pointer by the number of bytes transmitted. Bit 1 (Transmit In Progress) of the Current Serial Port Status field will be set.

The Transmit Buffer may be checked at any time during the transmit routine, so the BIOS writes data to the Transmit Buffer *before* the value in the Transmit Tail Pointer field is incremented.

Transmit Pointer field rules

The following rules apply to programming the transmit pointer fields:

- The Transmit Tail Pointer field value will always get closer to the Transmit Head Pointer value as interrupts occur.
 - If the value of the Transmit Tail Pointer comes to the end of the Transmit Buffer, the Transmit Tail Pointer field value wraps around to 0.
 - A Transmit Buffer Empty condition occurs when the value in the Transmit Tail Pointer equals the value in the Transmit Head Pointer field.
 - If a Transmit Buffer Empty condition happens after data is sent to the serial port, the BIOS stops sending data and sets bit 6 (Transmit Buffer Empty) of the Current Serial Port Status field (the transmit interrupt is still enabled). If the empty buffer condition persists on the next transmit interrupt (or any transmit interrupt), the BIOS will then disable the transmit interrupt.
 - If the transmit interrupt is disabled because the transmit buffer is empty, the Request Block is canceled by the BIOS (although it is up to the caller to deallocate the Request Block).
 - If a Request Block is canceled by the BIOS, the Current Serial Port Status field will have bit 6 (Transmit Buffer Empty) and bit 7 (Transmitter Holding Register Empty) set.
-

continued

Function: 0Eh — Transmit, Continued

Transmit buffer full

This condition occurs when the value in the Transmit Head Pointer field is one less than the value in the Transmit Tail Pointer field.

It also occurs if zero is in the Transmit Tail Pointer field and the value in the Transmit Head Pointer field is one less than the value in the Transmit Buffer Length field.

Adding data during a transmit interrupt

The values in the Transmit Buffer Segment, Transmit Buffer Offset, and Transmit Buffer Length fields can be altered across calls to the transmit interrupt routine.

The BIOS removes the data from the buffer before changing the Transmit Tail Pointer field, therefore the caller can add data to be transmitted during the process of a transmit interrupt by placing the data in the Transmit Buffer and logically incrementing the value in the Transmit Head Pointer field.

The caller must not permit the value in the Transmit Head Pointer field to equal the value in the Transmit Tail Pointer field while it adds data to the transmit buffer.

Maximum number of characters transmitted

The maximum number of characters that a single view of the Transmit Buffer can indicate to be transmitted is one less than the value in the Transmit Buffer Length field.

Number of bits per character

If the Number of Bits Per Character field in function 05h Reset/Initialize is less than 8, the high order bit(s) of each byte are transmitted as is.

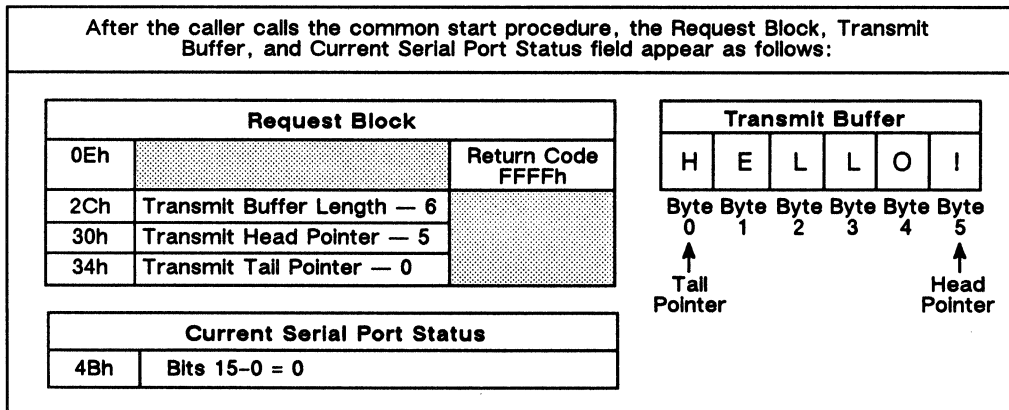
continued

Function: 0Eh — Transmit, Continued

Transmit sequence example

After the caller initializes the Request Block with a Transmit function (0Eh or 10h) and calls the start routine, the following actions take place:

1. The BIOS tests the Transmit Head and Tail Pointers to ensure that they agree with the Transmit Buffer Length. The maximum number of characters that can be transmitted in this example is 6.



continued

Function: 0Eh — Transmit, Continued

Transmit sequence example, cont'd

2. The BIOS start routine enables the transmitter interrupt but doesn't send any data yet.
3. BIOS tests the Transmitter Holding Register (bit 7 of the Current Serial Port Status field), finds it empty, and generates a transmit interrupt, setting the Return Code field to 0001h, Resume Stage after Interrupt.

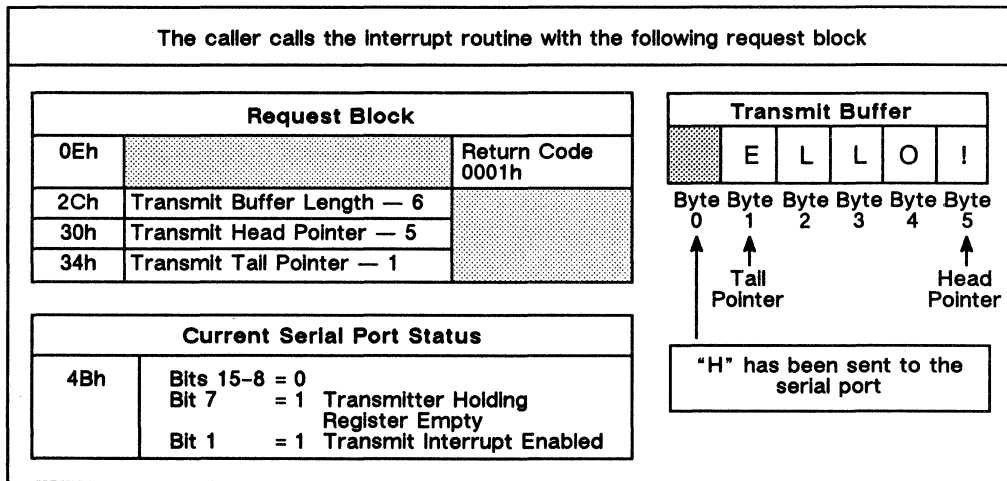
BIOS returns to the caller after the start routine with the following:		
Current Serial Port Status		
4Bh	Bits 15-8 = 0	
	Bits 7 = 1 Transmitter Holding Register Empty	
	Bits 6 = 0	
	Bit 1 = 1 Transmit Interrupt Enabled	
	Bit 0 = 0	
Request Block		
0Eh		Return Code 0001h
2Ch	Transmit Buffer Length — 6	
30h	Transmit Head Pointer — 5	
34h	Transmit Tail Pointer — 0	

continued

Function: 0Eh — Transmit, Continued

Transmit sequence example, cont'd

- The caller, seeing that the BIOS is waiting for the next interrupt, calls the interrupt routine. A character is sent to the active serial port and the Transmit Tail Pointer is incremented by one. Byte zero is sent in our example below.



continued

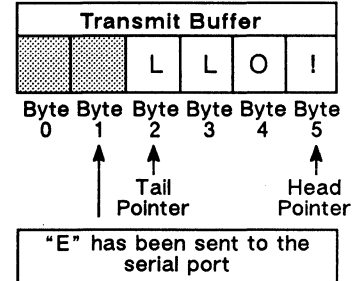
Function: 0Eh — Transmit, Continued

Transmit sequence example, cont'd

- Since the Transmitter Holder Register Empty bit is still set, an interrupt is generated by the processor. The caller calls the transmit interrupt routine again, and byte one is sent to the serial port. Again, the Transmit Tail Pointer is incremented by one. This process is repeated until all bytes in the Transmit Buffer are sent (assuming the Transmit Head Pointer is not moved by the caller). Actually, standard transmit procedure is that transmission occurs while the buffer is being filled, so the buffer is never empty until all bytes are transmitted.

Bit 7 in the current Serial Port Status full is set, so the processor generates another interrupt (the Return Code sent to the caller is 0001h again). The caller calls the interrupt routine to transmit another byte.

Request Block		
0Eh		Return Code 0001h
2Ch	Transmit Buffer Length — 6	
30h	Transmit Head Pointer — 5	
34h	Transmit Tail Pointer — 2	

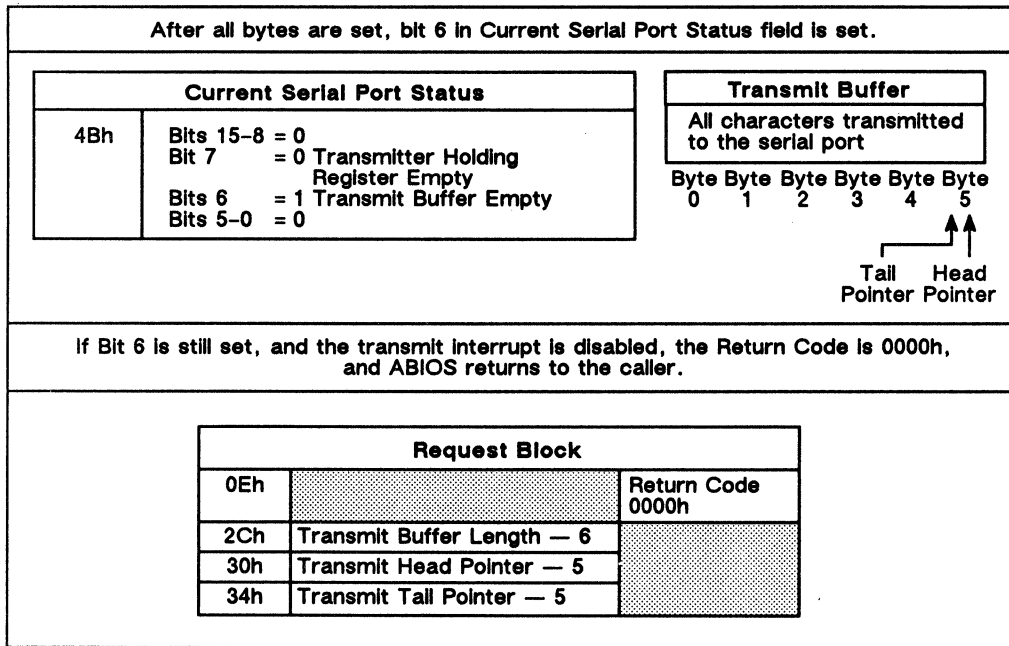


continued

Function: 0Eh — Transmit, Continued

Transmit sequence example, cont'd

6. When all bytes are sent, the Transmit Tail Pointer and Head Pointer point to the same location and bit 6 (Transmit Buffer Empty) is set. If bit 6 is still set on the next transmit interrupt, the ABIOS disables the transmit interrupt, sets a Return Code of 0000h, and returns to the caller.



continued

Function: 0Eh — Transmit, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Eh)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	Word	Reserved (Initialize to 0000h)		
12h	Word	Transmit Buffer Offset		
14h	Word	Transmit Buffer Segment		
18h	Word	Reserved (Initialize to 0000h)		
1Ah	DWord	Reserved (Initialize to 0000h)		
2Ch	Word	Transmit Buffer Length (Return Code is 0000h; no action taken if zero length input)		
2Eh	Word	Reserved (Initialize to 0000h)		
30h	Word	Transmit Head Pointer		
32h	Word	Reserved (Initialize to 0000h)		
34h	Word	Transmit Tail Pointer	Transmit Tail Pointer	
36h	Word	Reserved (Initialize to 0000h)	Reserved	
4Bh	Word		Current Serial Port Status (Initialize to 0000h) Bits 15-13 = Reserved Bit 12 = 1 Overrun Error, with Null Data Byte Found Bit 11 = 1 Break Detected Bit 10 = 1 Framing Error Bit 9 = 1 Parity Error Bit 8 = 1 Overrun Error Bit 7 = 1 Transmit Buffer Empty and Transmitter Holding Register Empty Bit 6 = 1 Transmit Buffer Empty Bit 5 = 1 Receive Buffer Full and Data Deleted Bit 4 = 1 Receive Buffer Full Bits 3-2 = Reserved Bit 1 = 1 Transmit Interrupt in Progress Bit 0 = 1 Receive Interrupt in Progress	

continued

Function: 0Eh — Transmit, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0005h	Not My Interrupt, Resume Stage after Interrupt
0081h	Spurious Interrupt
8000h	Device Busy, Request Refused
9000h	Bad Com Port
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Fh — Receive

Description

This function is a discrete multistaged request that enables the receive interrupt. Data can be read from the serial port when a receive interrupt is generated after this function is invoked.

Pointer field definitions

The Receive Head Pointer must point to the first byte of the data block to be received. The value in this field should be relative to the beginning of the Receive Buffer. The value in this field must always be between zero and the value in the Receive Buffer Length field minus one.

The Receive Tail Pointer should point to a location one byte past the last byte to be received. The value in this field should be relative to the beginning of the Receive Buffer. The value in this field must always be between zero and the value in the Receive Buffer Length field minus one.

Head Pointer and Tail Pointer fields

The value in the Receive Head Pointer field points to the first character received position that will be filled by the BIOS. The Value in the Receive Tail Pointer field points to the first received character to be removed by the caller.

Updating the Receive Tail Pointer field

When a receive interrupt occurs, the interrupt routine increments the value of the Receive Tail Pointer by the number of bytes received. Bit 0 (Receive In Progress) of the Current Serial Port Status field will be set.

The Receive Buffer may be checked at any time during the receive routine, so data is written to the Receive Buffer *before* the value in the Receive Tail Pointer field is incremented.

continued

Function: 0Fh — Receive, Continued

Receive Pointer rules

The following rules apply to processing the receive head and tail pointers:

- The Receive Tail Pointer field value will always get closer to the Receive Head Pointer value as interrupts occur.
- If the value of the Receive Tail Pointer comes to the end of the Receive Buffer, the Receive Tail Pointer field value wraps around to 0.
- A Receive Buffer Full condition occurs when the value in the Receive Tail Pointer equals the value in the Receive Head Pointer field.
- If a Receive Buffer Full condition happens after data is received from the serial port, the BIOS stops receiving data and sets bit 4 (Receive Buffer Full) of the Current Serial Port Status field (the receive interrupt is still enabled). If the full buffer condition persists on the next receive interrupt (or any receive interrupt), the BIOS will then disable the receive interrupt.
- If the receive interrupt is disabled because the receive buffer is empty, the Request Block is canceled by the BIOS (although it is up to the caller to deallocate the Request Block).
- If a Request Block is canceled by the BIOS, the Current Serial Port Status field will have bit 4 (Receive Buffer Empty) and bit 5 (Receive Buffer Full With Data Deleted) set.

Receive buffer empty

This condition occurs when the value in the Receive Head Pointer field equals the value in the Receive Tail Pointer field. The BIOS never sets the value of the Receive Head Pointer field equal to the value in the Receive Tail Pointer field, however, the caller may do so, which is a quick way to stop receiving.

continued

Function: 0Fh — Receive, Continued

How the caller can remove data during a receive interrupt

The values in the Receive Buffer Segment, Receive Buffer Offset, and Receive Buffer Length fields can be altered across calls to the receive interrupt routine.

The BIOS removes the data from the buffer before changing the Receive Tail Pointer field. The caller can remove data during the process of a receive interrupt by removing the data from the Receive Buffer and logically incrementing the value in the Receive Head Pointer field.

The caller must not permit the value in the Transmit Head Pointer field to equal the value in the Transmit Tail Pointer field during the removal of data from the buffer.

Maximum number of characters received

The maximum number of characters that a single view of the Receive Buffer can indicate to be received is one less than the value in the Receive Buffer Length field.

Ending a receive interrupt

A receive interrupt can only be terminated by invoking BIOS Serial Communications Service function 12h Cancel.

Number of bits per character

If the Number of Bits Per Character field in function 05h Reset/Initialize is less than 8, the high order bit(s) of each byte are transmitted as is.

continued

Function: 0Fh — Receive, Continued

Null Stripping Mode

If...	the following action is taken
Null Stripping Mode is disabled	All received characters are stored unaltered in the Receive Buffer.
Null Stripping Mode is enabled	No received zeros are stored in the Receive Buffer.
An Overrun occurs and a Null Data Byte caused the error	The Null Data Byte is discarded. The Current Serial Port Status indicates that an overrun error was found and the Null Data Byte was found and discarded and bit 12 of the Current Serial Port Status field is set.

Hardware errors

If a hardware error occurs at the serial port, the BIOS stores the current data byte in the receive buffer, sets bits 8–10 of the Current Serial Port Status field, and returns to the caller. The BIOS returns on the *first* hardware error, though there may be additional errors and additional data in the hardware buffer.

If there are additional hardware errors at the serial port, the caller must handle them and any data that is in the buffer.

The following table lists the actions taken for each type of error.

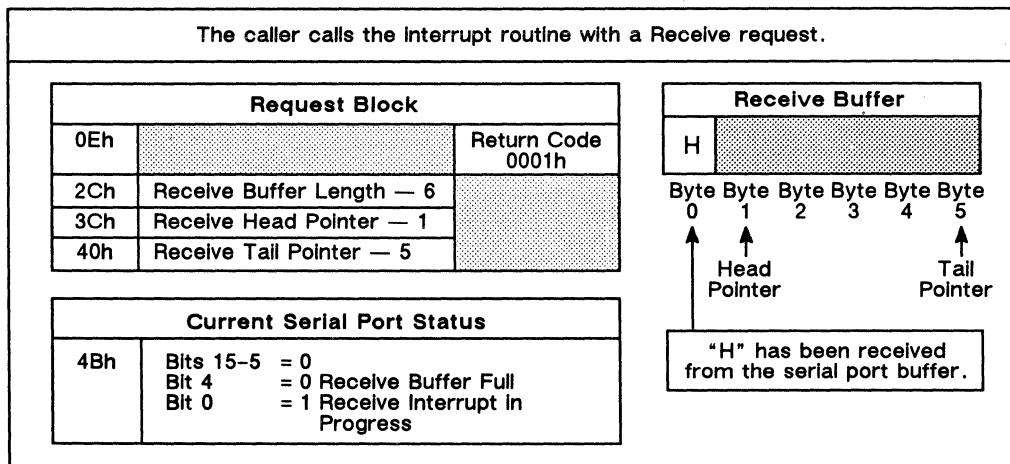
Error Type	Action taken by BIOS
Break Interrupt	Data Byte set to zero.
Overrun	Overrun character is lost, the Data Byte contains the character that caused the overrun.
Parity Error	Data Byte contains the character with the incorrect parity.
Framing Error	Data Byte contains the character that has no valid stop bit.

continued

Function: 0Fh — Receive, Continued

Receive sequence example, cont'd

2. The BIOS start routine enables the receive interrupt but doesn't read any data yet.
3. When data is available at the serial port, an interrupt is generated.
4. The caller, seeing that the BIOS is waiting for the next interrupt, calls the interrupt routine. A character is read from the active serial port and the Receive Tail Pointer is incremented by one. Byte zero is received in the example below.



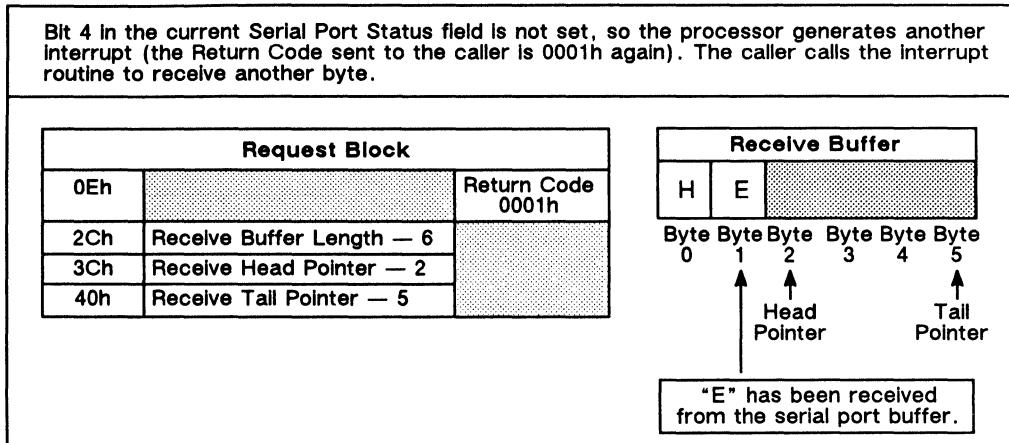
continued

Function: 0Fh — Receive, Continued

Receive sequence example, cont'd

5. Since there are five empty buffer bytes, the data received is read into each one in turn when the next receive interrupts are generated. The Receive Head Pointer is incremented by one each time.

Bit 4 in the current Serial Port Status field is not set, so the processor generates another interrupt (the Return Code sent to the caller is 0001h again). The caller calls the interrupt routine to receive another byte.



Note: If two or more characters are received at the same time, the Receive Head Pointer and the Receive Tail Pointer are incremented by 2 or more. The Tail Pointer may then wrap to byte 0. The calling routine should anticipate this possibility.

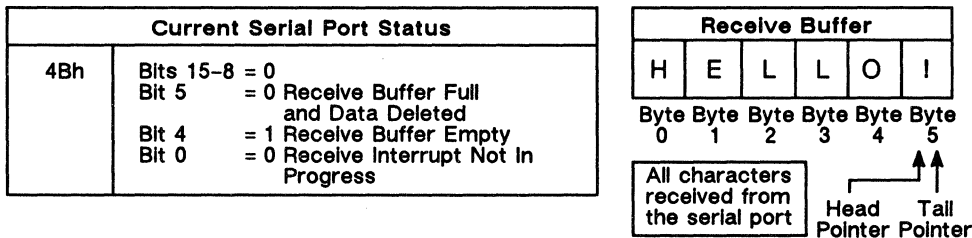
continued

Function: 0Fh — Receive, Continued

Receive sequence example, cont'd

6. When all bytes that the receive buffer has room for are received, the Receive Head pointer will point to the same byte as the Receive Tail Pointer, (or the Receive Tail Pointer will be one byte less than the Receive Buffer Length). At this point, a Return Code of 0000h is generated and control is returned to the caller. In actual practice, the tail pointer will chase the head pointer and the buffer never actually fills.

After all bytes in the receive buffer are filled, and the Tail Pointer and Head Pointer are the same, bit 4 in the Current Serial Port Status field is set.



If bit 4 is still set, and the receive interrupt is disabled, the Return Code is set to 0000h, and BIOS returns control to the caller.

Request Block		
0Eh		Return Code 0000h
2Ch	Receive Buffer Length — 6	
3Ch	Receive Head Pointer — 5	
40h	Receive Tail Pointer — 5	

7. If the Buffer Full condition exists upon entry into the receive function interrupt routine, the character that caused the interrupt is lost, and bit 5 of the Current Serial Port Status field (Receive Buffer Full and Data Deleted) is set.

continued

Function: 0Fh – Receive, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Fh)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
18h	Word	Reserved (Initialize to 0000h)		
1Ah	DWord	Reserved (Initialize to 0000h)		
20h	Word	Reserved (Initialize to 0000h)		
22h	Word	Receive Buffer Offset		
24h	Word	Receive Buffer Segment		
28h	Byte	Null Stripping Mode 00h = Disabled 01h = Enabled		
38h	Word	Receive Buffer Length Note: No action occurs if this field is 0.		
3Ah	Word	Reserved (Initialize to 0000h)		
3Ch	Word	Receive Head Pointer	Receive Head Pointer	
3Eh	Word	Reserved (Initialize to 0000h)	Reserved	
40h	Word	Receive Tail Pointer		
42h	Word	Reserved (Initialize to 0000h)		

continued

Function: 0Fh — Receive, Continued

Request Block Structure, cont'd

Offset	Size	Input:	Output:
4Bh	Word		Current Serial Port Status (Initialize to 0000h) Bits 15-13 = Reserved Bit 12 = 1 Overrun Error, with Null Data Byte Found Bit 11 = 1 Break Detected Bit 10 = 1 Framing Error Bit 9 = 1 Parity Error Bit 8 = 1 Overrun Error Bit 7 = 1 Transmit Buffer Empty and Transmitter Holding Register Empty Bit 6 = 1 Transmit Buffer Empty Bit 5 = 1 Receive Buffer Full and Data Deleted Bit 4 = 1 Receive Buffer Full Bits 3-2 = Reserved Bit 1 = 1 Transmit Interrupt in Progress Bit 0 = 1 Receive Interrupt in Progress

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0005h	Not My Interrupt, Resume Stage after Interrupt
0081h	Spurious Interrupt
8000h	Device Busy, Request Refused
9000h	Bad Com Port
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 10h — Transmit and Receive

Description

This function is a discrete multistaged request that starts both the transmit and receive interrupts, or starts only the receive interrupt. It allows faster access to the transmit and receive functions than would be the case if separate function requests were used.

Adding a transmit function to a receive-only function

Invoke a receive-only request with this function by only enabling the receive parameters (set the Transmit Buffer Length to zero), then calling the start routine. The Return Code field, as always, must be initialized to FFFFh when calling the start routine.

The transmit function can be requested later by recalling the start routine with an appropriate value in the Transmit Buffer Length field. *Do not initialize the Return Code field to FFFFh when recalling the start routine.*

If this function is restarted to enable the transmit after being initially set to receive only, the Return Code field is undefined when returning from recalling the start routine and should be ignored.

Transmit and Receive function characteristics

This function performs the transmit function as described for function 0Eh and the receive function as described for function 0Fh, except that during a receive interrupt, the transmit interrupt is never disabled. In processing this function, the BIOS does not test if the Transmit Head Pointer and Tail Pointer are equal.

continued

Function: 10h — Transmit and Receive, Continued

Input fields to initialize

After the first time this function is called via a start routine (the Return Code field should be initialized to FFFFh the first time), the caller should initialize the Current Serial Port Status field and the Return Code field from the previous interrupt to 0000h before the start routine for this function is called again.

Multiple receive interrupts

After servicing a receive interrupt, the BIOS tests for a transmit interrupt pending condition.

If a transmit Interrupt is pending, it is serviced by the BIOS and Return Code 0009h Attention, Resume Stage after Interrupt is generated. But if a second receive interrupt is pending, the BIOS does not service the pending transmit interrupt, returning to the caller instead.

continued

Function: 10h – Transmit and Receive, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0010h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	Word	Reserved (Initialize to 0000h)		
12h	Word	Transmit Buffer Offset		
14h	Word	Transmit Buffer Segment		
18h	Word	Reserved (Initialize to 0000h)		
1Ah	DWord	Reserved (Initialize to 0000h)		
20h	Word	Reserved (Initialize to 0000h)		
22h	Word	Receive Buffer Offset		
24h	Word	Receive Buffer Segment		
28h	Byte	Null Stripping Mode 00h = Disabled 01h = Enabled		
2Ch	Word	Transmit Buffer Length (bytes) If zero and first call, Receive-only function. If zero and second call, Return Code = 0001h and no action is performed		
2Eh	Word	Reserved (Initialize to 0000h)		
30h	Word	Transmit Buffer Head		
32h	Word	Reserved (Initialize to 0000h)		
34h	Word	Transmit Buffer Tail		Transmit Buffer Tail
36h	Word	Reserved (Initialize to 0000h)		Reserved
38h	Word	Receive Buffer Length If zero, no action performed, Return Code = 0000h		
3Ah	Word	Reserved (Initialize to 0000h)		
3Ch	Word	Receive Buffer Head		Receive Buffer Head
3Eh	Word	Reserved (Initialize to 0000h)		Reserved
40h	Word	Receive Buffer Tail		
42h	Word	Reserved (Initialize to 0000h)		

continued

Function: 10h — Transmit and Receive, Continued

Request Block Structure, cont'd

Offset	Size	Input:	Output:
4Bh	Word		Current Serial Port Status (Initialize to 0000h) Bits 15-13 = Reserved Bit 12 = 1 Overrun Error, with Null Data Byte Found Bit 11 = 1 Break Detected Bit 10 = 1 Framing Error Bit 9 = 1 Parity Error Bit 8 = 1 Overrun Error Bit 7 = 1 Transmit Buffer Empty and Transmitter Holding Register Empty Bit 6 = 1 Transmit Buffer Empty Bit 5 = 1 Receive Buffer Full and Data Deleted Bit 4 = 1 Receive Buffer Full Bits 3-2 = Reserved Bit 1 = 1 Transmit Interrupt in Progress Bit 0 = 1 Receive Interrupt in Progress
4Dh	Word		Com Port Status of Previous Interrupt
4Fh	Word		Return Code of Previous Interrupt

continued

Function: 10h — Transmit and Receive, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0005h	Not My Interrupt, Resume Stage after Interrupt
0009h	Attention, Resume Stage after Interrupt The Com Port Status of the Previous Interrupt field (offset 4Dh) contains the original status. The Return Code of the Previous Interrupt (offset 4Fh) has the first Return Code.
0081h	Spurious Interrupt
8000h	Device Busy, Request Refused
9000h	Bad Com Port Indicates a hardware failure
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 11h — Modem Status

Description

This function is a discrete multistaged request that returns the Modem Status upon exit from the start and interrupt routines. After this function is invoked, the modem status interrupt is enabled.

ABIOS notes changes in modem status immediately

If the Request Block that invokes this function is processed before a Transmit or Receive function (functions 0Eh, 0Fh, or 10h), the ABIOS notices a change in the modem status even though there may be higher priority interrupts in the interrupt identification register that are still pending. This allows an interrupt handler routine to notice a change in Modem Status before receiving or transmitting data.

continued

Function: 11h – Modem Status, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0011h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
18h	Word	Reserved (Initialize to 0000h)		
1Ah	DWord	Reserved (Initialize to 0000h)		
4Ah	Byte		Modem Status Bit 7 = 1 Data Carrier Detect Bit 6 = 1 Ring Indicator Bit 5 = 1 Data Set Ready Bit 4 = 1 Clear to Send Bit 3 = 1 Change in Data Carrier Detect Bit 2 = 1 Trailing Edge Ring Indicator Bit 1 = 1 Change in Data Set Ready Bit 0 = 1 Change in Clear To Send	
4Bh	Byte		Current Serial Port Status (Initialize to 0000h) Bits 15–13 = Reserved Bit 12 = 1 Overrun Error, with Null Data Byte Found Bit 11 = 1 Break Detected Bit 10 = 1 Framing Error Bit 9 = 1 Parity Error Bit 8 = 1 Overrun Error Bit 7 = 1 Transmit Buffer Empty and Transmitter Holding Register Empty Bit 6 = 1 Transmit Buffer Empty Bit 5 = 1 Receive Buffer Full and Data Deleted Bit 4 = 1 Receive Buffer Full Bits 3–2 = Reserved Bit 1 = 1 Transmit Interrupt in Progress Bit 0 = 1 Receive Interrupt in Progress	

continued

Function: 11h — Modem Status, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0005h	Not My Interrupt, Resume Stage after Interrupt
0081h	Spurious Interrupt
8000h	Device Busy, Request Refused
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 12h — Cancel

Description

This function cancels a request for serial transmission (functions 0Eh or 10h), receiving (functions 0Fh or 10h), modem status (function 11h) or any combination of these three functions. It also cancels the associated Request Blocks, except for function 10h Transmit and Receive (see below). All interrupts generated by the request are disabled, except for function 10h Transmit and Receive, which is handled differently (see below).

Note: The caller must deallocate all canceled Request Blocks. The BIOS does not deallocate Request Blocks.

Canceling function 10h Transmit and Receive

if...	then...
Function 10h Transmit and Receive has both the transmit and receive interrupts enabled, and function 12h Cancel is invoked with the intent of disabling both interrupts,	the Request Block for function 10h is canceled by the BIOS.
Function 12h Cancel is to disable only the transmit interrupt,	the caller may re-invoke the 10h Transmit and Receive function via the start routine after the function 12h Cancel request is completed.
Function 12h Cancel is invoked with the intent of disabling only the receive interrupt,	the caller must complete the following steps to re-enable the receive interrupt: <ul style="list-style-type: none">▪ disable the transmit interrupt▪ call the start routine with function 10h Transmit and Receive specified in the Request Block
Function 10h Transmit and Receive has only the receive interrupt enabled, and function 12h Cancel is invoked to disable the receive interrupt,	the entire function 10h Transmit and Receive Request Block is canceled by the BIOS.
Function 10h Transmit and Receive has only the transmit interrupt enabled, and function 12h Cancel is invoked to disable the transmit interrupt	the entire function 10h Request Block is canceled by the BIOS.

continued

Function: 12h — Cancel, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0012h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
18h	Word	Reserved (Initialize to 0000h)		
51h	Byte	Interrupt Operation to be Canceled Bit 7 = Reserved Bit 6 = Reserved Bit 5 = 1 Modem Status interrupt Bit 4 = Reserved Bit 3 = 1 Transmit interrupt Bit 2 = 1 Receive interrupt Bit 1 = Reserved Bit 0 = Reserved		

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 13h — Return Line Status

Description

This function, a single-staged request, reads the Modem Line Status.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0013h)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Line Status Bit 7 = Reserved Bit 6 = 1 Transmitter Empty Bit 5 = 1 Transmitter Holding Register Empty Bit 4 = 1 Break Interrupt Bit 3 = 1 Framing Error Bit 2 = 1 Parity Error Bit 1 = 1 Overrun Error Bit 0 = 1 Data Ready
49h	Byte		

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 14h — Return Modem Status

Description

This function is a single-staged request that reads the current status of the modem.

If the Modem Status interrupt is enabled, no processing takes place and the Request Refused, Device Busy Return Code (8000h) is returned.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0014h)	
08h	Word	Reserved (initialize to 0000h)	
0Ah	Word	Reserved (initialize to 0000h)	
0Ch	Word	Return Code	Return Code
0Eh	Word		Modem Status Bit 7 = 1 Data delta Detect Bit 6 = 1 Ring Indicator Bit 5 = 1 Data Set Ready Bit 4 = 1 Clear To Send Bit 3 = 1 Delta Data Carrier Detect Bit 2 = 1 Trailing Edge Ring Indicator Bit 1 = 1 Delta Data Set Ready Bit 0 = 1 Delta Clear To Send
4Ah	Byte		

continued

Function: 14h — Return Modem Status, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8000h	Device Busy, Request Refused
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 15h — Enable FIFO Control

Description

This function is a single-staged request that enables FIFO (first-in, first-out) control for the serial port. The FIFO registers are not cleared.

This function may only be used if the BIOS is operating on an 80386-based, MCA-based system which has an NS16550A serial controller. If the serial controller is not an NS16550A, unpredictable results may occur.

Reusing this function

BIOS calling routines should not reuse this Request Block for other Serial Communications Service function requests, since the structure of this Request Block is different from the other Serial Communications Service Request Blocks.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0015h)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Return Code
28h	Word	FIFO Trigger Level, where: 00h — 1 Byte 01h — 4 Bytes 02h — 8 Bytes 03h — 14 Bytes	Time-out

continued

Function: 15h — Enable FIFO Control, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8000h	Device Busy, Request Refused
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Chapter 13

ABIOS Parallel Port Service

Overview

Introduction

The ABIOS Parallel Port Service provides access to the system's parallel ports. Parallel ports on personal computers are most often used to send data to printers. This ABIOS service supports only the Transmit Mode of the parallel port.

continued

Overview, Continued

Summary of Parallel Port Service functions

Function	Description
00h	Default Interrupt Handler
01h	Return Logical ID Parameters
02h	Reserved
03h	Read Device Parameters
04h	Set Device Parameters
05h	Reset/Initialize Parallel Port
06h-08h	Reserved
09h	Print Block
0Ah	Reserved
0Bh	Cancel Print Block
0Ch	Return Printer Status

In this chapter

This chapter includes information about the following topics:

- Hardware Environment
 - Error Handling
 - Parallel Port Service functions
-

Hardware Environment

Introduction

The BIOS Parallel Port Service is associated with hardware interrupt request 7. The BIOS supports a parallel port that can transfer eight bits of data at standard TTL levels. The parallel port can be called port 1 through 8, must be IBM PS/2-compatible, and must have a bidirectional mode, supporting both input and output. The parallel port also supports level-sensitive interrupts and a readable interrupt pending status.

Parallel port addresses

The following table lists the parallel port addresses for the most frequently used parallel ports. The I/O port addresses for parallel ports 4 through 8 vary among different systems.

Parallel Port Number	Data Address	Status Address	Control Address
1	03BCh	03BDh	03BEh
2	0378h	0379h	037Ah
3	0278h	0279h	027Ah

Parallel port extended mode

The extended mode of the parallel port can be selected through the system-based POS registers. The extended mode adds a bidirectional interface.

Error Handling

Parallel Port Service errors

If the parallel port is busy when the caller invokes an BIOS Parallel Port Service function, Return Code 8000h Device In Use is returned. The caller must then re-invoke the function, except when using function 09h Print Block. See the function 09h Print Block description for more information about re-invoking function 09h Print Block.

How errors are reported

BIOS signals the status (Successful, Resume Stage after Interrupt, etc.) of each BIOS request by returning a one word Return Code at offset 0Ch in the Request Block.

If Bit 15 of the Return Code field is set, the Parallel Port Service function requested has an error. The caller's Return Code handler routine should then test Bits 14, 13, 12, and 8 to determine the class of error that has occurred. The return code handler routine should then test the remaining bits to determine the precise nature of the error.

Function: 00h — Default Interrupt Handler

Description

This function is a single-staged or multistaged request that handles unexpected hardware interrupts by resetting the interrupt at the device level. It is invoked through the interrupt routine.

When invoked

This function is invoked by calling the interrupt routine with a function code of 0000h. It is only invoked if a given Logical ID has no outstanding Request Blocks waiting for an interrupt.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length (10h)	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0000h)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out

Return Code

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0005h	Not My Interrupt, Resume Stage after Interrupt
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 01h — Return Logical ID Parameters

Description

This function is a single-staged request that returns the parameters for the specified Logical ID.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length (20h)		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0001h)		
08h	Word	Reserved (initialize to 0000h)		
0Ah	Word	Reserved (initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Byte			Hardware Interrupt Level (07h)
11h	Byte			Arbitration Level (FFh)
12h	Word		Device ID (0005h)	
14h	Word		Count of Units	
16h	Word		Logical ID Flags Bits 15-4 = Reserved Bit 3 = 0 No overlap across units = 1 Overlap across units supported Bit 2 = 0 Reserved Bits 1-0 = Transfer Data Pointer Mode 01 = Logical Pointer Required	
18h	Word		Request Block Length (for other functions)	
1Ah	Byte	Reserved (initialize to 0000h)	Secondary Device ID	
1Bh	Byte		Revision Level	
1Ch	Word	Reserved (initialize to 0000h)		
1Eh	Word	Reserved (initialize to 0000h)		

continued

Function: 01h — Return Logical ID Parameters, Continued

Return Code

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Ah — Reserved

Function: 03h — Read Device Parameters

Description

This function is a single-staged request that returns the parameters of the specified Logical ID.

Input field values

The value in the Time to Wait for Printer Initialization field is the same as the value in the Time To Wait Before Continuing Request (microseconds) field returned by the Reset/Initialize Parallel Port function (05h).

The value in the Printer Interrupt Time-out field is the wait time for an interrupt at the device level returned by the Print Block function (09h).

continued

Function: 03h — Read Device Parameters, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0003h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
16h	Word	Reserved (Initialize to 0000h)		
20h	DWord			Time to Wait for Printer Initialization (microseconds)
29h	Byte		Interrupt Level	
2Ah	Word		Printer Interrupt Time-out Bits 15-3 = Time-out (In seconds) Bits 2-0 = Reserved	

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 04h — Set Device Parameters

Description

This function is a single-staged request that sets device specific information as specified in the Request Block.

Input parameters

The BIOS uses the parameters input in this function until they are changed by another function 04h request for this service.

The value in the Time To Wait Before Continuing Request field returned by the Reset/Initialize Parallel Port function (05h) can be entered into the Time to Wait for Printer Initialization field at offset 20h.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0004h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
16h	Word	Reserved (Initialize to 0000h)		
20h	DWord	Time to Wait for Printer Initialization (microseconds) Note: This field must not be 0.		
2Ah	Word	Printer Interrupt Time-out Bits 15-3 = Time-out (In seconds) Bits 2-0 = Reserved Note: This field must not be 0.		

continued

Function: 04h — Set Device Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8000h	Device In Use If the Parallel Port is busy, this Code is returned and the function is terminated.
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C006h	Invalid Time To Wait The Time to Wait for Printer Initialization field must not be zero. If it is zero, this Return Code is generated, no update is performed, and the function is terminated.
FFFFh	Return Code Field Not Valid

Function: 05h — Reset/Initialize Parallel Port

Description

This function is a discrete multistaged request that initializes or resets the parallel printer.

Printer Busy Status

After completing this function, the printer may indicate a Busy status because it is performing a self test. Bit 7 (Busy) of the Printer Status at offset 28h in the Request Block indicates this condition.

Output considerations

The Time To Wait Before Continuing Request field value is returned only when the Return Code is 0002h Resume Stage after Time Delay. The value in this field may not be valid if the Return Code is not 0002h.

The Printer Status field and all other output fields are not valid unless this function has completed.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0005h)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Return Code
16h	Word	Reserved (Initialize to 0000h)	Time-out
20h	DWord		Time To Wait Before Continuing Request (microseconds)
28h	Byte		Printer Status Bit 7 = Busy Bit 6 = Acknowledge Bit 5 = End of paper Bit 4 = Selected Bit 3 = I/O error Bit 2 = Interrupt Bits 1, 0 = Reserved

continued

Function: 05h — Reset/Initialize Parallel Port, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0002h	Resume Stage after Time Delay
8000h	Device In Use
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Functions: 06h – 08h — Reserved

Function: 09h — Print Block

Description

This function is a discrete multistaged request that sends the block of characters pointed to by Data Pointer 1 to the parallel port.

Using the Print Block function

The following steps must be completed before the Print Block function can be invoked:

1. Invoke function 01h Read Device Parameters to determine the Data Pointer Mode (logical for Parallel Port Data Pointers). If there is an outstanding Print Block function already enabled, function 0Bh Cancel Print Block must be invoked before function 9Fh is invoked.
2. Invoke function 09h Print Block
3. Check the Printer Status. The Printer Status field data is valid only if this function is completed. The BIOS does the following, depending on the Printer Status bit settings:

If...	and...	then BIOS...
the Printer Status is Busy (Bit 7) and the BIOS reads the Printer Status field for 91.2 microseconds	the Status is still Busy,	returns 8001h and terminates.
the printer is switched offline during a Print Block (Bit 2) and the BIOS reads the Printer Status field for 91.2 microseconds	the Status is still Interrupt on,	returns 8001h and terminates.
the printer is out of paper, has an I/O error, or is not selected (Bits 3, 4, or 5) and the BIOS reads the Printer Status field for 91.2 microseconds	either Bits 3, 4, or 5 of the Printer Status are still on,	returns 9000h and terminates.

4. The caller can then complete the Print Block function as described below.
-

continued

Function: 09h — Print Block, Continued

Completing a Print Block Request after an unexpected termination

If this function is terminated, the Number of Characters Printed field contains the part of the print block that has already been printed, in bytes.

The unprinted portion of the print block can be printed by invoking function 09h Print Block again after correcting the condition that caused the termination of the initial Print Block function. The number of characters remaining to be printed is equal to the original print block length minus the value in the Number of Characters Printed field.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0009h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	Word	Reserved (Initialize to 0000h)		
12h	DWord	Data Pointer 1 (Logical Pointer)		
16h	Word	Reserved (Initialize to 0000h)		
18h	Word	Reserved (Initialize to 0000h)		
1Ah	DWord	Reserved		
1Eh	Word	Reserved (Initialize to 0000h)		
24h	Word	Number of Characters to Print	Number of Characters Printed	
26h	Word		Printer Status Bit 7 = Busy Bit 6 = Acknowledge Bit 5 = End of paper Bit 4 = Selected Bit 3 = I/O error Bit 2 = Interrupt Bits 1, 0 = Reserved	
28h	Byte			

continued

Function: 09h — Print Block, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0005h	Not My Interrupt, Resume Stage after Interrupt
8000h	Device In Use
8001h	Device Busy If the printer does not indicate that it is ready to accept data within 91.2 microseconds, the Printer Status field is updated, the printer interrupt is disabled, and control is returned to the caller with this Code.
9000h	Printer Error Printer I/O error, out of paper, paper jam, or printer not selected.
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Ah — Reserved

Function: 0Bh — Cancel Print Block

Description

This function is a single-staged request that cancels an outstanding function 09h Print Block by disabling the Printer Interrupt at the device level and changing the Return Code.

The Printer Status field is valid only if this function has successfully completed.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (000Bh)	
08h	Word	Reserved (initialize to 0000h)	
0Ah	Word	Reserved (initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out
16h	Word	Reserved (initialize to 0000h)	
26h	Word		Number of Characters Printed
28h	Byte		Printer Status Bit 7 = Busy Bit 6 = Acknowledge Bit 5 = End of paper Bit 4 = Selected Bit 3 = I/O error Bit 2 = Interrupt Bits 1, 0 = Reserved

continued

Function: 0Bh — Cancel Print Block, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Ch — Return Printer Status

Description

This function is a single-staged or multistaged request that returns the printer status.

The printer status data is valid only if the function is completed successfully (the Return Code is 0000h).

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Ch)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
16h	Word	Reserved (Initialize to 0000h)		
28h	Byte			Printer Status Bit 7 = Busy Bit 6 = Acknowledge Bit 5 = End of paper Bit 4 = Selected Bit 3 = I/O error Bit 2 = Interrupt Bits 1, 0 = Reserved

continued

Function: 0Ch — Return Printer Status, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0002h	Resume Stage after Time Delay
8000h	Device In Use
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Chapter 14

ABIOS System Timer Service

Overview

Description

The ABIOS System Timer Service provides access to the PS/2-compatible System Timer. The System Timer Interrupt is handled through function 00h Default Interrupt Handler.

Summary of System Timer Service functions

Function	Description
00h	Default Interrupt Handler
01h	Return Logical ID Parameters
02h-0Ah	Reserved

In this chapter

In this chapter, the following topics are discussed:

- Hardware Environment
 - Error Handling
 - ABIOS System Timer Service functions
-

Hardware Environment

Introduction

PS/2-compatible operation of the Programmable Timer is similar to the operation of the Intel 8254A Programmable Interval Timer. Unlike the 8254A, the PS/2 programmable timer provides no channel 1 and adds a channel 3 with limited functionality.

Programmable Timer

The PS/2-compatible Programmable Timer is a counter and timer that provides three channels. All channels are driven by a 1.19 MHz oscillator signal. Each "tick" of channel 0 generates hardware interrupt request 0.

Timer channel differences

There are some differences between the three timer channels:

Counters 0 and 2:

- are independent 16-bit counters,
- can be preset, and
- can count in BCD (binary coded decimal) or in binary.

Counter 3 (associated only with channel 0):

- is only 8 bits,
 - can be preset,
 - counts in binary only, and
 - can only count downward.
-

Error Handling

How errors are reported

ABIOS signals the status (Successful, Resume Stage after Interrupt, etc.) of each ABIOS request by returning a one word Return Code at offset 0Ch in the Request Block.

If Bit 15 of the Return Code field is set, the System Timer Service function requested has an error. The caller's Return Code handler routine should then test Bits 14, 13, 12, and 8 to determine the class of error that has occurred. The return code handler routine should then test the remaining bits to determine the precise nature of the error.

Function: 00h — Default Interrupt Handler

Description

This function is a single-staged or multistaged request that handles unexpected hardware interrupts by resetting the interrupt at the device level. This function is only invoked if a given Logical ID has no outstanding Request Blocks waiting for an interrupt.

If successful, this function resets the Timer 0 channel at I/O port 0061h and returns to the caller with a Return Code of 0000h.

If another interrupt routine is entered by any other caller while this function is being processed, Return Code 0005h is generated.

Request Block structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length (10h)	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0000h)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0005h	Not My Interrupt, Resume Stage after Interrupt
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 01h — Return Logical ID Parameters

Description

This function is a single-staged request that returns information about the specified Logical ID.

Request Block structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length (20h)		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0001h)		
08h	Word	Reserved (initialize to 0000h)		
0Ah	Word	Reserved (initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Byte			Hardware Interrupt Level (00h)
11h	Byte			Arbitration Level (FFh)
12h	Word		Device ID (0007h)	
14h	Word		Count of Units (0000h)	
16h	Word		Logical ID Flags (0000h) Bits 15-4 = Reserved Bit 3 = 0 No overlap across units Bit 2 = 0 Reserved Bits 1-0 = Transfer Data Pointer Mode 00 = No Pointers Required	
18h	Word		Request Block Length (for other functions)	
1Ah	Byte		Reserved (initialize to 0000h)	Secondary Device ID
1Bh	Byte		Revision Level	
1Ch	Word	Reserved (initialize to 0000h)		
1Eh	Word	Reserved (initialize to 0000h)		

continued

Function: 01h — Return Logical ID Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0005h	Not my interrupt, Resume Stage after Interrupt
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Functions: 02h – 0Ah — Reserved

Chapter 15

ABIOS Real Time Clock Service

Overview

Introduction

The ABIOS Real Time Clock Service provides access to the system's real time clock (RTC) functions.

Real time clock interrupt types

This service controls the operation of the following RTC interrupt types:

- **Alarm interrupt**

An interrupt that is activated at intervals of from once per second to once per day.

- **Periodic interrupt**

An interrupt that is activated at a prespecified interval, which can be from .5 second to 30.517 ms.

- **Update-Ended interrupt**

An interrupt that is activated every time the system clock is updated, or once per second.

See the Time-of-Day Service chapter in *CBIOS for IBM PS/2 Computers and Compatibles* in the Addison-Wesley Phoenix Technical Reference Series for more information on these interrupt types.

continued

Overview, Continued

Summary of BIOS Real Time Clock Functions

Code	Description
00h	Default Interrupt Handler
01h	Return Logical ID Parameters
02h	Reserved
03h	Read Device Parameters
04h	Set Device Parameters
05h-0Ah	Reserved
0Bh	Set Alarm Interrupt
0Ch	Cancel Alarm Interrupt
0Dh	Set Periodic Interrupt
0Eh	Cancel Periodic Interrupt
0Fh	Set Update-Ended Interrupt
10H	Cancel Update-Ended Interrupt
11h	Read Time and Date
12h	Write Time and Date

In this chapter

This chapter presents information on the following topics:

- Hardware Environment
 - Real Time Clock Data Definitions
 - Error Handling
 - Real Time Clock functions.
-

Hardware Environment

MC146818A real time clock

The BIOS Real Time Clock Service supports a real time clock compatible with a Motorola MC146818A. It is associated with interrupt request 8. NMIs are disabled any time the BIOS Real Time Clock service accesses system CMOS RAM (first 64 bytes of CMOS RAM). The real time clock is assumed to be battery-backed so that the time and date are maintained when the computer is powered off.

Accessing the Real Time Clock CMOS RAM

To write to the Real Time Clock CMOS RAM:

- inhibit interrupts.
- write the CMOS RAM address to which the data is to be written to I/O port 0070h.
- write the data to be written to I/O port 0071h.

To read from RTC CMOS RAM:

- inhibit interrupts.
 - write the CMOS RAM address from which the data is to be read to I/O Port 0070h.
 - read from I/O port 0071h.
-

RTC CMOS RAM Addresses

The following table lists the I/O port addresses used to access RTC CMOS RAM:

I/O Address	Length	Description
0070h	1 Byte	CMOS RAM address, where: Bit 7 = 1 NMI disabled Bits 6-0 = 0 CMOS RAM address
0071h	1 Byte	CMOS RAM data port

Real Time Clock Data

Real time clock data definitions

Real time clock information uses CMOS RAM addresses 00h-0Eh. These data definitions are presented below in offset order.

Location	Size	Description
00h	1 Byte	Current second in binary coded decimal (BCD)
01h	1 Byte	Second alarm in BCD
02h	1 Byte	Current minute in BCD
03h	1 Byte	Minute alarm in BCD
04h	1 Byte	Current hour in BCD
05h	1 Byte	Hour alarm in BCD
06h	1 Byte	Current day of week in BCD
07h	1 Byte	Current date in BCD
08h	1 Byte	Current month in BCD
09h	1 Byte	Current year in BCD
0Ah	1 Byte	Status register A, where: Bit 7 = 1 Update in progress Bits 6-4 = Divider identifying the time-based frequency to use Bits 3-0 = Rate selection bits that define output frequency and periodic interrupt rate
0Bh	1 Byte	Status register B, where: Bit 7 = 0 Run = 1 Halt Bit 6 = 1 Enable periodic interrupt Bit 5 = 1 Enable alarm interrupt Bit 4 = 1 Enable update-ended interrupt Bit 3 = 1 Enable square wave interrupt Bit 2 = 1 Calendar is in binary format = 0 Calendar is in BCD format Bit 1 = 1 24-hour mode = 0 12-hour mode Bit 0 = 1 Enable Daylight Savings Time
0Ch	1 Byte	Status register C, where: Bits 7-4 = IRQF, PF, AF, and UF flags, respectively Bits 3-0 = Reserved
0Dh	1 Byte	Status register D, where: Bit 7 = 1 Real time clock has power Bits 6-0 = Reserved

continued

Real Time Clock Data, Continued

Real time clock data definitions, cont'd

Location	Size	Description
0Eh	1 Byte	Diagnostic status, where: Bit 7 = 1 Real time clock lost power Bit 6 = 1 CMOS RAM checksum is bad Bit 5 = 1 Invalid configuration information found at POST Bit 4 = 1 Memory size compare error at POST Bit 3 = 1 Fixed disk or adapter fails initialization Bit 2 = 1 CMOS RAM time found invalid Bit 1 = 1 Adapters do not match configuration Bit 0 = 1 Time-out reading an adapter ID

Error Handling

How errors are reported

ABIOS signals the status (Successful, Resume Stage after Interrupt, etc.) of each ABIOS request by returning a one word Return Code at offset 0Ch in the Request Block.

If Bit 15 of the Return Code field is set, the Real Time Clock Service function requested has an error. The caller's Return Code handler routine should then test Bits 14, 13, 12, and 8 to determine the class of error that has occurred. The return code handler routine should then test the remaining bits to determine the precise nature of the error.

Function: 00h — Default Interrupt Handler

Description

This function is a single-staged or multistaged request that handles unexpected hardware interrupts by resetting the interrupt at the device level.

When to invoke

This function is invoked by calling the interrupt routine with a function code of 0000h. It is only invoked if a given Logical ID has no outstanding Request Blocks waiting for an interrupt.

Request Block structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length (10h)	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0000h)	
08h	Word	Reserved (initialize to 0000h)	
0Ah	Word	Reserved (initialize to 0000h)	
0Ch	Word	Return Code	Return Code
0Eh	Word		Time-out

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0005h	Not My Interrupt, Resume Stage after Interrupt
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 01h — Return Logical ID Parameters

Description

This function is a single-staged request that returns information about the specified Logical ID.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length (20h)		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0001h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Byte			Hardware Interrupt Level (08h)
11h	Byte			Arbitration Level (FFh)
12h	Word		Device ID (0008h)	
14h	Word		Count of Units (0000h)	
16h	Word		Logical ID Flags (0000h) Bits 15-4 = Reserved Bit 3 = 0 No overlap across units Bit 2 = 0 Reserved Bits 1-0 = Transfer Data Pointer Mode 00 = No Pointers Required	
18h	Word		Request Block Length (for other functions)	
1Ah	Byte		Reserved (Initialize to 0000h)	Secondary Device ID
1Bh	Byte		Revision Level	
1Ch	Word	Reserved (Initialize to 0000h)		
1Eh	Word	Reserved (Initialize to 0000h)		

continued

Function: 01h — Return Logical ID Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 02h — Reserved

Function: 03h — Read Device Parameters

Description

This function is a multistaged request that returns the most recent interrupt settings and Real Time Clock Status for this device in the Request Block. NMIs are disabled when ABIOS accesses system CMOS RAM.

Real Time Clock processing

The Periodic Interrupt Rate field is valid only if the periodic interrupt is enabled; the Alarm fields are valid only if the alarm interrupt is enabled.

ABIOS test for update-in-progress

If the Return Code is 8000h Device in Use or 8001h Real Time Clock Not Started the real time clock is halted. This request is not processed until the real time clock is set.

continued

Function: 03h — Read Device Parameters, Continued

Request Block structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0003h)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		
10h	Byte		
11h	Byte		
12h	Byte	Function	Return Code
13h	Byte	Reserved (Initialize to 0000h)	Time-out
14h	Byte	Reserved (Initialize to 0000h)	Periodic Interrupt Rate, Bits 7-4= Reserved Bits 3-0= Rate value set 0000b = None 0001b = 30.517 ms 0010b = 61.035 ms 0011b = 122.07 ms 0100b = 244.141 ms 0101b = 488.281 ms 0110b = 976.562 ms 0111b = 1.953125 ms 1000b = 3.90625 ms 1001b = 7.8125 ms 1010b = 15.625 ms 1011b = 31.250 ms 1100b = 62.500 ms 1101b = 125.00 ms 1110b = 250.00 ms 1111b = 500.00 ms
16h	Word	Reserved (Initialize to 0000h)	Real Time Clock Status Byte Bit 7 = Set bit status 0 Clock started 1 Clock not started Bit 6 = Periodic interrupt bit 0 Interrupt disabled 1 Interrupt enabled Bit 5 = Alarm interrupt bit 0 Interrupt disabled 1 Interrupt enabled Bit 4 = Update-ended interrupt bit 0 Interrupt disabled 1 Interrupt enabled Bits 3-2 = Reserved Bit 1 = Clock mode 0 12-hour clock 1 24-hour clock
			Alarm Hour in BCD (00-23)
			Alarm Minute in BCD (00-59)
			Alarm Seconds in BCD (00-59)

continued

Function: 03h — Read Device Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation If the Alarm Interrupt is disabled, a Return Code of 0000h is generated.
8000h	Device in Use
8001h	Real Time Clock Not Started If the real time clock is not started, this code is generated and no other fields are valid.
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 04h — Set Device Parameters

Description

This function is a single-staged request that sets the Real Time Clock Hour Modes (either 12-hour clock or 24-hour clock), and the Daylight Savings Update field (either enabled or disabled). If the RTC is not functioning, BIOS does not perform this function and generates Return Code 8001h, RTC Not Started. BIOS disables NMIs when accessing system CMOS RAM.

Request Block structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0004h)		
08h	Word	Reserved (initialize to 0000h)		
0Ah	Word	Reserved (initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
16h	Word	Reserved (initialize to 0000h)		
19h	Byte	Real Time Clock Mode, where: Bits 7-2 = Reserved Bit 1 = Hour mode 0 12-hour clock 1 24-hour clock Bit 0 = Daylight Savings Update, where: 0 = Enabled 1 = Disabled		

continued

Function: 04h — Set Device Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Functions: 05h - 0Ah — Reserved

Function: 0Bh — Set Alarm Interrupt

Description

This function is a multistaged request that sets the time for an alarm interrupt. If the RTC is not functioning, ABIOS does not perform this function and generates Return Code 8001h RTC Not Started. ABIOS disables NMIs when accessing system CMOS RAM.

ABIOS test for update-in-progress

If the Return Code is 8000h Device in Use or 8001h Real Time Clock Not Started the real time clock alarm is halted. This request is not processed until the real time clock alarm is set.

Real time clock update cycle

In the IBM PS/2 BIOS, if the real time clock is in an update cycle when this function is requested, Return Code 8000h, Device Busy, Request Refused, is generated. The Phoenix BIOS aborts the clock update cycle and performs the function in this case.

Setting the alarm

The following steps must be followed to set the alarm function:

1. If there is an outstanding Set Alarm Interrupt already enabled, function 0Ch Cancel Alarm Interrupt must be invoked before function 0Bh is invoked.
 2. Invoke function 0Bh Set Alarm Interrupt.
 3. Check the Return Code.
 4. If the Return Code is 0001h, the Interrupt Pending Status field indicates the interrupts that occurred.
 5. Query the Return Code just one time; make sure that all indicated interrupts are processed, sequentially.
-

continued

Function: 0Bh — Set Alarm Interrupt, Continued

Request Block structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (000Bh)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out
12h	1 Byte	Alarm Hour in BCD (00-23)	
13h	1 Byte	Alarm Minute in BCD (00-59)	
14h	1 Byte	Alarm Seconds in BCD (00-59)	
16h	DWord	Reserved (Initialize to 0000h)	
1Ah	1 Byte		Interrupt Pending Status, where: Bit 7 = Reserved Bit 6 = 1 Periodic interrupt Bit 5 = 1 Alarm interrupt Bit 4 = 1 Update-ended interrupt Bits 3-0 = Reserved

continued

Function: 0Bh — Set Alarm Interrupt, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation Never returned by this function. If it is present, the request for this function has not been processed. The interrupt Pending Status indicates if an Update-Ended interrupt has been set.
0001h	Resume Stage after Interrupt
0005h	Not My Interrupt, Resume Stage after Interrupt
8000h	Device in Use
8001h	Real Time Clock Not Started
8002h	Interrupt Already Enabled
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Ch — Cancel Alarm Interrupt

Description

This function is a single-staged request that disables the alarm interrupt. BIOS disables NMIs before performing this function.

Request Block structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (000Ch)	
08h	Word	Reserved (initialize to 0000h)	
0Ah	Word	Reserved (initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out
16h	Word	Reserved (initialize to 0000h)	

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Dh — Set Periodic Interrupt

Description

This function is a multistaged request that sets the interval for the periodic interrupt. If the RTC is not functioning ABIOS does not perform this function, and generates Return Code 8001h RTC Not Started. ABIOS disables NMI's when accessing system CMOS RAM.

Real time clock update cycle

In the IBM PS/2 BIOS, if the real time clock is in an update cycle when this function is requested, Return Code 8000h Device Busy Request Refused is generated. The Phoenix BIOS aborts the clock update cycle and performs the function in this case.

Setting the periodic interrupt

The following steps must be followed to set the periodic function:

1. If there is an outstanding Set Periodic Interrupt already enabled, function 0Eh Cancel Periodic Interrupt must be invoked before function 0Dh.
 2. Invoke function 0Dh Set Periodic Interrupt.
 3. Check the Return Code.
 4. If the Return Code is 0001h, the Interrupt Pending Status field indicates the interrupt that occurred.
 5. Query the Return Code just one time; make sure that all indicated interrupts are processed sequentially.
-

Request Block structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Dh)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		
0Eh	Word			
				Return Code
				Time-out

continued

Function: 0Dh — Set Periodic Interrupt, Continued

Request Block structure, cont'd

Offset	Size	Input:	Output:
10h	Byte	Periodic Interrupt Rate, Bits 7-4= Reserved Bits 3-0= Rate value set 0000b = None 0001b = 30.517 ms 0010b = 61.035 ms 0011b = 122.07 ms 0100b = 244.141 " 0101b = 488.281 " 0110b = 976.562 " 0111b = 1.953125 ms 1000b = 3.90625 ms 1001b = 7.8125 ms 1010b = 15.625 ms 1011b = 31.250 ms 1100b = 62.500 ms 1101b = 125.00 ms 1110b = 250.00 ms 1111b = 500.00 ms	
16h	DWord	Reserved (Initialize to 0000h)	
1Ah	Byte		Interrupt Pending Status, where Bit 7 = Reserved Bit 6 = 1 Periodic interrupt Bit 5 = 1 Alarm interrupt Bit 4 = 1 Update-ended interrupt Bits 3-0 = Reserved

continued

Function: 0Dh — Set Periodic Interrupt, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation Never returned by this function. If it is present, the request for this function has not been processed. The Interrupt Pending Status indicates if an Update-ended interrupt has been set.
0001h	Resume Stage after Interrupt
0005h	Not My Interrupt, Resume Stage after Interrupt
8000h	Device In Use
8001h	Real Time Clock Not Started
8002h	Interrupt Already Enabled
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Eh — Cancel Periodic Interrupt

Description

This function is a single-staged request that disables the periodic interrupt. BIOS disables NMIs before performing this function.

Request Block structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (000Eh)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out
16h	Word	Reserved (Initialize to 0000h)	

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Fh — Set Update-Ended Interrupt

Description

This function is a multistaged request that enables the update-ended interrupt. The update-ended interrupt generates an interrupt that is activated every time the system clock is updated, or once per second. If the RTC is not functioning, BIOS does not perform this function and generates Return Code 8001h RTC Not Started. BIOS disables NMIs when accessing system CMOS RAM.

Real time clock update cycle

In the IBM PS/2 BIOS, if the real time clock is in an update cycle when this function is requested, Return Code 8000h Device Busy Request Refused is generated. The Phoenix BIOS aborts the clock update cycle and performs the function in this case.

Request Block cannot be changed

The Request Block field must not be changed during any intermediate stage of a request for this function.

Setting the update-ended interrupt

The following steps must be followed to set the update-ended function:

1. If there is an outstanding Set Update-Ended Interrupt already enabled, function 10h Cancel Update-Ended Interrupt must be invoked before function 0Fh is invoked.
 2. Invoke function 0Fh Set Update-Ended Interrupt.
 3. Check the Return Code.
 4. If the Return Code is 0001h, the Interrupt Pending Status field indicates the interrupt that occurred.
 5. Query the Return Code just one time; make sure that all indicated interrupts are processed sequentially.
-

continued

Function: 0Fh — Set Update-Ended Interrupt, Continued

Request Block structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Fh)		
08h	Word	Reserved (initialize to 0000h)		
0Ah	Word	Reserved (initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
16h	DWord	Reserved (initialize to 0000h)		
1Ah	Byte			Interrupt Pending Status, where: Bit 7 = Reserved Bit 6 = 1 Periodic interrupt Bit 5 = 1 Alarm interrupt Bit 4 = 1 Update-ended Interrupt Bits 3-0 = Reserved

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation Never returned by this function. If it is present, the request for this function has not been processed. The Interrupt Pending Status indicates if an update-ended interrupt has been set.
0001h	Resume Stage after Interrupt
0005h	Not My Interrupt, Resume Stage after Interrupt
8000h	Device In Use
8001h	Real Time Clock Not Started
8002h	Interrupt Already Enabled
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 10h — Cancel Update-Ended Interrupt

Description

This function is a single-staged request that disables the Update-Ended Interrupt. BIOS disables NMLs before performing this function.

Request Block structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0010h)	
08h	Word	Reserved (initialize to 0000h)	
0Ah	Word	Reserved (initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out
16h	Word	Reserved (initialize to 0000h)	

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 11h — Read Time and Date

Description

This function is a multistaged request that reads the current setting of the real time clock. The output Time and Date fields are only valid if the Return Code is 0000h Successful Operation. BIOS disables NMIs before performing this function. If the RTC is not functioning, BIOS does not perform this function and generates Return Code 8001h RTC Not Started.

Note: Function 12h Write Date and Time must be called before this function is invoked.

BIOS test for update-in-progress

If the Return Code is 8000h Device in Use or 8001h Real Time Clock Not Started the real time clock is halted. This request is not processed until the real time clock is set.

Request Block structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0011h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	Word	Reserved (Initialize to 0000h)		
12h	Byte		Alarm Hour in BCD (00-23)	
13h	Byte		Alarm Minute in BCD (00-59)	
14h	Byte		Alarm second in BCD (00-59)	
15h	Byte		Century in BCD, where: 0 = 20th century 1 = 21st century	
16h	Byte		Year in BCD (00-99)	
17h	Byte		Month in BCD (01-12)	
18h	Byte		Day in BCD (01-31)	

continued

Function: 11h — Read Time and Date, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8000h	Device In Use
8001h	Real Time Clock Not Started
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 12h — Write Time and Date

Description

This function is a single-staged request that starts the real time clock (if it is not already started) and sets the time and date information as specified. If the clock is already started, the clock update cycle is aborted, the new time and date are set, and a new clock update cycle is started. BIOS disables NMIs before performing this function. If the RTC is not functioning, BIOS does not perform this function and generates Return Code 8001h RTC Not Started.

Real time clock update cycle

In the IBM PS/2 BIOS, if the real time clock is in an update cycle when this function is requested, Return Code 8000h Device Busy Request Refused is generated. The Phoenix BIOS aborts the clock update cycle and performs the function in this case.

continued

Function: 12h – Write Time and Date, Continued

Request Block structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0012h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	Word	Reserved (Initialize to 0000h)		
12h	Byte	Hour in BCD (00-23)		
13h	Byte	Minute in BCD (00-59)		
14h	Byte	Second in BCD (00-59)		
15h	Byte	Century in BCD, where: 0 = 20th century 1 = 21st century		
16h	Byte	Year in BCD (00-99)		
17h	Byte	Month in BCD (01-12)		
18h	Byte	Day in BCD (01-31)		

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Chapter 16

BIOS System Services

Overview

Introduction

The BIOS System Services provide functions that allow address mode switching, enable the system speaker, read system configuration information, and enable/disable Address Line 20.

Summary of System Services functions

Function	Description
01h	Return Logical ID Parameters
02h	Reserved
03h	Read System Configuration
04h-0Ah	Reserved
0Bh	Switch to Real Mode
0Ch	Switch to Protected Mode
0Dh	Enable Address Line 20
0Eh	Disable Address Line 20
0Fh	Enable Speaker

continued

Overview, Continued

In this chapter

In this chapter, the following topics are discussed:

- Error Handling
 - BIOS System Services functions
-

Error Handling

How errors are reported

BIOS signals the status (Successful, Resume Stage after Interrupt, etc.) of each BIOS request by returning a one word Return Code at offset 0Ch in the Request Block.

If Bit 15 of the Return Code field is set, the System Services function requested has an error. The caller's Return Code handler routine should then test Bits 14, 13, 12, and 8 to determine the class of error that has occurred and then test the remaining bits to determine the nature of the error.

Function: 01h — Return Logical ID Parameters

Description

This function is a single-staged request that returns information about the specified Logical ID.

Request Block structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length (20h)		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0001h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Byte		Hardware Interrupt Level (FFh)	
11h	Byte		Arbitration Level (FFh)	
12h	Word		Device ID (0009h)	
14h	Word		Count of Units (0000h)	
16h	Word			Logical ID flags (0000h)
				Bits 15–4 = Reserved Bit 3 = 0 No overlap across units = 1 Bit 2 = 0 Reserved Bits 1–0 = Transfer Data Pointer Mode 00 No Pointers Required
18h	Word		Request Block Length (for other functions)	
1Ah	Byte	Reserved (Initialize to 0000h)	Secondary Device ID	
1Bh	Byte		Revision Level	
1Ch	Word	Reserved (Initialize to 0000h)		
1Eh	Word	Reserved (Initialize to 0000h)		

continued

Function: 01h — Return Logical ID Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0005h	Not My Interrupt, Resume Stage after Interrupt
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 03h — Read System Configuration

Description

This function is a single-staged request that returns information about the configuration of this system.

Processing

ABIOS disables all interrupts, including NMIs while processing this request.

Identifying PS/2 models

The following information about PS/2 model and submodel byte information is stored in the BIOS ROM data area (the motherboard ID can be read from the POS registers). These areas can be accessed from CBIOS.

This information identifies various PS/2 models. It should not be used to determine processor type, CMOS RAM size, the number of adapter slots in a system, memory register use, or other system-specific data. The Feature Configuration Table contains the above data.

Model	Submodel	Motherboard ID	Description
F8h	00h	FEFFh	Model 80
F8h	09h	FDFFh	Model 70
FCh	05h	F7FFh	Model 60
FCh	04h	FBFFh	Model 50
FAh	00h	N/A	Model 30
FAh	01h	N/A	Model 25
FFh	FFh	Any Other Value	Unknown System Board

continued

Function: 03h — Read System Configuration, Continued

Request Block structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0003h)	
08h	Word	Reserved (initialize to 0000h)	
0Ah	Word	Reserved (initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		
66h	Word		Base Memory in 1K Blocks
68h	Word		Expansion Memory in 1K Blocks
6Ah	Byte		POS Slot 5 Configuration Byte for Channel 3
6Bh	Byte		Copy of System Board POS 2 Slot
6Ch	Byte		Model Byte — set to FFh if the Adapter ID is an unknown type.
6Dh	Byte		Submodel Byte — set to FFh if the Adapter ID is an unknown type.

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Functions: 04h – 0Ah — Reserved

Function: 0Bh — Switch to Real Mode

Description

This function is a single-staged or multistaged request that switches processing to the real mode of the 80286, 80386SX, or 80386 microprocessor and disables Address Line 20 to the microprocessor, which effectively denies access to any address above 1 MB.

ROM BIOS Data Area pointer

Data Pointer 0 in the real mode Common Data Area points to the ROM BIOS Data Area.

Interrupts disabled during processing

While processing this function, the BIOS disables interrupts, including the NMI I/O Channel Check and Parity Check. The NMI Watchdog Timer Time-out and DMA Arbitration Bus Time-out cannot be disabled.

Successful completion

If all stages of the request are completed, the address mode of the processor is switched to real mode and out of BIOS. The caller should have set AH to a nonzero value before calling this function. BIOS clears AH when this function is successfully completed. Upon successful completion of this function, control is returned to the caller at the location pointed to by the Resume Pointer Field.

If AH is still a nonzero value, this function request has not been successfully completed.

continued

Function: 0Bh — Switch to Real Mode, Continued

Request between stages

If the request is between stages, a Return Code of 0005h is present and none of the Request Block parameters are changed yet.

Request unsuccessful

If there is an invalid parameter in the request for this function (Return Code C00xh), BIOS returns to the caller with the appropriate Return Code set. The caller must then process the error and request the function again.

It is the caller's responsibility to re-enable interrupts, including the NMI, upon return, either successful or unsuccessful, from a request for this function.

Request Block structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Bh)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	DWord	Reserved (Initialized to 0000h)		
60h	DWord	Resume Pointer		
64h	Word	Selector to a Dummy Descriptor Field		

continued

Function: 0Bh — Switch to Real Mode, Continued

Error handling

This function does not update the Return Code Field unless there is an error in one of the input parameters. If there is an error in one of the input parameters, the BIOS returns with all registers preserved. Otherwise, this function jumps to the address that the Resume Pointer points to with all registers changed and only CS:IP and AH containing meaningful information.

Set AH to a nonzero value before calling this function and then test AH for zero. A nonzero value indicates an error condition. See the System Services chapter in *CBIOS for PS/2 Computers and Compatibles* for more information on how to handle the error condition indicated in AH.

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0005h	Not My Interrupt, Resume Stage after Interrupt
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Ch — Switch to Protected Mode

Description

This function is a single-staged request that switches processing to the protected mode of the 80286, 80386SX, or 80386 microprocessor and enables Address Line 20 to the microprocessor.

Interrupts disabled during processing

While processing this function, the BIOS disables interrupts, including the NMI I/O Channel Check and Parity Check.

The NMI Watchdog Timer Time-out and DMA Arbitration Bus Time-out cannot be disabled.

Caller responsibilities

The caller must have loaded the Global Descriptor Table and the Local Descriptor Table referred to in the Request Block before calling BIOS. Selector 20h in the descriptor table references the caller's code.

continued

Function: 0Ch — Switch to Protected Mode, Continued

Function completion

Upon completion of this function, control is returned to the caller at the location pointed to by the Resume Pointer Field, with BIOS having changed the contents of CS:IP to point to a protected mode location instead of a real mode location.

Note: It is the caller's responsibility to re-enable interrupts, including the NMI, upon return from this function.

Request Block structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Ch)		
08h	Word	Reserved (initialize to 0000h)		
0Ah	Word	Reserved (initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	DWord	Reserved (initialized to 0000h)		
48h	Word			Segment Descriptor Table Limit
4Ah	Word		Segment Descriptor Table Base — (Bits 0-15)	
4Ch	Byte		Segment Descriptor Table Base — (Bits 16-23)	
4Dh	Byte		Segment Descriptor Table Access Rights Byte	
4Eh	Word		Segment Descriptor Table Reserved Bytes	
4Fh	17 Bytes		Reserved (do not initialize)	
60h	DWord	Resume Pointer		
64h	15 Bytes	Reserved (do not initialize)		

continued

Function: 0Ch — Switch to Protected Mode, Continued

Error handling

This function does not update the Return Code Field unless there is an error in one of the input parameters. If there is an error in one of the input parameters, the BIOS returns with all registers preserved. Otherwise, this function jumps to the protected mode address pointed to by the Resume Pointer.

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Dh — Enable Address Line 20

Description

This function is a single-staged request that enables Address Line 20 to the microprocessor, which permits access to any valid address above 1 MB.

Request Block structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (000Dh)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out
10h	DWord	Reserved (initialized to 0000h)	

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Eh — Disable Address Line 20

Description

This function is a single-staged request that disables Address Line 20 to the microprocessor, denying access to any address above 1 MB.

Request Block structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (000Eh)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out
10h	DWord	Reserved (Initialized to 0000h)	

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Fh — Enable Speaker

Description

This function is a single-staged request that enables the system speaker at the specified frequency for the specified duration upon successful completion of this function.

The system speaker is enabled by programming Mode 3 of System Timer Channel 2.

Request Block structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (000Fh)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out
10h	DWord	Reserved (Initialize to 0000h)	
60h	Word	Frequency Divisor (1.19 MHz divided by the frequency = the frequency divisor). Ex: The frequency divisor is 1331, for a frequency of 886 Hz.	
66h	Byte	Duration Counter in sixty-fourths of a second	

continued

Function: 0Fh — Enable Speaker, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation If either the input frequency or duration is zero, the function is not performed and the Return Code is 0000h.
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Chapter 17

ABIOS Nonmaskable Interrupt (NMI) Service

Overview

Introduction

The ABIOS Nonmaskable Interrupt Service clears a nonmaskable interrupt at the NMI source, which is usually at the device level.

Note: This service clears the source of the interrupt after the source of the NMI is logged, but does not re-enable NMIs. The caller must reenable NMIs.

NMI processing

A Nonmaskable Interrupt (NMI) occurs because an error condition exists somewhere in the system.

continued

Overview, Continued

Types of NMI

The types of NMI are:

- System Board Memory Parity Check
 - Adapter Card (I/O) Channel Check
 - Watchdog Timer Time-out
 - DMA Arbitration Bus Time-out
-

Summary of Nonmaskable Interrupt Service functions

Function	Description
01h	Return Logical ID Parameters
02h-05h	Reserved
06h	Enable NMI
07h	Disable NMI
08h	NMI Continuous Read

In this chapter

This chapter includes information about the following topics:

- Error Handling
 - BIOS Nonmaskable Interrupt Service functions
-

Error Handling

How errors are reported

ABIOS signals the status (Successful, Resume Stage after Interrupt, etc.) of each ABIOS request by returning a one word Return Code at offset 0Ch in the Request Block.

If Bit 15 of the Return Code field is set, the Nonmaskable Interrupt Service function requested has an error. The caller's Return Code handler routine should test Bits 14, 13, 12, and 8 for the class of error that has occurred and then test the remaining bits to determine the nature of the error.

Function: 01h — Return Logical ID Parameters

Description

This function is a single-staged request that returns the parameters for the specified Logical ID.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length (20h)		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0001h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Hardware Interrupt Level (FEh)
10h	Byte			Arbitration Level (FFh)
11h	Byte			Device ID (0009h)
12h	Word		Count of Units (0000h)	
14h	Word		Logical ID flags (0000h)	
16h	Word		Bits 15-4 = Reserved Bit 3 = 0 No overlap across units = 1 Bit 2 = 0 Reserved Bits 1-0 = Transfer Data Pointer Mode 00 No Pointers Required	
18h	Word		Request Block Length (for other functions)	
1Ah	Byte	Reserved (Initialize to 0000h)	Secondary Device ID	
1Bh	Byte		Revision Level	
1Ch	Word	Reserved (Initialize to 0000h)		
1Eh	Word	Reserved (Initialize to 0000h)		

continued

Function: 01h — Return Logical ID Parameters, Continued

Return Code

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Functions: 02h – 05h — Reserved

Function: 06h — Enable NMI

Description

This function is a single-staged request that enables the NMI for System Board Memory Parity Checks and Adapter Card I/O Channel Checks.

Note: The BIOS cannot enable the DMA Arbitration Bus Time-out and the Watchdog Timer Time-out NMIs.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0006h)	
08h	Word	Reserved (initialize to 0000h)	
0Ah	Word	Reserved (initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out
21h	Word	Reserved (initialize to 0000h)	

Return Code

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 07h — Disable NMI

Description

This function is a single-staged request that disables the NMI for System Board Memory Parity Checks and Adapter Card I/O Channel Checks.

Note: The BIOS cannot disable the DMA Arbitration Bus Time-out and the Watchdog Timer Time-out types of NMIs.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0007h)	
08h	Word	Reserved (initialize to 0000h)	
0Ah	Word	Reserved (initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out
21h	Word	Reserved (initialize to 0000h)	

Return Code

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 08h — NMI Continuous Read

Description

This function is a continuous multistaged request. It sets up a permanently resident Request Block which is used by the NMI hardware interrupt routine and the caller to handle NMIs.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0008h)	
08h	Word	Reserved (initialize to 0000h)	
0Ah	Word	Reserved (initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out (0000h)
10h	Word		Type of NMI 00h = Reserved 01h = Parity Check 02h = Channel Check 03h = DMA Bus Time-out 04h = Watchdog Time-out
1Eh	Byte		DMA Arbitration Level that Initiated the DMA Bus Time-out
1Fh	Byte		Slot Number that Initiated the I/O Channel Check
21h	Word		Reserved (initialize to 0000h)

continued

Function: 08h — NMI Continuous Read, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0005h	Not My Interrupt, Resume Stage after Interrupt
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Chapter 18

ABIOS Pointing Device Service

Overview

Introduction

The ABIOS Pointing Device Service provides routines to handle pointing devices. A mouse is the most commonly used pointing device, but this service also handles other devices such as trackballs and touchpads.

continued

Overview, Continued

Summary of Pointing Device Service functions

Function	Description
00h	Default Interrupt Handler
01h	Return Logical ID Parameters
02h	Reserved
03h	Read Device Parameters
04h	Reserved
05h	Reset/Initialize Pointing Device
06h	Enable Pointing Device
07h	Disable Pointing Device
08h	Pointing Device Continuous Read
09h-0Ah	Reserved
0Bh	Set Sample Rate
0Ch	Set Resolution
0Dh	Set Scaling Factor
0Eh	Read Pointing Device Identification Code

In this chapter

This chapter includes information about the following topics:

- Hardware Environment
 - Error Handling
 - BIOS Pointing Device Service functions
-

Hardware Environment

The BIOS Pointing Device Service is associated with hardware interrupt request 12. The BIOS Pointing Device Service supports a Pointing Device and a Pointing Device/Keyboard Controller such as an appropriately programmed Intel 8042 or its equivalent.

Error Handling

How errors are reported

BIOS signals the status (Successful, Resume Stage after Interrupt, etc.) of each BIOS request by returning a one word Return Code at offset 0Ch in the Request Block.

If Bit 15 of the Return Code field is set, the Pointing Device Service function requested has an error. The caller's Return Code handler routine should test Bits 14, 13, 12, and 8 to determine the class of error that has occurred and then test the remaining bits to determine the precise nature of the error.

Function: 00h — Default Interrupt Handler

Description

This function is a single-staged or multistaged request that handles unexpected hardware interrupts by resetting the interrupt at the device level.

When to invoke

This function is invoked by calling the interrupt routine with a function code of 0000h. It is only invoked if a given Logical ID has no outstanding Request Blocks waiting for an interrupt.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length (10h)	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0000h)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0005h	Not My Interrupt, Resume Stage after Interrupt
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 01h — Return Logical ID Parameters

Description

This function is a single-staged request that returns the parameters for the specified Logical ID.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length (20h)		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0001h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Hardware Interrupt Level (0Ch)
10h	Byte			Arbitration Level (FFh)
11h	Byte			Device ID (000Bh)
12h	Word		Count of Units (0000h)	
14h	Word		Logical ID flags (0000h) Bits 15-4 = Reserved Bit 3 = 0 No overlap across units Bit 2 = 0 Reserved Bits 1-0 = Transfer Data Pointer Mode 00 = No Pointers Required	
16h	Word		Request Block Length (for other functions)	
18h	Word		Secondary Device ID	
1Ah	Byte		Revision Level	
1Bh	Byte			
1Ch	Word		Reserved (Initialize to 0000h)	
1Eh	Word	Reserved (Initialize to 0000h)		

continued

Function: 01h — Return Logical ID Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0005h	Not My Interrupt, Resume Stage after Interrupt
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 03h — Read Device Parameters

Description

This function is a single-staged or multistaged request that returns the Pointing Device Status for the specified Pointing Device.

continued

Function: 03h — Read Device Parameters, Continued

Request Block structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0003h)		
08h	Word	Reserved (initialized to 0000h)		
0Ah	Word	Reserved (initialized to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word		Time-out	
10h	Byte		Interface Status, where: Bits 7-6 = Reserved Bit 5 = Interface enable 0 Disabled 1 Enabled Bits 4-0 = Reserved	
11h	Byte		Data Package Size (00h-08h)	
12h	2 Bytes		Flag Word, where: Bits 15-7 = Reserved Bit 6 = Mode 0 Stream mode 1 Remote/poll mode Bit 5 = Status 0 Disabled 1 Enabled Bit 4 = Scaling 0 1:1 Scaling 1 2:1 Scaling Bit 3 = Reserved Bit 2 = Left button pressed Bit 1 = Reserved Bit 0 = Right button pressed	
14h	Word		Current Resolution, where: 00h = 1 count per mm 01h = 2 counts per mm 02h = 4 counts per mm 03h = 8 counts per mm	
16h	Word		Current Sample Rate, where: 0Ah = 10 reports/second 14h = 20 reports/second 28h = 40 reports/second 3Ch = 60 reports/second 50h = 80 reports/second 64h = 100 reports/second C8h = 200 reports/second	
18h	DWord		Time to Wait (in microseconds) Before Resuming Requests	
1Ch	Word		Reserved (initialize to 0000h)	

continued

Function: 03h — Read Device Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0005h	Not My Interrupt, Resume Stage after Interrupt
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 04h — Reserved

Function: 05h — Reset/Initialize Pointing Device

Description

This function is a single-staged or multistaged request that resets and initializes a Pointing Device.

Pointing Device initialized state

When the pointing device is initialized,

- data package size is not changed,
 - resolution is set to 4 counts per millimeter,
 - sample rate is set to 100 reports per second,
 - scaling factor is set to 1:1, and
 - the pointing device is disabled.
-

Using this function

The steps below must be followed before this function can be used:

1. invoke function 08h Continuous Read,
 2. invoke function 06h Enable Pointing Device,
 3. invoke function 05h Reset/Initialize Pointing Device, and
 4. invoke functions 0Bh, 0Ch, 0Dh, or 0Eh, as needed.
-

continued

Function: 05h — Reset/Initialize Pointing Device, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0005h)		
08h	Word	Reserved (Initialized to 0000h)		
0Ah	Word	Reserved (Initialized to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	Byte			Pointing Device Completion Code
11h	Byte			Pointing Device Identification Code
18h	Word		Time to Wait (in microseconds) Before Continuing Request	
1Ch	Word	Reserved (Initialize to 0000h)		

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
0005h	Not My Interrupt, Resume Stage after Interrupt
8000h	Device In Use
8001h	Resend
8002h	Two Consecutive Resends Found
8003h	System Lock
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 06h — Enable Pointing Device

Description

This function is a single-staged or multistaged request that enables the Pointing Device.

Using this function

The steps below must be followed before this function can be used:

1. invoke function 08h Continuous Read,
 2. invoke function 06h Enable Pointing Device,
 3. invoke function 05h Reset/Initialize Pointing Device, and
 4. invoke functions 0Bh, 0Ch, 0Dh, or 0Eh, as needed.
-

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0006h)	
08h	Word	Reserved (initialized to 0000h)	
0Ah	Word	Reserved (initialized to 0000h)	
0Ch	Word	Return Code	Return Code
0Eh	Word		Time-out
18h	4 Bytes		
1Ch	Word	Reserved (initialize to 0000h)	

continued

Function: 06h — Enable Pointing Device, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
0005h	Not My Interrupt, Resume Stage after Interrupt
8000h	Device in Use
8001h	Resend
8002h	Two Consecutive Resends Found
8003h	System Lock
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 07h — Disable Pointing Device

Description

This function is a single-staged or multistaged request that disables a pointing device.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0007h)	
08h	Word	Reserved (Initialized to 0000h)	
0Ah	Word	Reserved (Initialized to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out
18h	DWord		Time to Wait (In microseconds) Before Continuing Request
1Ch	Word	Reserved (Initialize to 0000h)	

continued

Function: 07h — Disable Pointing Device, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
0005h	Not My Interrupt, Resume Stage after Interrupt
8000h	Device in Use
8001h	Resend
8002h	Two Consecutive Resends Found
8003h	System Lock
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 08h — Pointing Device Continuous Read

Description

This function is a continuous multistaged request that reads a pointing device while the device remains disabled.

When to use

This function must be invoked before invoking any other Pointing Device Service function except function 01h.

Using this function

The steps below must be followed before this function can be used:

1. invoke function 08h Continuous Read,
 2. invoke function 06h Enable Pointing Device,
 3. invoke function 05h Reset/Initialize Pointing Device, and
 4. invoke functions 0Bh, 0Ch, 0Dh, or 0Eh, as needed.
-

continued

Function: 08h — Pointing Device Continuous Read, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0008h)		
08h	Word	Reserved (initialized to 0000h)		
0Ah	Word	Reserved (initialized to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Byte	Data Package Size, where: 08h = 8 Bytes 00h = Reserved 01h = 1 Byte 02h = 2 Bytes 03h = 3 Bytes 04h = 4 Bytes 05h = 5 Bytes 06h = 6 Bytes 07h = 7 Bytes		
12h	4	Reserved (initialize to 0000h)		
1Ch	Bytes		Pointing Device Data, where: BYTE 1 Bit 7 = 0 No Y Data Overflow = 1 Y Data Overflow Bit 6 = 0 No X Data Overflow = 1 X Data Overflow Bit 5 = 0 Y Data Sign is Positive = 1 Y Data sign is Negative Bit 4 = 0 X Data Sign is Positive = 1 X Data Sign is Negative Bit 3 = 1 Reserved Bit 2 = 0 Reserved Bit 1 = 1 Right Button Pressed Bit 0 = 1 Left Button Pressed BYTE 2 = Pointing Device X Data BYTE 3 = Pointing Device Y Data BYTES 4-12 = Reserved Note: X and Y refer to standard X/Y grid coordinates.	

continued

Function: 08h — Pointing Device Continuous Read, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
0005h	Not My Interrupt, Resume Stage after Interrupt
0009h	Attention, Data Available, Resume Stage after Interrupt
8000h	Device In Use
8003h	System Lock
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Mouse Controller Parameter
FFFFh	Return Code Field Not Valid

Functions: 09h – 0Ah — Reserved

Function: 0Bh — Set Sample Rate

Description

This function is a single-staged or multistaged request that sets the sample rate for a pointing device.

Using this function

Invoke the following functions in the order given to use this function:

1. Function 08h Continuous Read
2. Function 06h Enable Pointing Device
3. Function 05h Reset/Initialize Pointing Device
4. Function 0Bh Set Sample Rate

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function(000Bh)	
08h	Word	Reserved (initialized to 0000h)	
0Ah	Word	Reserved (initialized to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out
12h	Word	Sample Rate, where: 0Ah = 10 reports/second 14h = 20 reports/second 28h = 40 reports/second 3Ch = 60 reports/second 50h = 80 reports/second 64h = 100 reports/second C8h = 200 reports/second	
18h	DWord		Time to Wait (In microseconds) Before Continuing Request

continued

Function: 0Bh — Set Sample Rate, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
0005h	Not My Interrupt, Resume Stage after Interrupt
8000h	Device in Use
8001h	Resend
8002h	Two Consecutive Resends Found
8003h	System Lock
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Ch — Set Resolution

Description

This function is a single-staged request that sets the resolution for a pointing device by specifying the desired resolution rate in the Sample Rate field at offset 12h in the Request Block.

Using this function

Invoke the following functions in the order given to use this function:

1. Function 08h Continuous Read
2. Function 06h Enable Pointing Device
3. Function 05h Reset/Initialize Pointing Device
4. Function 0Ch Set Resolution

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (000Ch)	
08h	Word	Reserved (initialized to 0000h)	
0Ah	Word	Reserved (initialized to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out
12h	Word	Sample Rate, where: 00h = 1 count per mm 01h = 2 counts per mm 02h = 4 counts per mm 03h = 8 counts per mm	
18h	DWord		Time to Wait (in microseconds) Before Continuing Request
1Ch	Word	Reserved (initialize to 0000h)	

continued

Function: 0Ch – Set Resolution, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
0005h	Not My Interrupt, Resume Stage after Interrupt
8000h	Device in Use
8001h	Resend
8002h	Two Consecutive Resends Found
8003h	System Lock
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Dh — Set Scaling Factor

Description

This function is a single-staged or multistaged request that sets the Scaling Factor for a pointing device by specifying the desired value in the Scaling Factor field at offset 10h in the Request Block.

Using this function

Invoke the following functions in the order given to use this function:

1. Function 08h Continuous Read,
 2. Function 06h Enable Pointing Device,
 3. Function 05h Reset/Initialize Pointing Device, and
 4. Function 0Dh Set Scaling Factor.
-

continued

Function: 0Dh — Set Scaling Factor, continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Dh)		
08h	Word	Reserved (Initialized to 0000h)		
0Ah	Word	Reserved (Initialized to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out (Not valid unless Return Code is 0002h).
10h	Byte	Scaling Factor, where: 00h = Reserved 01h = Set to 1:1 02h = Set to 2:1		
18h	DWord		Time to Wait (In microseconds) Before Continuing Request	
1Ch	Word	Reserved (Initialize to 0000h)		
28h	Byte		Retry Mode Enable (bit 5 on)	
29h	Byte		Mouse Data Buffer Index	
2Ah	Word		Retry Parameters	
2Ch	Word		Reserved	
2Eh	Byte		Count of Status Bytes in RB	
2Fh	Byte		Current Interrupt Level (09h)	
30h	Byte		Arbitration Level (0Ch)	
31h	Word		Device ID	
33h	Word		Keyboard Data Register (60h)	
35h	Word		8042 Status Register (64h)	
37h	Byte		Reserved	
38h	Byte		Pointer to Beginning of Request Block Status Area (2Eh)	

continued

Function: 0Dh — Set Scaling Factor, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
0005h	Not My Interrupt, Resume Stage after Interrupt
8000h	Device In Use
8001h	Resend
8002h	Two Consecutive Resends Found
8003h	System Lock
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Eh — Read Pointing Device Identification Code

Description

This function is a single-staged or multistaged request that returns the Pointing Device Identification Code from the pointing device/keyboard interface.

Using this function

Invoke the following functions in the order given to use this function:

1. Function 08h Continuous Read,
 2. Function 06h Enable Pointing Device,
 3. Function 05h Reset/Initialize Pointing Device, and
 4. Function 0Eh Read Pointing Device Identification Code.
-

continued

Function: 0Eh — Read Pointing Device Identification Code,

Continued

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (000Eh)	
08h	Word	Reserved (initialized to 0000h)	
0Ah	Word	Reserved (initialized to 0000h)	
0Ch	Word	Return Code	Return Code
0Eh	Word		Time-out
10h	Byte	Auxiliary Device Identification Code	Not valid unless Return Code is 0002h.
18h	DWord		Time to Wait (in microseconds) Before Continuing Request
1Ch	Word	Reserved (initialize to 0000h)	
28h	Byte		Retry Mode Enable (bit 5 on)
29h	Byte		Mouse Data Buffer Index
2Ah	Word		Retry Parameters
2Ch	Word		Reserved
2Eh	Byte		Count of Status Bytes in RB
2Fh	Byte		Current Interrupt Level (09h)
30h	Byte		Arbitration Level (0Ch)
31h	Word		Device ID
33h	Word		Keyboard Data Register (60h)
35h	Word		8042 Status Register (64h)
37h	Byte		Reserved
38h	Byte		Pointer to Beginning of Request Block Status Area (2Eh)

continued

Function: 0Eh — Read Pointing Device Identification Code, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0001h	Resume Stage after Interrupt
0002h	Resume Stage after Time Delay
0005h	Not My Interrupt, Resume Stage after Interrupt
8000h	Device in Use
8001h	Resend
8002h	Two Consecutive Resends Found
8003h	System Lock
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Chapter 19

ABIOS CMOS RAM Service

Overview

Introduction

The ABIOS CMOS RAM service provides access to battery-backed CMOS memory. In addition, this service provides error checking through the Recompute Checksum function.

Summary of ABIOS CMOS RAM Service functions

Function	Description
01h	Return Logical ID Parameters
02h	Reserved
03h	Read Device Parameters
04h-07h	Reserved
08h	Read CMOS RAM
09h	Write to CMOS RAM
0Ah	Reserved
0Bh	Recompute Checksum

continued

Overview, Continued

In this chapter

This chapter includes information about the following topics:

- Hardware Environment
 - CMOS RAM Data Definitions
 - Extended CMOS RAM Data Definitions
 - Error Handling
 - BIOS CMOS RAM Service functions
-

Hardware Environment

Introduction

Information may be stored in two areas of CMOS RAM. The table below describes the CMOS RAM areas.

Table of CMOS RAM areas

Data Area	I/O Location	Size	Description
CMOS RAM Data Area	070h & 071h	64 Bytes	These bytes are located on the Motorola MC146818A Real Time Clock CMOS chip (or its equivalent). All implementations of the BIOS make use of this area to store real time clock, POST, and system configuration data.
Extended CMOS RAM Data Area	074h, 075h, and 076h	2048 Bytes	When implemented on systems that employ more than four adapter slots, the BIOS requires an additional 2K of CMOS RAM. This extended CMOS RAM is primarily used to store POS data.

CMOS RAM Data

Introduction

The Motorola MC146818A Real Time Clock CMOS chip (or its equivalent) contains 50 bytes of CMOS RAM data. Addresses 00h–0Dh access Real Time Clock data, which are listed in Chapter 15. Addresses 0Eh–3Fh access CMOS RAM data and are documented in this chapter.

CMOS RAM Definitions

The CMOS RAM data contained in the MC146818A chip is accessed by both the ABIOS and the CBIOS. CMOS RAM definitions are listed here for the convenience of ABIOS users. This information is repeated in *CBIOS for IBM PS/2 Computers and Compatibles*.

Note: CMOS RAM locations 19h–30h are defined only for PS/2–Compatible models containing four or fewer adapter slots.

Configuration data definitions

The system configuration information data definitions use CMOS RAM addresses 0Eh–3Fh. They are:

Location	Size	Description
0Eh	1 Byte	Diagnostic status, where: Bit 7 = 1 Real time clock lost power Bit 6 = 1 CMOS RAM checksum is bad Bit 5 = 1 Invalid configuration information found at POST Bit 4 = 1 Memory size compare error at POST Bit 3 = 1 Fixed disk or adapter fails initialization Bit 2 = 1 CMOS RAM time found invalid Bit 1 = 1 Adapters do not match configuration Bit 0 = 1 Time-out reading an adapter ID
0Fh	1 Byte	Reason for shutdown, where: 00h = Power on or soft reset 01h = Memory size pass 02h = Memory test pass 03h = Memory test fail 04h = POST end; boot system 05h = JMP DWord pointer with end-of-interrupt 06h = Protected tests pass 07h = Protected tests fail 08h = Memory size fail 09h = INT 15h Block Move 0Ah = JMP DWord pointer without end-of-interrupt 0Bh = Used by ABIOS

continued

CMOS RAM Data, Continued

Configuration data definitions, cont'd

Location	Size	Description
10h	1 Byte	Type of Diskette Drives: Bits 7-4 = Drive type of drive 0, where: 0000b = No drive 0001b = 360K drive 0010b = 1.2 MB drive 0011b = 720K drive 0100b = 1.44 MB drive Bits 3-0 = Drive type of drive 1, where: 0000b = No drive 0001b = 360K drive 0010b = 1.2 MB drive 0011b = 720K drive 0100b = 1.44 MB drive
11h	1 Byte	Type of fixed disk drive 0
12h	1 Byte	Type of fixed disk drive 1
13h	1 Byte	Password Configuration Bits 7-5 = 0 Reserved Bit 4 = 0 BIOS initializes keyboard to normal speed 1 BIOS initializes keyboard to fast speed* Bits 3-2 = Reserved Bit 1 = 1 Network password installed 0 Network password not installed Bit 0 = 1 Power-on password installed 0 Power-on password not installed
14h	1 Byte	Equipment installed, where: Bits 7-6 = Number of diskette drives, where: 00b = 1 Diskette drive 01b = 2 Diskette drives Bits 5-4 = Primary display, where: 00b = Reserved 01b = VGA in 40-column mode 10b = VGA in 80-column mode 11b = VGA in monochrome mode Bits 3-2 = Reserved Bit 1 = 1 80387 installed Bit 0 = 0 Diskette drive installed
15h	1 Byte	Base memory in 1K, low byte
16h	1 Byte	Base memory in 1K, high byte
17h	1 Byte	Expansion memory in 1K, low byte
18h	1 Byte	Expansion memory in 1K, high byte
19h	1 Byte	Adapter ID for channel 0, low byte
* If the machine has extended CMOS RAM, the value for the fast speed is taken from offset 702h.		

continued

CMOS RAM Data, Continued

Configuration data definitions, cont'd

Location	Size	Description
1Ah	1 Byte	Adapter ID for channel 0, high byte
1Bh	1 Byte	Adapter ID for channel 1, low byte
1Ch	1 Byte	Adapter ID for channel 1, high byte
1Dh	1 Byte	Adapter ID for channel 2, low byte
1Eh	1 Byte	Adapter ID for channel 2, high byte
1Fh	1 Byte	Adapter ID for channel 3, low byte
20h	1 Byte	Adapter ID for channel 3, high byte
21h	1 Byte	POS 2 configuration byte for channel 0
22h	1 Byte	POS 3 configuration byte for channel 0
23h	1 Byte	POS 4 configuration byte for channel 0
24h	1 Byte	POS 5 configuration byte for channel 0
25h	1 Byte	POS 2 configuration byte for channel 1
26h	1 Byte	POS 3 configuration byte for channel 1
27h	1 Byte	POS 4 configuration byte for channel 1
28h	1 Byte	POS 5 configuration byte for channel 1
29h	1 Byte	POS 2 configuration byte for channel 2
2Ah	1 Byte	POS 3 configuration byte for channel 2
2Bh	1 Byte	POS 4 configuration byte for channel 2
2Ch	1 Byte	POS 5 configuration byte for channel 2
2Dh	1 Byte	POS 2 configuration byte for channel 3
2Eh	1 Byte	POS 3 configuration byte for channel 3
2Fh	1 Byte	POS 4 configuration byte for channel 3
30h	1 Byte	POS 5 configuration byte for channel 3
31h	1 Byte	Copy of system board POS 2
32h	1 Byte	CRC for offsets 10-31, high byte
33h	1 Byte	CRC for offsets 10-31, low byte
34h	1 Byte	Miscellaneous information Bits 7-4 = Actual number of RS-232 ports installed Bits 3-0 = Block move status before reset to real mode

continued

CMOS RAM Data, Continued

Configuration data definitions, cont'd

Location	Size	Description
35h	1 Byte	Low byte of actual expansion memory size
36h	1 Byte	High byte of actual expansion memory size
37h	1 Byte	Century in BCD
38h-3Eh	1 Byte	Power-on password
39h	1 Byte	Power-on password checksum

Extended CMOS RAM Data

Introduction

The BIOS uses an additional 2K of CMOS RAM to store Programmable Option Select (POS) data for MCA-compatible computer systems that include more than four expansion slots. Fixed disk parameter data for fixed disks other than those listed in the ROM BIOS Fixed Disk Parameter Table is also stored in this additional CMOS RAM.

Extended CMOS RAM data definitions

The table below outlines the contents of the extended CMOS RAM data area.

CMOS RAM Offset	Size	Description
0000h	1 Byte	LSB of adapter ID for channel 0
0001h	1 Byte	MSB of adapter ID for channel 0
0002h	1 Byte	Number of POS values used
0003h	1 Byte	POS 2 for channel 0
0004h	1 Byte	POS 3 for channel 0
0005h	1 Byte	POS 4 for channel 0
0006h	1 Byte	POS 5 for channel 0

continued

Extended CMOS RAM Data, Continued

Extended CMOS RAM data definitions, cont'd

CMOS RAM Offset	Size	Description
0007-0022h		Reserved
0023h	1 Byte	LSB of adapter ID for channel 1
0024h	1 Byte	MSB of adapter ID for channel 1
0025h	1 Byte	Number of POS values used
0026h	1 Byte	POS 2 for channel 1
0027h	1 Byte	POS 3 for channel 1
0028h	1 Byte	POS 4 for channel 1
0029h	1 Byte	POS 5 for channel 1
002A-0045h		Reserved
0046h	1 Byte	LSB of adapter ID for channel 2
0047h	1 Byte	MSB of adapter ID for channel 2
0048h	1 Byte	Number of POS values used
0049h	1 Byte	POS 2 for channel 2
004Ah	1 Byte	POS 3 for channel 2
004Bh	1 Byte	POS 4 for channel 2
004Ch	1 Byte	POS 5 for channel 2
004D-0068h		Reserved
0069h	1 Byte	LSB of adapter ID for channel 3
006Ah	1 Byte	MSB of adapter ID for channel 3
006Bh	1 Byte	Number of POS values used
006Ch	1 Byte	POS 2 for channel 3
006Dh	1 Byte	POS 3 for channel 3
006Eh	1 Byte	POS 4 for channel 3
006Fh	1 Byte	POS 5 for channel 3
0070-008Bh		Reserved
008Ch	1 Byte	LSB of adapter ID for channel 4
008Dh	1 Byte	MSB of adapter ID for channel 4
008Eh	1 Byte	Number of POS values used
008Fh	1 Byte	POS 2 for channel 4

continued

Extended CMOS RAM Data, Continued

Extended CMOS RAM data definitions, cont'd

CMOS RAM Offset	Size	Description
0090h	1 Byte	POS 3 for channel 4
0091h	1 Byte	POS 4 for channel 4
0092h	1 Byte	POS 5 for channel 4
0093-00AEh		Reserved
00AFh	1 Byte	LSB of adapter ID for channel 5
00B0h	1 Byte	MSB of adapter ID for channel 5
00B1h	1 Byte	Number of POS values used
00B2h	1 Byte	POS 2 for channel 5
00B3h	1 Byte	POS 3 for channel 5
00B4h	1 Byte	POS 4 for channel 5
00B5h	1 Byte	POS 5 for channel 5
00B6-00D1h		Reserved
00D2h	1 Byte	LSB of adapter ID for channel 6
00D3h	1 Byte	MSB of adapter ID for channel 6
00D4h	1 Byte	Number of POS values used
00D5h	1 Byte	POS 2 for channel 6
00D6h	1 Byte	POS 3 for channel 6
00D7h	1 Byte	POS 4 for channel 6
00D8h	1 Byte	POS 5 for channel 6
00D9-00F4h		Reserved
00F5h	1 Byte	LSB of adapter ID for channel 7
00F6h	1 Byte	MSB of adapter ID for channel 7
00F7h	1 Byte	Number of POS values used
00F8h	1 Byte	POS 2 for channel 7
00F9h	1 Byte	POS 3 for channel 7
00FAh	1 Byte	POS 4 for channel 7
00FBh	1 Byte	POS 5 for channel 7
00FC-0160h		Reserved

continued

Extended CMOS RAM Data, Continued

Extended CMOS RAM data definitions, cont'd

CMOS RAM Offset	Size	Description
0161-0162h	1 Word	Set to a value to make POST's CRC for extended CMOS RAM locations 0-162 equal zero. Maintained by the Reference Diskette.
0163-0165h	3 Bytes	Actual extended memory size when over 65 MB
0166-0175h	16 Bytes	Fixed Disk Parameter Table for drive 0
0176-0185h	16 Bytes	Fixed Disk Parameter Table for drive 1
0186h	1 Byte	POST uses this offset to test whether extended CMOS RAM can be accessed correctly
0187h-0188h		Reserved
0189h-018Dh		Reserved for reference diskette POST use
018Eh	1 Byte	Number of Micro Channel slots.
018Fh-0388h		Reserved
0389h	1 Byte	Error Log Number (0-5)
038Ah	20 Bytes	Error Log Block 0
039Eh	20 Bytes	Error Log Block 1
03B2h	20 Bytes	Error Log Block 2
03C6h	20 Bytes	Error Log Block 3
03DAh	20 Bytes	Error Log Block 4
03EEh	20 Bytes	Error Log Block 5
0402h-06FFh	20 Bytes	Reserved
0700h-07FFh	20 Bytes	Reserved

Error Handling

How errors are reported

BIOS signals the status (Successful, Resume Stage after Interrupt, etc.) of each BIOS request by returning a one word Return Code at offset 0Ch in the Request Block.

If Bit 15 of the Return Code field is set, the CMOS RAM Service function requested has an error. The caller's Return Code handler routine should test Bits 14, 13, 12, and 8 for the class of error that has occurred and then test the remaining bits to determine the nature of the error.

Error checking

The potential for error inherent in CMOS technology makes error checking advisable. Test the Return Code field after all CMOS RAM Service function requests. A defective battery causes Return Code 80FFh, CMOS RAM Battery Bad, and a checksum error causes Return Code 80FEh, CMOS RAM Checksum Invalid, but other Return Code values may occur.

Function: 01h — Return Logical ID Parameters

Description

This function is a single-staged request that returns the parameters for the specified Logical ID.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length (20h)		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0001h)		
08h	Word	Reserved (initialize to 0000h)		
0Ah	Word	Reserved (initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Byte			Hardware Interrupt Level (FFh)
11h	Byte			Arbitration Level (FFh)
12h	Word		Device ID (000Eh)	
14h	Word		Count of Units	
16h	Word		Logical ID flags Bits 15-4 = Reserved Bit 3 = 0 No overlap across units = 1 Overlap across units supported Bit 2 = 0 Reserved Bits 1-0 = Transfer Data Pointer Mode 01 = Logical Pointer Required	
18h	Word		Request Block Length (for other functions)	
1Ah	Byte	Reserved (initialize to 0000h)	Secondary Device ID	
1Bh	Byte		Revision Level	
1Ch	Word	Reserved (initialize to 0000h)		
1Eh	Word	Reserved (initialize to 0000h)		

continued

Function: 01h — Return Logical ID Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 02h — Reserved

Function: 03h — Read Device Parameters

Description

This function, a single-staged request, returns the CMOS RAM Service device parameters. Specifically, the Read Device Parameters function returns the locations of extended CMOS RAM available to the user. The remaining areas of Extended CMOS RAM and all of CMOS RAM are reserved.

continued

Function: 03h — Read Device Parameters, Continued

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0003h)	
08h	Word	Reserved (initialize to 0000h)	
0Ah	Word	Reserved (initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		
16h	Word	Reserved (initialize to 0000h)	
22h	Word		Start of User RAM
24h	Word		Length of User RAM

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
80FEh	CMOS RAM Checksum Invalid
80FFh	CMOS RAM Battery Bad
C000h	Invalid Logical ID
C001h	Invalid Function Number
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid CMOS RAM Parameter
FFFFh	Return Code Field Not Valid

Functions: 04h – 07h — Reserved

Function: 08h — Read CMOS RAM

Description

This function, a multistaged request, returns the data stored in the CMOS RAM location pointed to by Data Pointer 1 or 2. The Transfer Data Pointer Mode, which determines the Data Pointer fields format, is returned in function 01h, Return Logical ID Parameters function.

Note: BIOS disables nonmaskable interrupts while processing CMOS RAM accesses through functions 08h or 09h.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0008h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Word	Reserved (Initialize to 0000h)		
12h	DWord	Data Pointer 1 (Logical Pointer)		
16h	Word	Reserved (Initialize to 0000h)		
1Ah	DWord	Reserved		
20h	Word	Flag Word, where: Bit 15 = 1 NMI disabled Bits 14-1 = Reserved Bit 0 = RAM type 0 CMOS RAM 1 Extended CMOS RAM		
22h	Word	Starting Address		
24h	Word	Number of bytes to transfer: If field value = 0, no action is taken; Return Code field Operation Completed successfully is set.		

continued

Function: 08h — Read CMOS RAM, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
80FEh	CMOS RAM Checksum Invalid
80FFh	CMOS RAM Battery Bad
C000h	Invalid Logical ID
C001h	Invalid Function Number
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid CMOS RAM Parameter This field is set and no action is taken if the bytes in the Number of Bytes to Transfer field plus the byte count for the Starting RAM Address field in the Request Block is greater than the maximum amount of RAM. It could also mean that there is no Extended CMOS RAM.
FFFFh	Return Code Field Not Valid

Function: 09h — Write to CMOS RAM

Description

This function, a single-staged request, writes the number of bytes of data specified at offset 24h and pointed to by Data Pointer 1 or 2 to the CMOS RAM or extended CMOS RAM location specified at offset 22h, Starting Address. The Transfer Data Pointer Mode, which determines the Data Pointer fields format, is returned in function 01h, Return Logical ID Parameters function.

Note: BIOS disables nonmaskable interrupts while processing CMOS RAM accesses through functions 08h or 09h.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0009h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Word	Reserved (Initialize to 0000h)		
12h	DWord	Data Pointer 1 (Logical Pointer)		
16h	Word	Reserved (Initialize to 0000h)		
1Ah	DWord	Reserved		
20h	Word	Flag Word, where: Bit 15 = 1 NMI disabled Bits 14-1 = Reserved Bit 0 = RAM type 0 CMOS RAM 1 Extended CMOS RAM		
22h	Word	Starting Address		
24h	Word	Number of bytes to transfer: If field value = 0, no action is taken and the Return Code field Operation Completed Successfully is set.		

continued

Function: 09h — Write to CMOS RAM, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
80FEh	CMOS RAM Checksum Invalid
80FFh	CMOS RAM Battery Bad
C000h	Invalid Logical ID
C001h	Invalid Function Number
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid CMOS RAM Parameter This field is set and no action is taken if the bytes in the Number of Bytes to Transfer field plus the byte count for the Starting RAM Address field in the Request Block is greater than the maximum amount of RAM. It could also mean that there is no Extended CMOS RAM.
FFFFh	Return Code Field Not Valid

Function: 0Ah — Reserved

Function: 0Bh — Recompute Checksum

Description

This function recomputes the checksum for either CMOS RAM or extended CMOS RAM.

When to Use

Invoke this function following a function 08h, Read CMOS RAM or 09h, Write to CMOS RAM request.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Bh)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Word	Reserved (Initialize to 0000h)		
12h	Dword	Reserved (Initialize to 0000h)		
16h	Word	Reserved (Initialize to 0000h)		
1Ah	Dword	Reserved (Initialize to 0000h)		
20h	Word	Flag Word, where: Bit 15 = 1 for NMI disabled Bits 14-1 = Reserved Bit 0 = RAM type 0 CMOS RAM 1 Extended CMOS RAM		

continued

Function: 0Bh — Recompute Checksum, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
80FEh	CMOS RAM Checksum Invalid
80FFh	CMOS RAM Battery Bad
C000h	Invalid Logical ID
C001h	Invalid Function Number
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid CMOS RAM Parameter
FFFFh	Return Code Field Not Valid

Chapter 20

ABIOS Direct Memory Access (DMA) Service

Overview

Introduction

The ABIOS Direct Memory Access (DMA) Service allows programs to allocate arbitration levels and DMA channels, and initiate transfers through the ABIOS. It is unnecessary for the caller to program the DMA Controller directly.

DMA transfer operation steps

To complete a DMA transfer, the caller must

1. Receive an arbitration level by invoking function 0Bh, Allocate Arbitration Level.
 2. Complete the Mode Control field (offset 1Ch in the Request Block) and the Transfer Control Bytes fields (offsets 1Dh and 1Eh in the Request Block) to specify the DMA controller functions for the request.
 3. Prepare for a transfer to or from a device, using functions 10h, 11h, or 12h, by coding the appropriate input information.
 4. Disable the arbitration level (function 0Dh Disable Arbitration Level).
 5. Deallocate the arbitration level (function 0Ch Deallocate Arbitration Level).
-

continued

Overview, Continued

Summary of BIOS DMA Service functions

Function	Description
01h	Return Logical ID Parameters
02h	Reserved
03h	Read Device Parameters
04h-0Ah	Reserved
0Bh	Allocate Arbitration Level
0Ch	Deallocate Arbitration Level
0Dh	Disable Arbitration Level
0Eh	DMA Transfer Status
0Fh	Abort DMA Operation
10h	DMA Transfer from Memory to I/O
11h	Read from I/O and Write to Memory
12h	Load DMA Controller Parameters

In this chapter

The following topics are discussed in this chapter:

- Hardware Environment
 - Error Handling
 - BIOS DMA Service functions
-

Hardware Environment

DMA functionality

The hardware environment for DMA transfers is described here in order to explain the background against which the DMA BIOS functions operate. The BIOS, however, serves as a shield between underlying hardware and requests of the operating system, eliminating the need for the caller to access the DMA controller directly.

Direct Memory Access (DMA) allows large amounts of data to be transferred from a physical device to system memory or vice versa without microprocessor involvement. A program may initiate a DMA transfer and have no need to copy each byte or word individually, freeing the processor for more complex tasks. DMA transfers are typically from/to a fixed I/O port address to/from a continually incremented memory address.

DMA functionality in Micro Channel systems is a superset of the functionality of two Intel 8237 DMA Controllers, one addressed at every port, starting with Port 0000h, and one addressed at every other port, starting at port 00C0h. Access to 8237-compatible DMA functions and to additional functions *for all channels* is provided at I/O ports 0018h and 001Ah. Data output to port 0018h selects the channel and function, and data output to or input from 001Ah goes to or from the selected internal register.

Bus sharing

The system microprocessor and any currently-transferring DMA users can share the bus by taking turns directing bus cycles (driving the Micro Channel's address lines and certain control signals). An arbitration process determines which of these possible bus masters is ready to direct a cycle. Competing bus masters (DMA devices) are assigned varying priorities, which are weighed during arbitration. Each bus master gets control of the bus for a number of cycles as determined by the arbitration process.

DMA device

A DMA device (or bus master) is one that enters into arbitration for the channel. If it wins, it receives addresses and control signals from the DMA controller so it can read or write data.

continued

Hardware Environment, Continued

DMA Controller

A DMA controller is a device that monitors the arbitration process and gives addresses and control signals to the device that won the bus through arbitration. The controller does not enter into the arbitration itself. PS/2-compatible Micro Channel-based systems provide a DMA controller that supports DMA transfers to/from up to eight devices at once.

DMA hardware registers

The DMA controller maintains several hardware registers for each DMA channel. The key registers are:

- a memory address where the next byte or word will be transferred to or from,
 - a count of the remaining bytes to transfer ((transfer count),
 - a flag (mode) controlling the transfer direction (to memory or to the device), and
 - transfer status flags for each channel (status).
-

DMA hardware registers

The DMA Controller has ten sets of registers, summarized below:

Register	Size (bits)	Number of Registers	How Allocated
Memory Address	24	8	1 per Channel
I/O Address	16	8	1 Per Channel
Transfer Count	16	8	1 Per Channel
Temporary Holding	16	1	All Channels
Mask	4	2	1 for Channels 7-4 1 for Channels 3-0
Arbus	4	2	1 for Channel 4 1 for Channel 0
Mode	8	8	1 per Channel
Status	8	2	1 for Channel 7-4 1 for Channel 3-0
Function	8	1	All Channels
Refresh	9	1	Independent of DMA

continued

Hardware Environment, Continued

Mode Control Field

The Mode Control field provides an opportunity for the caller to use the Autoinitialization and Programmed I/O (PIO) features of the DMA Controller.

- **Autoinitialization**

Specifies if the DMA Controller will initialize automatically when the transfer reaches the terminal count.

- **Programmed I/O**

Specifies that the I/O address is to be programmed to the DMA Controller, driving the I/O address on the bus during the DMA cycles.

Transfer Control Bytes

These fields provide an opportunity for the caller to specify the physical address of the memory and I/O fields for BIOS DMA Service functions 10h, 11h, and 12h.

- **Count Control**

Specifies if the physical address is decremented or incremented during a transfer.

- **Device Size**

Specifies whether this is an 8-bit or 16-bit transfer.

Microprocessor and DMA

It is possible for the microprocessor to address the DMA controller and access the DMA registers. The microprocessor can control the DMA modes, transfer addresses, transfer counts, channel masks, and page registers.

Direct DMA Controller access

Reading directly from or writing directly to any DMA Controller port may cause unpredictable results.

continued

DMA data transfer

After a DMA device wins the arbitration bus and the DMA controller is programmed to service the request, a transfer can take place.

DMA transfers can be:

- single transfer,
 - multiple transfer (burst mode), or
 - read verification.
-

Burst mode

Burst mode is a method of DMA transfer that allows a device to remain inactive for long periods and then send large amounts of data in a short time. Some peripheral devices, e.g. a fixed disk, transfer their data in bursts that are frequently separated by long periods of inactivity. Burst mode is a way of making these devices more efficient. The device asks to be serviced only when it has data to transfer and then does so in large quantities.

Arbitration process

Arbitration is a process through which devices compete for control of the Micro Channel on a prioritized basis. Arbitration is organized in levels of priority, and on each level there can be a number of competing devices.

continued

Hardware Environment, Continued

Arbitration levels

Arbitration levels are predefined or programmable levels of priority assigned to devices that compete for possession of the channel.

Central Arbitration Control is a hardware function that allows intelligent peripherals to share and control access to the system.

Arbitration Levels are numbered from 00h to 0Fh, and include -1, and -2. In addition, there are Arbitration Levels -1 and -2, which exist only on the system board. Of the former set, Arbitration Level 00h has the highest priority; Level 0Eh has the lowest. All arbitration level priorities are assigned sequentially; 00h through 0Eh are the highest through lowest priorities. Level 0Fh is reserved for use by the microprocessor.

The BIOS DMA Service reads the fixed disk arbitration level from the CBIOS extended data area at BIOS initialization and uses this arbitration level throughout the BIOS session.

continued

Hardware Environment, Continued

Arbitration levels table

The following table summarizes the arbitration levels:

Arbitration Levels	Primary Assignment
FEh	Memory Refresh
FFh	NMI
00h	DMA Channel 0*
01h	DMA Channel 1
02h	DMA Channel 2
03h	DMA Channel 3
04h	DMA Channel 4*
05h	DMA Channel 5
06h	DMA Channel 6
07h	DMA Channel 7
08h	Reserved
09h	Reserved
0Ah	Reserved
0Bh	Reserved
0Ch	Reserved
0Dh	Reserved
0Eh	Reserved
0Fh	Reserved for System Microprocessor
* These DMA Channels can be programmed to any arbitration level	

ABIOS functions allow these arbitration levels to be allocated, deallocated or disabled (except levels -1, -2, and 0Fh, which are permanently allocated).

continued

Hardware Environment, Continued

DMA channel flags

Additional channel flags control whether

- a transfer is to be repeated forever,
 - the memory address is to be decremented or incremented after each cycle,
 - the DMA controller maintains I/O address (this is hardwired in most devices that use DMA), or
 - a byte or a word is transferred at each cycle.
-

Physical and Virtual DMA channels

DMA channels can be either physical or virtual. A physical channel can only have one arbitration level, but a virtual channel can be programmed to own any arbitration level not currently assigned to a different channel. Thus, a virtual DMA channel can have many arbitration levels.

Functionally, there is no difference between physical and virtual channels. Priority is determined by the arbitration level only, where level 00h is the highest priority and level 0Eh the lowest.

Virtual DMA Channels and the Arbus register

The arbitration level assignment for channels 0 and 4 can be programmed using the two 4-bit Arbus registers. The Arbus registers permit dynamic reassignment of the arbitration ID value by which the DMA controller responds to DMA requests for bus arbitration. Channels 0 and 4 can then service devices at any arbitration level.

DMA channels

See Section 2 for a complete list of I/O port addresses for DMA channels. The DMA channel addresses are from 0000h–001Fh and from 0080h–00DFh.

continued

Hardware Environment, Continued

DMA extended mode

An extended mode register is available for each programmable DMA channel and is used when a DMA channel requests a DMA data transfer. DMA channels must match the transfer size of the DMA slave, which is programmed by Bit 6 of the extended mode register. DMA read transfers of 16 bits from 8-bit memory or 8-bit memory-mapped I/O devices are not supported.

The following table describes the DMA extended mode register:

Bit Number	Description
7	= 0 Reserved
6	= 0 8-bit transfer = 1 16-bit transfer
5	= 0 Reserved
4	= 0 Reserved
3	= 0 Read memory transfer = 1 Write memory transfer
2	= 0 Verify = 1 Transfer data
1	= 0 Reserved
0	= 0 I/O address 00h = 1 User programs the I/O address

Error Handling

How errors are reported

BIOS signals the status (Successful, Resume Stage after Interrupt, etc.) of each BIOS request by returning a one word Return Code at offset 0Ch in the Request Block.

If Bit 15 of the Return Code field is set, the DMA Service function requested has an error. The caller's Return Code handler routine should then test Bits 14, 13, 12, and 8 to determine the class of error that has occurred and then test the remaining bits to determine the precise nature of the error.

Function: 01h — Return Logical ID Parameters

Description

This function is a single-staged request that returns information about the specified Logical ID.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length (20h)		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0001h)		
08h	Word	Reserved (initialize to 0000h)		
0Ah	Word	Reserved (initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Byte			Hardware Interrupt Level (FFh)
11h	Byte			Arbitration Level (FFh)
12h	Word		Device ID (000Fh)	
14h	Word		Count of Units (0001h)	
16h	Word		Logical ID Flags Bits 15-4 = Reserved Bit 3 = 0 No overlap across units = 1 Overlap across units supported Bit 2 = 0 Reserved Bits 1-0 = Transfer Data Pointer Mode 00 = No Pointers Required	
18h	Word		Request Block Length (for other functions)	
1Ah	Byte		Reserved (initialize to 0000h)	Secondary Device ID
1Bh	Byte			Revision Level
1Ch	Word		Reserved (initialize to 0000h)	
1Eh	Word	Reserved (initialize to 0000h)		

continued

Function: 01h — Return Logical ID Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 02h — Reserved

Function: 03h — Read Device Parameters

Description

This function is a single-staged request that returns the DMA Parameters for the specified Logical ID and Unit.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0003h)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Return Code
10h	Word		Time-out
12h	Byte		Maximum Address in Megabytes
14h	Byte		Maximum DMA Transfer Size in Kilobytes
15h	Byte		Number of Arbitration Levels
16h	Word		Number of DMA Channels
		Reserved (Initialize to 0000h)	

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Functions: 04h – 0Ah — Reserved

Function: 0Bh — Allocate Arbitration Level

Description

This function is a single-staged request that allocates an arbitration level. The Allocate Arbitration Level function can be used to exclude other well-behaved tasks from using this Arbitration Level while a DMA transfer is performed. Valid arbitration levels are 00h through 0Eh. 0Fh is reserved.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (000Bh)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out
16h	Word	Reserved (Initialize to 0000h)	
1Fh	Byte	Arbitration Level to Allocate	

continued

Function: 0Bh — Allocate Arbitration Level, continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8001h	Arbitration Level Not Available
8006h	No Channel Available
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Ch — Deallocate Arbitration Level

Description

This function is a single-staged request that frees an arbitration level and its associated DMA channel so another task can use it.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (000Ch)	
08h	Word	Reserved (initialize to 0000h)	
0Ah	Word	Reserved (initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out
16h	Word	Reserved (initialized to 0000h)	
1Fh	Byte	Arbitration Level to deallocate	

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8002h	Arbitration Level Not Allocated
8004h	Transfer in Process No Channel Available
8007h	Arbitration Level Not Disabled
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Dh — Disable Arbitration Level

Description

This function is a single-staged request that disables the specified arbitration level for the specified device. It may be called after a DMA transfer is complete.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (000Dh)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out
16h	Word	Reserved (Initialized to 0000h)	
1Fh	Byte	Arbitration Level to disable	

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8002h	Arbitration Level Not Allocated
8004h	Transfer In Process
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Eh – DMA Transfer Status

Description

This function is a single-staged request that returns the number of bytes that remain to be transferred from the DMA controller.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (000Eh)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Time-out
16h	Word	Reserved (initialized to 0000h)	
18h	DWord		Number of Bytes Still to be Transferred
1Fh	Byte	Arbitration Level to Check	

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8002h	Arbitration Level Not Allocated
8003h	Arbitration Level Disabled
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Fh — Abort DMA Operation

Description

This function is a single-staged request that disables a DMA operation on an arbitration level and returns the number of bytes to be transferred and the address of the bytes to be transferred.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (000Fh)	
08h	Word	Reserved (initialize to 0000h)	
0Ah	Word	Reserved (initialize to 0000h)	
0Ch	Word	Return Code	Return Code
0Eh	Word		Time-out
10h	DWord		The Physical Address at the Time the Abort was Issued
16h	Word	Reserved (initialized to 0000h)	
18h	DWord		Data Count Not Transferred When Abort Was Issued
1Fh	Byte	Arbitration Level to Abort the Operation On	

continued

Function: 0Fh — Abort DMA Operation, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8002h	Arbitration Level not Allocated
8003h	Arbitration Level Disabled
8005h	No Transfer In Process
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid DMA Parameter
FFFFh	Return Code Field Not Valid

Function: 10h — DMA Transfer from Memory to I/O

Description

This function is a single-staged request that programs the DMA controller with the values specified in the Request Block and transfers the specified number of bytes from the specified memory location to the specified I/O port address (0018h or 001Ah).

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0010h)		
08h	Word	Reserved (initialize to 0000h)		
0Ah	Word	Reserved (initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	DWord	Physical Address of Memory		
14h	DWord	Physical Address of I/O		
18h	DWord	Count of Data to Transfer (in bytes)		
1Ch	Byte	Mode Control, where: Bits 7-3 = Reserved Bit 2 = Programmable I/O Bit 1 = Reserved Bit 0 = Auto Initialization		
1Dh	Byte	Transfer Control Byte 1, where: Bits 7-3 = Reserved Bit 2 = Count control = 0 Increment = 1 Decrement Bit 1 = Reserved Bit 0 = Device size = 0 8-bit = 1 16-bit		
1Eh	Byte	Transfer Control Byte 2, where: Bits 7-1 = Reserved Bit 1 = Device size = 0 8-bit = 1 16-bit		
1Fh	Byte	Arbitration Level to Use		

continued

Function: 10h — DMA Transfer from Memory to I/O, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8002h	Arbitration Level Not Allocated
8004h	Transfer In Process
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid DMA Parameter
FFFFh	Return Code Field Not Valid

Function: 11h — Read from I/O and Write to Memory

Description

This function is a single-staged request that reads the specified number of bytes from the specified I/O port address (0018h or 001Ah) and writes the bytes to the specified memory location.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0011h)		
08h	Word	Reserved (Initialze to 0000h)		
0Ah	Word	Reserved (Initialze to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	DWord	Physical Address of Memory		
14h	DWord	Physical Address of I/O		
18h	DWord	Count of Data to Transfer (in bytes)		
1Ch	Byte	Mode Control, where: Bits 7-3 = Reserved Bit 2 = Programmable I/O Bit 1 = Reserved Bit 0 = Auto initialization		
1Dh	Byte	Transfer Control Byte 1, where: Bits 7-3 = Reserved Bit 2 = Count control 0 Increment 1 Decrement Bit 1 = Reserved Bit 0 = Device size 0 8-bit 1 16-bit		
1Eh	Byte	Transfer Control Byte 2, where: Bits 7-1 = Reserved Bit 1 = Device size 0 8-bit 1 16-bit		
1Fh	Byte	Arbitration Level to Use		

continued

Function: 11h — Read from I/O and Write to Memory, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8002h	Arbitration Level Not Allocated
8004h	Transfer In Process
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 12h — Load DMA Controller Parameters

Description

This function is a single-staged request that programs the DMA Controller with the specified input values.

Request Block structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0012h)		
08h	Word	Reserved (initialize to 0000h)		
0Ah	Word	Reserved (initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	DWord	Physical Address of Memory		
14h	DWord	Physical Address of I/O		
18h	DWord	Count of Data to Transfer (In bytes)		
1Ch	Byte	Mode Control, where: Bits 7-3 = Reserved Bit 2 = Programmable I/O Bit 1 = Reserved Bit 0 = Autoinitialization		
1Dh	Byte	Transfer Control Byte 1, where: Bits 7-3 = Reserved Bit 2 = Count control 0 Increment 1 Decrement Bit 1 = Reserved Bit 0 = Device size 0 8-bit 1 16-bit		
1Eh	Byte	Transfer Control Byte 2, where: Bits 7-1 = Reserved Bit 0 = Device size 0 8-bit 1 16-bit		
1Fh	Byte	Arbitration Level to Use		

continued

Function: 12h — Load DMA Controller Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8002h	Arbitration Level Not Allocated
8004h	Transfer in Process
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Chapter 21

ABIOS Programmable Option Select Service

Overview

Description

The ABIOS Programmable Option Select (POS) Service provides functions that access and manipulate the eight POS registers on each adapter card and the system board and provide access to the CMOS RAM associated with the POS feature.

Identifying the correct CMOS RAM locations

The ABIOS Programmable Option Select Service determines if CMOS RAM and Extended CMOS RAM are available in the system. It examines the POS System ID byte, which identifies the machine type. This test is performed for every function call to this service.

continued

Overview, Continued

Programmable Option Select

POS is designed to eliminate switches from the system board and adapter cards. In place of switches, POS provides eight programmable registers. These registers are set by information contained in Adapter Descriptor Files supplied by adapter card manufacturers.

Adapter Description Files

Adapter Description Files (ADFs) are provided by the adapter card manufacturer for each adapter card. A reference diskette reads a unique adapter ID number from each card, matches it with an ADF file, and configures the system according to the information provided by the ADF.

Summary of Programmable Option Select Service functions

Function	Description
00h	Default Interrupt Handler
01h	Return Logical ID Parameters
03h-0Ah	Reserved
0Bh	Read Stored POS Data from CMOS RAM
0Ch	Write Stored POS Data from CMOS RAM
0Dh	Read POS Data from an Adapter
0Eh	Write Dynamic POS Data from Adapter

In this chapter

This chapter includes information about the following topics:

- Hardware Environment
 - Error Handling
 - ABIOS POS Service functions
-

Hardware Environment

Introduction

The ABIOS POS Service assumes that the system is based on IBM's Micro Channel or its equivalent. POS is an integral part of the Micro Channel Architecture (MCA).

Adapter slots

ABIOS supports systems with up to eight adapter cards (or more with customization). ABIOS will work with any system regardless of the number of adapter slots available.

Adapter card identification

Each adapter card must have a unique 2-byte identifier.

Adapter description files

POS data is accumulated in adapter description files (ADFs) for each adapter. A reference diskette reads .ADF files, which are created by adapter manufacturers, and stores configuration information in CMOS RAM. The BIOS POST routine reads the CMOS RAM and writes the configuration information to the POS registers of the adapters and the system board.

continued

Hardware Environment, Continued

Programmable option select I/O Ports

The following table lists the POS I/O port addresses:

I/O Port	Read/Write Status	Description
0094h	R/W	System Board Setup Enable Register, where: Bit 7 = 0 Setup system board functions 1 Enable system board function Bit 6 = Reserved Bit 5 = 0 Setup VGA 1 Enable VGA Bits 4-0 = Reserved
0095h	N/A	Reserved
0096h	R/W	Channel Position Select Register, where: 00h = No channel selected 08h = Channel 1 selected 09h = Channel 2 selected 0Ah = Channel 3 selected 0Bh = Channel 4 selected 0Ch = Channel 5 selected 0Dh = Channel 6 selected 0Eh = Channel 7 selected 0Fh = Channel 8 selected 80h = Channel reset Note: Bits 4, 5, and 6 are set to 1 when read
0097h	N/A	Reserved
0100h	R/O	POS Register 0 — Adapter Identification Byte (Least Significant Byte)
0101h	R/O	POS Register 1 — Adapter Identification Byte (Most Significant Byte)
0102h	R/W	POS Register 2 — Option Select Data Byte 1 Bits 7-1 = Reserved Bit 0 = 1 Card enable
0103h	R/W	POS Register 3 — Option Select Data Byte 3
0104h	R/W	POS Register 4 — Option Select Data Byte 4
0105h	R/W	POS Register 5 — Option Select Data Byte 4, where: Bit 7 = 1 Channel Check active Bit 6 = 0 Channel Check exception status available in POS registers 6 and 7 = 1 No status available Bits 5-0 = Reserved
0106h	R/O	POS Register 6 — Subaddress extension (LSB)
0107h	R/O	POS Register 7 — Subaddress Extension (MSB)

Error Handling

How errors are reported

ABIOS signals the status (Successful, Resume Stage after Interrupt, etc.) of each ABIOS request by returning a one word Return Code at offset 0Ch in the Request Block.

If Bit 15 of the Return Code field is set, the Programmable Option Select Service function requested has an error. The caller's Return Code handler routine should test Bits 14, 13, 12, for the class of error that has occurred and then test the remaining bits to determine the nature of the error.

Function: 01h — Return Logical ID Parameters

Description

This function is a single-staged request that returns information about the specified Logical ID.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length (20h)		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0001h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Byte		Hardware Interrupt Level (FFh)	
11h	Byte		Arbitration Level (FFh)	
12h	Word		Device ID (0010h)	
14h	Word		Count of Units (01)	
16h	Word		Logical ID Flags Bits 15-4 = Reserved Bit 3 = 0 No overlap across units Bit 2 = 0 Reserved Bits 1-0 = Transfer Data Pointer Mode 00 = No Pointers Required	
18h	Word		Request Block Length (for other functions)	
1Ah	Byte		Reserved (Initialize to 0000h)	Secondary Device ID
1Bh	Byte			Revision Level
1Ch	Word		Reserved (Initialize to 0000h)	
1Eh	Word	Reserved (Initialize to 0000h)		

continued

Function: 01h — Return Logical ID Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Functions: 02h – 0Ah — Reserved

Function: 0Bh — Read Stored POS Data from CMOS RAM

Description

This function is a single-staged request that places the two POS Adapter Identification bytes from CMOS RAM into the Request Block at offset 12h. It also moves the contents of the POS Option Select Data Bytes 1–4 from CMOS RAM to the address specified by the data buffer pointer at offset 16h in the Request Block.

continued

Function: 0Bh — Read Stored POS Data from CMOS RAM, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Bh)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	Byte	Slot Number, where: Bits 7-4 = Reserved Bits 3-0 = Slot Number, where: 0000b = System board 0001b = Slot 1 0010b = Slot 2 0011b = Slot 3 0100b = Slot 4 0101b = Slot 5 0110b = Slot 6 0111b = Slot 7 1000b = Slot 8		
11h	Byte	Reserved (Initialize to 0000h)		
12h	Word			Adapter ID data
14h	Word	Reserved (Initialize to 0000h)		
16h	DWord	Pointer to Data Buffer	Option Select Data Bytes 1-4	
1Ch	Word	Reserved (Initialize to 0000h)		

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
80FEh	CMOS RAM Checksum Invalid
80FFh	CMOS RAM Has Bad Battery
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Ch — Write Stored POS Data from CMOS RAM

Description

This function is a single-staged request that copies the Adapter ID data from offset 12h in the Request Block to CMOS RAM. It also copies the first four bytes of the contents of the data buffer at the address specified in offset 16h of the Request Block to POS Registers 2–5 (Option Select Data Bytes 1–4) in CMOS RAM.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Ch)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word		Time-out	
10h	Byte	Slot Number, where: Bits 7–4 = Reserved Bits 3–0 = Slot Number, where: 0000b = System board 0001b = Slot 1 0010b = Slot 2 0011b = Slot 3 0100b = Slot 4 0101b = Slot 5 0110b = Slot 6 0111b = Slot 7 1000b = Slot 8		
11h	Byte	Reserved (Initialize to 0000h)		
12h	Word	Adapter ID		Adapter ID data from CMOS
14h	Word	Reserved (Initialize to 0000h)		
16h	DWord	Pointer to Data Buffer		Option Select Data Bytes 1–4
1Ch	Word	Reserved (Initialize to 0000h)		

continued

Function: 0Ch — Write Stored POS Data from CMOS RAM,
Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
80FEh	CMOS RAM Checksum Invalid
80FFh	CMOS RAM Has Bad Battery
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Dh — Read POS Data from an Adapter

Description

This function is a single-staged request that reads the POS Adapter Identification bytes directly from I/O ports 0100h and 0101h and enters this data in the Request Block at offset 12h. It also copies the POS Registers 2-5 (Option Select Data Bytes 1-4) from I/O ports 0102h-0105h to the data buffer pointed to by the address in offset 16h of the Request Block.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Ch)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	Byte	Slot Number, where: Bits 7-4 = Reserved Bits 3-0 = Slot Number, where: 0000b = System board 0001b = Slot 1 0010b = Slot 2 0011b = Slot 3 0100b = Slot 4 0101b = Slot 5 0110b = Slot 6 0111b = Slot 7 1000b = Slot 8		
11h	Byte	Reserved (Initialize to 0000h)		
12h	Word		Adapter Card ID	
14h	Word	Reserved (Initialize to 0000h)		
16h	DWord	Pointer to Data Buffer	Option Select Data Bytes 1-4	
1Ch	Word	Reserved (Initialize to 0000h)		

continued

Function: 0Dh — Read POS Data from an Adapter, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 0Eh — Write Dynamic POS Data from an Adapter

Description

This function is a single-staged request that writes the first four bytes of data found at the address specified in the Request Block at offset 16h to I/O ports 0102h–0105h (POS Registers 2–5 containing Option Select Data Bytes 1–4).

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Eh)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	Byte	Slot Number, where: Bits 7–4 = Reserved Bits 3–0 = Slot Number, where: 0000b = System board 0001b = Slot 1 0010b = Slot 2 0011b = Slot 3 0100b = Slot 4 0101b = Slot 5 0110b = Slot 6 0111b = Slot 7 1000b = Slot 8		
11h	Byte	Reserved (Initialize to 0000h)		
14h	Word	Reserved (Initialize to 0000h)		
16h	DWord	Pointer to Data Buffer		
1Ch	Word	Reserved (Initialize to 0000h)		

continued

Function: 0Eh — Write Dynamic POS Data from an Adapter,

Continued

Return Code

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Chapter 22

ABIOS Keyboard Security Service

Overview

Introduction

The ABIOS Keyboard Security Service consists of a set of functions that allow the caller to implement the security features of PS/2-compatible systems. Security features include password-controlled access to the system.

Types of password

Micro Channel-based PS/2-compatible systems may support two types of passwords: a power-on password, and a keyboard security password. Password security is controlled through a reference diskette utility.

Power-on password

The power-on password limits access to a PS/2-compatible system by testing for a password when the system is first turned on. Passwords are controlled by the end user through the reference diskette. The power-on password is stored in nonvolatile CMOS RAM.

continued

Overview, Continued

Keyboard security password

The keyboard security password limits access to a PS/2-compatible system by testing for a password when the first attempt to use the keyboard is made. Passwords are controlled by the end user through a reference diskette. The keyboard security password is stored in 8042 RAM.

Password storage

The Keyboard Security password is stored in 8042 RAM and is thus volatile. Initially, the BIOS uses the power-on password as the keyboard security password. The power-on password is stored in nonvolatile CMOS RAM (at 38h) and is transferred to the 8042 during the BIOS POST initialization.

How passwords are set

The Reference Diskette controls the passwords for both the power-on password and the keyboard security password. See the manual which accompanies your reference diskette for more information about system password protection.

Summary of BIOS Keyboard Security Service functions

Function	Description
01h	Return Logical ID Parameters
02h	Reserved
03h	Read Device Parameters
04h-05h	Reserved
06h	Enable Keyboard Security
07h-0Ah	Reserved
0Bh	Write Password
0Ch	Write Invocation Byte
0Dh	Write Match Byte
0Eh	Write Filter Byte 1
0Fh	Write Filter Byte 2

continued

Overview, Continued

In this chapter

This chapter includes information about the following topics:

- Hardware Environment
 - System Scan Codes
 - Error Handling
 - BIOS Keyboard Security Service functions
-

Hardware Environment

Introduction

The BIOS Keyboard Security Service interfaces with the features of an appropriately programmed Intel 8042 (or equivalent) intelligent keyboard controller.

Storage of Keyboard Security data in 8042

The following table shows where the Keyboard Security function bytes are stored in the 8042 and the values to which they are initialized.

8042 RAM Location	Keyboard Security Function	Initialized Value
33h	Invocation Byte	0
34h	Match Byte	0
36h	Filter Byte 1	Left Shift Key
37h	Filter Byte 2	Right Shift Key

System Scan Codes

The following system scan codes, including multiple byte codes, can be used for the Write Password, Write Invocation and Write Match Byte functions.

Typewriter/Function Keys

Key #	U.S. Keyboard Legend	System Scan Codes (hex)
1	'~	0E
2	1!	16
3	2@	1E
4	3#	26
5	4\$	25
6	5%	2E
7	6	36
8	7&	3D
9	8*	3E
10	9(46
11	0)	45
12	-_	4E
13	=+	55
15	Backspace	66
16	Tab	0D
17	Q	15
18	W	1D
19	E	24
20	R	2D
21	T	2C
22	Y	35
23	U	3C
24	I	43
25	O	44

Key #	U.S. Keyboard Legend	System Scan Codes (hex)
26	P	4D
27	[{	54
28] }	5B
29	101-key only	5D
30	Caps Lock	58
31	A	1C
32	S	1B
33	D	23
34	F	2B
35	G	34
36	H	33
37	J	3B
38	K	42
39	L	4B
40	; :	4C
41	' "	52
42	102-key only	5D
43	Enter	5A
44	L Shift	12
45	102-key only	61
46	Z	1A
47	X	22
48	C	21
49	V	2A

continued

System Scan Codes, Continued

Typewriter/Function Keys, cont'd

Key #	U.S. Keyboard Legend	System Scan Codes (hex)
50	B	32
51	N	31
52	M	3A
53	, <	41
54	. >	49
55	/ ?	4A
57	R Shift	59
58	L Ctrl	14
60	L Alt	11
61	Space	29
62	R Alt	E0-11
64	R Ctrl	E0-14
90	Num Lock	77
91	7 Home	6C
92	4 Left	6B
93	1 End	69
96	8 Up	75
97	5	73
98	2 Down	72
99	0 Ins	70
100	*	7C

Key #	U.S. Keyboard Legend	System Scan Codes (hex)
101	9 PgUp	7D
102	6 Right	74
103	3 Page Down	7A
104	. Del	71
105	-	7B
106	+	79
108	Enter	E0-5A
110	Esc	76
112	F1	05
113	F2	06
114	F3	04
115	F4	0C
116	F5	03
117	F6	0B
118	F7	83
119	F8	0A
120	F9	01
121	F10	09
122	F11	78
123	F12	07
125	Scroll Lock	7E

Other keys

The rest of the keys send a series of codes that depend on the state of the shift keys (Ctrl, Alt, and Shift) and the Num Lock key (On or Off). Since the base scan code is the same as that for another key, an additional code (hex E0) is added to the base code so that it is unique. The following four tables summarize the scan codes for these other keys.

continued

System Scan Codes, Continued

Cursor/Control Keys

Key #	U.S. Keyboard Legend	Base Case or Shift + Num Lock	Shift Case*	Num Lock on
75	Insert	E0-70	E0 F0 12 E0 70	E0 12 E0 70
76	Delete	E0-71	E0 F0 12 E0 71	E0 12 E0 71
79	Left	E0-6B	E0 F0 12 E0 6B	E0 12 E0 6B
80	Home	E0-6C	E0 F0 12 E0 6C	E0 12 E0 6C
81	End	E0-69	E0 F0 12 E0 69	E0 12 E0 69
83	Up	E0-75	E0 F0 12 E0 75	E0 12 E0 75
84	Down	E0-72	E0 F0 12 E0 72	E0 12 E0 72
85	Page Up	E0-7D	E0 F0 12 E0 7D	E0 12 E0 7D
86	Page Down	E0-7A	E0 F0 12 E0 7A	E0 12 E0 7A
89	Right	E0-74	E0 F0 12 E0 74	E0 12 E0 74

* With the Left Shift key down, the F0 12 shift code is added to the other scan codes sent. With the Right Shift key down, F0 59 is added. When both keys are down, both sets of codes are sent with the rest of the scan code.

"/" Key on Numeric Keypad

Key #	U.S. Keyboard Legend	System Scan Codes (hex)	Shift Case*
95	/	E0 4A	E0 F0 12 E0 4A

* With the Left Shift key down, the F0 12 shift code is added to the other scan codes sent. With the Right Shift key down, F0 59 is added. When both keys are down, both sets of codes are sent with the rest of the scan code.

continued

System Scan Codes, Continued

Print Screen/Sys Req Key

Key #	U.S. Keyboard Legend	System Scan Codes (hex)	Ctrl Case Shift Case	Alt Case
124	Print Screen	E0 12 E0 7C	E0 7C	84

Pause/Break Key

Key #	U.S. Keyboard Legend	System Scan Codes (hex)	Ctrl Key Pressed
126	Pause	E1 14 77 E1 F0 14 F0 77	E0 7E E0 F0 7E

Error Handling

How errors are reported

ABIOS signals the status (Successful, Resume Stage after Interrupt, etc.) of each ABIOS request by returning a one word Return Code at offset 0Ch in the Request Block.

If Bit 15 of the Return Code field is set, the Keyboard Security Service function requested has an error. The caller's Return Code handler routine should test Bits 14, 13, 12, and 8 for the class of error that has occurred and then test the remaining bits to determine the precise nature of the error.

Function: 01h — Return Logical ID Parameters

Description

This function is a single-staged request that returns the parameters for the Logical ID associated with the keyboard hardware.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length (20h)		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0001h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word	Reserved (Initialize to 0000h)		
10h	Byte		Hardware Interrupt Level (FFh)	
11h	Byte		Arbitration Level (FFh)	
12h	Word		Device ID (0016h)	
14h	Word		Count of Units	
16h	Word		Logical ID flags Bits 15-4 = Reserved Bit 3 = 0 No overlap across units = 1 Overlap across units supported Bit 2 = 0 Reserved Bits 1-0 = Transfer Data Pointer Mode 00 No Pointers Required	
18h	Word		Request Block Length (for other functions)	
1Ah	Byte		Secondary Device ID	
1Bh	Byte		Revision Level	
1Ch	Word		Reserved (Initialize to 0000h)	
1Eh	Word		Reserved (Initialize to 0000h)	

continued

Function: 01h — Return Logical ID Parameters, Continued

Return Code

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0005h	Not My Interrupt, Resume Stage after Interrupt
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Function: 02h — Reserved

Function: 03h — Read Device Parameters

Description

This single-staged function returns the maximum password length (1–7 bytes).

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0003h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Byte		Maximum Password Length	
11h	Byte	Reserved (Initialize to 0000h)		

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
0005h	Not My Interrupt, Resume Stage after Interrupt
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Functions: 04h – 05h — Reserved

Function: 06h — Enable Keyboard Security

Description

This single-staged function enables keyboard security. It is called after the Write Password, Write Invocation Byte, Write Match Byte, and Write Filter Byte functions have initialized the password entry procedure.

Request Block structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0006h)		
08h	Word	Reserved (initialize to 0000h)		
0Ah	Word	Reserved (initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
11h	1 Byte	Reserved (initialize to 0000h)		

continued

Function: 06h — Enable Keyboard Security, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8000h	Device Busy
8003h	Device Inhibited
C000h	Invalid Logical ID
C001h	Invalid Function Number
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Keyboard Security Parameter
FFFFh	Return Code Field Not Valid

Functions: 07h – 0Ah — Reserved

Function: 0Bh — Write Password

Description

This single-staged function sets the password or changes an existing password. Write Password writes a string of system scan codes from 1–7 bytes long to 8042 RAM (see the System Scan Code tables in this chapter).

A password example

The password, ABC, is stored in the 8042 as follows:

1Eh 30h 2Eh 1Ch 00h

where

- 1Eh is the system scan code for A
- 30h is the system scan code for B
- 2Eh is the system scan code for C
- 1Ch is the system scan code for the Return key
- The 00h is set by the BIOS

The routine using this function matches each character as it is received from the keyboard. When the return key scan code is received, it is matched, which signals the end of the password.

Password storage

The Keyboard Security password is stored in 8042 RAM and is thus volatile. Initially, the BIOS uses the power-on password as the keyboard security password. The power-on password is stored in nonvolatile CMOS RAM (at 38h) and is transferred to the 8042 during the BIOS POST initialization.

continued

Function: 0Bh — Write Password, Continued

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0008h)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	Byte	Password Length in bytes		
11h	Byte	Reserved (Initialize to 0000h)		
12h	Byte	First Scan Code		
13h	Byte	Second Scan Code		
14h	Byte	Third Scan Code		
15h	Byte	Fourth Scan Code		
16h	Byte	Fifth Scan Code		
17h	Byte	Sixth Scan Code		
18h	Byte	Seventh Scan Code		

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8000h	Device Busy
8003h	Device Inhibited
C000h	Invalid Logical ID
C001h	Invalid Function Number
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Keyboard Security Parameter This value is set and no action is taken if the Password Length field is 0 or greater than 7 bytes
FFFFh	Return Code Field Not Valid

Function: 0Ch — Write Invocation Byte

Description

This single-staged function sets or changes the Invocation Byte. The Invocation Byte may contain any system scan code. The byte is stored at location 33h in 8042 RAM. The default value for this scan code is zero.

Invocation Byte usage

This byte acts as a signal from the 8042 to the caller indicating that a valid password has been loaded and that password security has been enabled. The Invocation Byte is returned to the caller to invoke security. If the Invocation Byte is zero, keyboard security is not enabled.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Ch)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Byte	Invocation Byte Scan Code		
11h	Byte	Reserved (Initialize to 0000h)		

continued

Function: 0Ch — Write Invocation Byte, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8000h	Device Busy
8003h	Device Inhibited
C000h	Invalid Logical ID
C001h	Invalid Function Number
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Keyboard Security Parameter
FFFFh	Return Code Field Not Valid

Function: 0Dh — Write Match Byte

Description

This function sets or changes the Match Byte. The byte is stored in location 34h of 8042 RAM. The default value is zero.

Match Byte usage

This byte is used as a signal from the 8042 to the caller that keyboard security has been disabled now that the keyboard input matches the system scan code(s) in the password field. The 8042 sends this byte to the caller using a keyboard interrupt. If the Match Byte is zero, it will not be used to signal the keyboard security disabled status to the caller.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Dh)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			Time-out
10h	Byte	Scan Code		
11h	Byte	Reserved (Initialize to 0000h)		

continued

Function: 0Dh — Write Match Byte, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8000h	Device Busy
8003h	Device Inhibited
C000h	Invalid Logical ID
C001h	Invalid Function Number
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Keyboard Security Parameter
FFFFh	Return Code Field Not Valid

Function: 0Eh — Write Filter Byte 1

Description

This function sets or changes Filter Byte 1. This byte contains a system scan code that is ignored if it is encountered during password validation. Filter Byte 1 is initialized to left shift key.

Usage of Filter Bytes

If unaltered, Filter Bytes 1 and 2 together ensure that keyboard security input is not case-sensitive.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (000Eh)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	
0Eh	Word		Return Code
10h	Byte	Filter Byte 1	Time-out
11h	Byte	Reserved (Initialize to 0000h)	

continued

Function: 0Eh — Write Filter Byte 1, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8000h	Device Busy
8003h	Device Inhibited
C000h	Invalid Logical ID
C001h	Invalid Function Number
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Keyboard Security Parameter
FFFFh	Return Code Field Not Valid

Function: 0Fh — Write Filter Byte 2

Description

This function sets or changes Filter Byte 2. This byte contains a system scan code that is ignored if it is encountered during password validation. Filter Byte 2 is initialized to right shift key.

Usage of Filter Bytes

If unaltered, Filter Bytes 1 and 2 together ensure that keyboard security input is not case-sensitive.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (000Fh)		
08h	Word	Reserved (Initialize to 0000h)		
0Ah	Word	Reserved (Initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Byte	Filter Byte 2		
11h	Byte	Reserved (Initialize to 0000h)		

continued

Function: 0Fh — Write Filter Byte 2, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
8000h	Device Busy
8003h	Device Inhibited
C000h	Invalid Logical ID
C001h	Invalid Function Number
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid Keyboard Security Parameter
FFFFh	Return Code Field Not Valid

Chapter 23

ABIOS Error Log Service

Overview

Introduction

The ABIOS Error Log Service is used by the operating system to read and write error messages to the system error log area in extended CMOS RAM.

This service can only be used by systems that have extended CMOS RAM.

ABIOS start routines

Requests for the ABIOS Error Log Service should use only the ABIOS start routine. There is no need to use the interrupt or time-out routines.

continued

Overview, Continued

Summary of Error Log Service functions

Function	Description
01h	Return Logical ID Parameters
02h-07h	Reserved
08h	Read Error Log
09h	Write to Error Log
0Ah	Reserved

In this chapter

In this chapter, the following topics are discussed:

- Hardware environment
 - Error handling
 - BIOS Error Log Service functions
-

Extended CMOS RAM

Introduction

The BIOS Error Log Service assumes that at least 2K of Extended CMOS RAM is available in the system.

CMOS RAM Error Log Area

Up to six 20-byte error log blocks can be stored in Extended CMOS RAM

Note: If all error log block positions contain error information, as a new error log block is read in, the oldest error log block is overwritten.

Extended CMOS RAM Error Block data area

Location	Description
389h	ERRNUM — Number of Error Log entries — can be 0-6
38Ah	Error Log Block 0
39Eh	Error Log Block 1
3B2h	Error Log Block 2
3C6h	Error Log Block 3
3DAh	Error Log Block 4
3EEh	Error Log Block 5

Using ERRNUM

ERRNUM is a pointer to the next available error log block. It is maintained by the operating system.

Pending error log blocks

The operating system must keep track of the total number of active error log blocks pending.

If more than six error log blocks are written to extended CMOS RAM, synchronization errors may occur unless the error log blocks are tracked by the operating system.

continued

Extended CMOS RAM, Continued

Error Log Block format

Offset	Length	Description
00h	1 Byte	Error ID Byte
01h	1 Byte	Interrupt Level
02h	1 Byte	Arbitration Level
03h	1 Byte	Device ID
04h	10 Bytes	Device-Specific Parameters
0Eh	1 Byte	Current Seconds (BCD)
0Fh	1 Byte	Current Minutes (BCD)
10h	1 Byte	Current Hours (BCD)
11h	1 Byte	Current Day (BCD)
12h	1 Byte	Current Month (BCD)
13h	1 Byte	Current Year (BCD)

I/O ports

The BIOS Error Log Service accesses the following I/O port addresses to access CMOS RAM locations.

I/O Address	Read/Write Status	Description
0074h	W	Extended CMOS RAM address register port, least significant byte
0075h	W	Extended CMOS RAM address register port, most significant byte
0076h	R/W	Extended CMOS RAM data register port

Error Handling

How errors are reported

ABIOS signals the status (Successful, Resume Stage after Interrupt, etc.) of each ABIOS request by returning a one word Return Code at offset 0Ch in the Request Block.

If Bit 15 of the Return Code field is set, the Error Log Service function requested has an error. The caller's Return Code handler routine should test Bits 14, 13, 12, and 8 to determine the class of error that has occurred and then test the remaining bits to determine the precise nature of the error.

Function: 01h — Return Logical ID Parameters

Description

This function is a single-staged request that returns information about the specified Logical ID.

Request Block structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length (20h)		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0001h)		
08h	Word	Reserved (initialize to 0000h)		
0Ah	Word	Reserved (initialize to 0000h)		
0Ch	Word	Return Code		Return Code
0Eh	Word			
10h	Byte			Hardware Interrupt Level (FFh)
11h	Byte			Arbitration Level (FFh)
12h	Word		Device ID (00011h)	
14h	Word		Count of Units (0001h)	
16h	Word		Logical ID Flags (0000h) Bits 15-4 = Reserved Bit 3 = 0 No overlap across units Bit 2 = 0 Reserved Bits 1-0 = Transfer Data Pointer Mode 00 = No Pointers Required	
18h	Word		Request Block Length (for other functions)	
1Ah	Byte		Reserved (initialize to 0000h)	Secondary Device ID
1Bh	Byte		Revision Level	
1Ch	Word	Reserved (initialize to 0000h)		
1Eh	Word	Reserved (initialize to 0000h)		

continued

Function: 01h — Return Logical ID Parameters, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
C000h	Invalid Logical ID
C001h	Invalid Function
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
FFFFh	Return Code Field Not Valid

Functions: 02h – 07h — Reserved

Function: 08h — Read Error Log

Description

This function, a single-staged request, returns the most current error log block from extended CMOS RAM. After the error log entry is read, the memory location it was stored in is cleared to zero. Up to six active error log entries can exist at one time.

Device-specific parameters

The operating system must determine the contents and format of the 10 bytes of device-specific parameters that are input at offset 14h of the Request Block in Function 09h, Write Error Log. Each device type should have a different set of device-specific parameters.

Request Block Structure

Offset	Size	Input:	Output:
00h	Word	Request Block Length	
02h	Word	Logical ID	
04h	Word	Unit	
06h	Word	Function (0008h)	
08h	Word	Reserved (Initialize to 0000h)	
0Ah	Word	Reserved (Initialize to 0000h)	
0Ch	Word	Return Code	Return Code
0Eh	Word		
10h	Byte		Error ID
11h	Byte		Interrupt Level
12h	Byte		Arbitration Level
13h	Byte		Device ID
14h	10 Bytes		Device-Specific Parameters
1Eh	Byte		Current Seconds (in BCD)
1Fh	Byte		Current Minutes (in BCD)
20h	Byte		Current Hours (in BCD)
21h	Byte		Current Day (in BCD)
22h	Byte	Current Month (in BCD)	
23h	Byte	Current Year (in BCD)	

continued

Function: 08h — Read Error Log, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
80FEh	CMOS RAM Checksum Invalid
80FFh	CMOS RAM Battery Bad
C000h	Invalid Logical ID
C001h	Invalid Function Number
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid CMOS RAM Parameter This field is set and no action is taken if the bytes in the Number of Bytes to Transfer field plus the byte count for the Starting RAM Address field in the Request Block is greater than the maximum amount of RAM.
FFFFh	Return Code Field Not Valid

Function: 09h — Write to Error Log

Description

This function, a single-staged request, writes a 20-byte error log block to the next available error log entry in the external CMOS RAM error log block data area.

Device-specific parameters

The operating system must determine the format and contents of the ten bytes of device-specific parameters that are input at offset 14h of the Request Block. Each device type should have a different set of device-specific parameters.

Request Block Structure

Offset	Size	Input:	Output:	
00h	Word	Request Block Length		
02h	Word	Logical ID		
04h	Word	Unit		
06h	Word	Function (0009h)		
08h	Word	Reserved (initialize to 0000h)		
0Ah	Word	Reserved (initialize to 0000h)		
0Ch	Word	Return code		Return code
0Eh	Word			
10h	Byte	Error ID Byte (must be 00h-0Dh)		
11h	Byte	Interrupt Level		
12h	Byte	Arbitration Level		
13h	Byte	Device ID		
14h	10 Bytes	Device-specific Parameters		
1Eh	Byte	Reserved	Current Seconds (in BCD)	
1Fh	Byte	Reserved	Current Minutes (in BCD)	
20h	Byte	Reserved	Current Hours (in BCD)	
21h	Byte	Reserved	Current Day (in BCD)	
22h	Byte	Reserved	Current Month (in BCD)	
23h	Byte	Reserved	Current Year (in BCD)	

continued

Function: 09h – Write to Error Log, Continued

Return Codes

This list contains only the most common Return Codes for this function. Test all valid bits in the Return Code field to ensure proper performance.

Code	Description
0000h	Successful Operation
80FEh	CMOS RAM Checksum Invalid
80FFh	CMOS RAM Battery Bad
C000h	Invalid Logical ID
C001h	Invalid Function Number
C003h	Invalid Unit Number
C004h	Invalid Request Block Length
C005h	Invalid CMOS RAM Parameter This field is set and no action is taken if the bytes in the Number of Bytes to Transfer field plus the byte count for the Starting RAM Address field in the Request Block is greater than the maximum amount of RAM.
FFFFh	Return Code Field Not Valid

Appendix A

ABIOS Return Codes

Overview

Description

The following tables list common ABIOS Return Codes, the description of the Return Code, the ABIOS Service that generated the code, and a summary of the Return Code field bit setting meaning. See Chapter 6 for more detailed information about Return Codes.

Return Code bit settings

Any program or routine that uses the ABIOS should check the bit settings for bits 15, 14, 13, 12, 7, 6, 5, 4, 3, 2, 1, and 0 in every Return Code. ABIOS is open and can be modified in many ways. Any code that is an extension of the ABIOS is able to set a combination of Return Code bit settings that would have a new and different meaning than those listed in the following tables.

It is also possible for bits 7–0 to take on different meanings depending on the settings of Bits 15, 13, 12, and 8. These meanings will be device-specific error indications that are listed in the service chapters (Chapters 8–23).

Return Codes that Indicate Action is Required

The following Return Codes indicate that a new stage in a multistaged request has occurred. The caller must handle the conditions indicated by these messages in order to continue processing. It is possible for other combinations of Bits 0, 1, 2, and 7 to occur. The caller's Return Code handling routine should test each meaningful bit of the Return Code field each time a Return Code is processed.

Return Code	Description	Action Required
0001h	Incomplete — resume stage after interrupt	A stage in a multistaged request has been completed; at the next interrupt on this device, the caller should resubmit this Request Block.
0002h	Resume stage after time delay	A stage in a multistaged request has been reached; after the appropriate time delay, the caller should resubmit this Request Block.
0005h	Incomplete — not my interrupt, resume stage after interrupt	A stage in a multistaged request has been completed; the operation is not complete, the interrupt that just occurred is not for this device, at the next interrupt on this device, the caller should resubmit this Request Block.
0009h	Attention, resume stage after interrupt	A stage in a multistaged request has been completed; the device needs attention, at the next interrupt on this device, the caller should resubmit this Request Block.
0081h	Unexpected interrupt reset	The interrupt for this device has been reset, restart the Request Block.

Return Codes that Indicate Termination

Non-error Return Codes

There are two conditions that may occur in a Return Code field that do not indicate an error:

- 0000h — Successful Operation. The request has been completed.
 - FFFFh — Return Code Field Not Valid. The request has not been processed by BIOS yet.
-

Return Code table

The following Return Codes indicate that the request has been terminated, either successfully or unsuccessfully. All Return Codes listed below (except 0000h and FFFFh) indicate an error. If Bit 15 is on, the Return Code is always an error.

Return Code	Description	Originating BIOS Service	Meaning
0000h	Successful Operation	Any	Successful operation
8000h	Device Busy, Request Refused	Any	Unsuccessful operation. see additional return codes.
8001h	Resend	Mouse	Unsuccessful operation
8001h	Device Busy	Parallel	Unsuccessful operation
8001h	Arbitration Level Not Available	DMA	Unsuccessful operation
8001h	Real Time Clock Not Started	RTC	Unsuccessful operation
8002h	Interrupt Already Enabled	RTC	Unsuccessful operation
8002h	Arbitration Level Not Allocated	DMA	Unsuccessful operation
8002h	Two Consecutive Resends Found	Mouse	Unsuccessful operation
8003h	System Lock	Mouse	Unsuccessful operation
8003h	Arbitration Level Disabled	DMA	Unsuccessful operation
8003h	Write-Protected Diskette	Diskette	Unsuccessful operation
8003h	Security Enabled, Keyboard Inhibited — Request Refused	Keyboard	Unsuccessful operation
8003h	Keyboard Locked, Request Refused	Keyboard Security	Unsuccessful operation

continued

Return Codes that Indicate Termination, Continued

Return Code table, cont'd

Return Code	Description	Originating BIOS Service	Meaning
8004h	Keyboard Locked, Request Refused	Keyboard	Unsuccessful operation
8004h	Transfer In Process	DMA	Unsuccessful operation
8005h	No Transfer In Process	DMA	Unsuccessful operation
8006h	Media Changed	Diskette	Unsuccessful operation
8006h	No Channel Available	DMA	Unsuccessful operation
8007h	Arbitration Level Not Disabled	DMA	Unsuccessful operation
800Dh	Media Not Present	Diskette	Unsuccessful operation
800Eh	Change Signal Not Available	Diskette	Unsuccessful operation
800Fh	Invalid CMOS Value	CMOS RAM	Unsuccessful operation
800Fh	DMA Arbitration Level Out Of Range	Fixed Disk	Unsuccessful operation
80FEh	Invalid CMOS Checksum	CMOS RAM	Unsuccessful operation, device error
80FFh	Bad CMOS Battery	CMOS RAM	Unsuccessful operation, device error
9000h	Bad Com Port	Serial	Unsuccessful operation device error
9000h	Keyboard Controller Always Busy	Keyboard	Unsuccessful operation device error
9000h	Printer Error	Serial	Unsuccessful operation device error
9000h	Printer Error	Parallel	Unsuccessful operation device error
9001h	Bad Command	Parallel	Unsuccessful operation device error
9001h	Keyboard Failed Reset	Keyboard	Unsuccessful operation device error
9002h	Address Mark Not Found	Parallel	Unsuccessful operation device error
9002h	Resend Error	Parallel	Unsuccessful operation device error
9003h	Keyboard Parity Error	Keyboard	Unsuccessful operation device error

continued

Return Codes that Indicate Termination, Continued

Return Code table, cont'd

Return Code	Description	Originating BIOS Service	Meaning
9004h	Record Not Found	Keyboard	Unsuccessful operation device error
9004h	General Hardware Time-out	Keyboard	Unsuccessful operation device error
9005h	Reset Failed	Fixed Disk	Unsuccessful operation device error
9006h	Undefined Mode From Keyboard	Keyboard	Unsuccessful operation device error
9007h	Controller Parameter Activity Failed	Fixed Disk	Unsuccessful operation device error
9009h	Controller Failure During Reset	Keyboard	Unsuccessful operation device error
900Ah	Defective Sector	Fixed Disk	Unsuccessful operation device error
900Bh	Bad Track	Fixed Disk	Unsuccessful operation device error
900Dh	Invalid Sector On Format	Fixed Disk	Unsuccessful operation device error
900Eh	CAM Detected During Read Or Verify	Fixed Disk	Unsuccessful operation device error
9010h	Uncorrectable ECC Or CRC Error	Fixed Disk	Unsuccessful operation device error
9020h	Bad Controller	Fixed Disk Diskette	Unsuccessful operation device error
9021h	Equipment Check	Fixed Disk	Unsuccessful operation device error
9040h	Bad Seek	Fixed Disk	Unsuccessful operation device error
9080h	Device Did Not Respond	Fixed Disk, Diskette	Unsuccessful operation device error
90AAh	Drive Not Ready	Fixed Disk	Unsuccessful operation device error
90BBh	Undefined Error	Fixed Disk	Unsuccessful operation device error
90CCh	Write Fault	Fixed Disk	Unsuccessful operation device error

continued

Return Codes that Indicate Termination, Continued

Return Code table, cont'd

Return Code	Description	Originating BIOS Service	Meaning
90CCh	Keyboard Controller Always Busy	Keyboard	Unsuccessful operation device error
90FFh	Incomplete Sense Operation	Fixed Disk	Unsuccessful operation device error
9100h	Controller Failure	Mouse	Unsuccessful operation device error
9100h	Keyboard Controller Always Busy	Keyboard	Unsuccessful operation device error
9101h	Keyboard Failed Reset	Keyboard	Unsuccessful operation device error
9101h	Bad Command	Parallel	Unsuccessful operation device error
9102h	Address Mark Not Found	Diskette	Unsuccessful operation device error
9102h	Resend Error	Keyboard	Unsuccessful operation retryable device error
9102h	Resend Error	Parallel	Unsuccessful operation device error
9103h	Keyboard Parity Error	Keyboard	Unsuccessful operation device error
9103h	Parity Error	Fixed Disk	Unsuccessful operation device error
9104h	General Device Time-out	Keyboard	Unsuccessful operation device time-out error
9104h	Requested Sector Not Found	Diskette	Unsuccessful operation retryable device error
9105h	Reset Failed	Fixed Disk	Unsuccessful operation retryable device error
9107h	Controller Parameter Activity Failed	Fixed Disk	Unsuccessful operation retryable device error
9108h	DMA Overrun On Operation	DMA, Fixed Disk Diskette	Unsuccessful operation retryable device error
9110h	Bad CRC On Diskette Read	Diskette	Unsuccessful operation retryable device error
9120h	Controller Failure	Diskette, Fixed Disk	Unsuccessful operation retryable device error

continued

Return Codes that Indicate Termination, Continued

Return Code bit table, cont'd

Return Code	Description	Originating BIOS Service	Meaning
9121h	Equipment Check	Fixed Disk, Diskette	Unsuccessful operation retryable device error
9140h	Seek Operation Failed	Diskette, Fixed Disk	Unsuccessful operation retryable device error
9180h	Device Did Not Respond	Diskette, Fixed Disk	Unsuccessful operation retryable device error
91AAh	Drive Not Ready	Fixed Disk	Unsuccessful operation retryable device error
91BBh	Undefined Error	Fixed Disk	Unsuccessful operation retryable device error
91CCh	Write Fault	Fixed Disk	Unsuccessful operation retryable device error
91FFh	Incomplete Sense Operation	Fixed Disk	Unsuccessful operation retryable device error
A000h	Time-out	Fixed Disk	Unsuccessful operation retryable device error
A001h	Bad Command	Fixed Disk	Unsuccessful operation retryable device error
A002h	Address Mark Not Found	Fixed Disk	Unsuccessful operation retryable device error
A004h	Record Not Found	Fixed Disk	Unsuccessful operation time-out error
A005h	Reset Failed	Fixed Disk	Unsuccessful operation time-out error
A007h	Parameter Activity Failed	Fixed Disk	Unsuccessful operation time-out error
A00Ah	Defective Sector	Fixed Disk	Unsuccessful operation time-out error
A00Bh	Bad Track	Fixed Disk	Unsuccessful operation time-out error
A00Dh	Invalid Sector On Format	Fixed Disk	Unsuccessful operation time-out error
A00Eh	CAM Detected During Read Or Verify	Fixed Disk	Unsuccessful operation time-out error
A010h	Uncorrectable ECC Or CRC Error	Fixed Disk	Unsuccessful operation time-out error

continued

Return Codes that Indicate Termination, Continued

Return Code table, cont'd

Return Code	Description	Originating BIOS Service	Meaning
A011h	ECC-corrected Data Error	Fixed Disk	Unsuccessful operation time-out error
A020h	Bad Controller	Fixed Disk	Unsuccessful operation time-out error
A021h	Equipment Check	Fixed Disk	Unsuccessful operation time-out error
A040h	Bad Seek	Fixed Disk	Unsuccessful operation time-out error
A080h	Device Did Not Respond	Fixed Disk	Unsuccessful operation time-out error
A0AAh	Drive Not Ready	Fixed Disk	Unsuccessful operation time-out error
A0BBh	Undefined Error	Fixed Disk	Unsuccessful operation time-out error
A0CCh	Write Fault	Fixed Disk	Unsuccessful operation time-out error
A0FFh	Incomplete Sense Operation	Fixed Disk	Unsuccessful operation time-out error
A100h	Time-out Occurred — No Other Error	Fixed Disk	Unsuccessful operation time-out error
A105h	Reset Failed	Fixed Disk	Unsuccessful operation device time-out error
A107h	Controller Parameter Activity Failed	Fixed Disk	Unsuccessful operation device time-out error
A120h	Controller Failure	Diskette, Fixed Disk	Unsuccessful operation retryable time-out error
A121h	Equipment Check	Fixed Disk	Unsuccessful operation retryable time-out error
A140h	Bad Seek	Fixed Disk	Unsuccessful operation retryable time-out error
A180h	Device Did Not Respond	Fixed Disk	Unsuccessful operation retryable time-out error
A1AAh	Drive Not Ready	Fixed Disk	Unsuccessful operation retryable time-out error
A1BBh	Undefined Error	Fixed Disk	Unsuccessful operation retryable time-out error

continued

Return Codes that Indicate Termination, Continued

Return Code table, cont'd

Return Code	Description	Originating BIOS Service	Meaning
A1CCh	Write Fault	Fixed Disk	Unsuccessful operation retryable time-out error
A1FFh	Incomplete Sense Operation	Fixed Disk	Unsuccessful operation retryable time-out error
B001h	Bad Command	Fixed Disk	Unsuccessful operation retryable time-out error
B001h	Keyboard Error	Keyboard	Unsuccessful operation retryable device time-out error
B020h	Controller Failure	Diskette	Unsuccessful operation retryable time-out error
B020h	Bad Controller	Fixed Disk	Unsuccessful operation retryable device time-out error
B021h	Equipment Check	Fixed Disk	Unsuccessful operation retryable device time-out error
B080h	Device Did Not Respond	Fixed Disk	Unsuccessful operation retryable device time-out error
B101h	Bad Command	Fixed Disk	Unsuccessful operation device time-out error
B101h	Keyboard Error	Keyboard	Unsuccessful operation device time-out error
B120h	Controller Failure	Diskette	Unsuccessful operation device time-out error
B120h	Bad Controller	Fixed Disk	Unsuccessful operation retryable device time-out error
B121h	Equipment Check	Fixed Disk	Unsuccessful operation retryable device time-out error
B180h	Device Did Not Respond	Fixed Disk	Unsuccessful operation retryable device time-out error
B0BBh	Undefined Error	Fixed Disk	Unsuccessful operation retryable device time-out error

continued

Return Codes that Indicate Termination, Continued

Return Code table, cont'd

Return Code	Description	Originating BIOS Service	Meaning
B0FFh	Sense Failed	Fixed Disk	Unsuccessful operation retryable device time-out error
B1BBh	Undefined Error	Fixed Disk	Unsuccessful operation retryable device time-out error
B1FFh	Sense Failed	Fixed Disk	Unsuccessful operation retryable device time-out error
C000h	Invalid Logical ID	Any	Unsuccessful operation parameter error
C001h	Invalid Function	Any	Unsuccessful operation parameter error
C003h	Invalid Unit Number	Any	Unsuccessful operation parameter error
C004h	Invalid Request Block Length	Any	Unsuccessful operation parameter error
C005h	Invalid Parameter	Any	Unsuccessful operation parameter error
C006h	Invalid Time To Wait	Parallel	Unsuccessful operation parameter error
C00Ch	Unsupported Media Type/Unestablished Media	Diskette	Unsuccessful operation parameter error
FFFFh	Return Code Not Valid	Any	Request block not processed

Glossary

ABIOS

Advanced BIOS. The BIOS designed to support multitasking operating systems such as OS/2. It comes packaged with a traditional Compatibility BIOS (CBIOS).

ABIOS Service

Each ABIOS service controls a hardware device. Each ABIOS Service can only be associated with one Device ID.

Adapter Card

A circuit board that can be installed into one of the expansion slots inside a PS/2-compatible computer in order to expand the capabilities of the computer.

Adapter Description Files (ADFs)

Text files supplied on a diskette by manufacturers of PS/2-compatible adapter cards. The ADFs contain information such as what resources are needed to use the card. The ADFs must be copied to a working copy of the Reference Diskette after the adapter card is installed.

continued

Adapter ROM

The read-only memory on the adapter, which contains code to control the adapter device. An adapter is a peripheral card that extends the operation of the system. For example, a fixed disk drive controller is an adapter that may have an adapter ROM. Adapter ROM code may include an BIOS ROM extension.

Address Bus

A set of signals which select a certain cell of memory or a certain device from the microprocessor to all parts of the system.

Anchor Pointer

A segment or selector with an assumed offset of zero, which is passed to the BIOS on each request made of the BIOS. It points to the real mode Common Data Area.

Arbitration

Arbitration is a process through which devices compete for possession of the channel on a prioritized basis.

Arbitration Level

Arbitration levels are the levels of priority assigned to devices that compete for possession of the channel.

Bimodal Operation

Refers to the ability of a program to operate in both the real address mode and the protected address mode of the 80286, 80386SX, or 80386 microprocessor. BIOS services are bimodal; they operate in either real or protected mode.

BIOS

Basic Input/Output System. Systems software that interfaces between the operating system and hardware.

Boot

Process of starting the computer.

continued

Glossary, Continued

Burst Mode

Burst mode is a method of DMA transfer that allows a device to remain inactive for long periods and then send large amounts of data in a short time.

Bus

One or more lines (conductors) that carry signals or power.

Byte

Eight contiguous bits; a bit is the smallest item of information that a computer can process.

Cache

Method of using a fast device to speed up access to a slow device.

CBIOS

Compatibility BIOS. The traditional single-tasking portion of a PS/2-compatible BIOS.

CBIOS Service

A software routine that services a given peripheral device, and provides an interface between the operating system and the hardware. These services are single task, call/return functions, as opposed to the device services offered by ABIOS.

CMOS

Acronym for Complementary Metal Oxide Semiconductor. In PS/2 compatibles, it is low-power memory that is battery-backed and therefore not lost when the computer is turned off.

Common Data Area

Contains a master list of pointers to the Function Transfer Table, Device Block, and Data Area (if any) associated with each ABIOS device service.

Configuration

The process of setting up all the parts of the computer so they run effectively.

continued

Continuous Multistaged Requests

BIOS functions that never reach a completion point but are continually repeated. They can be thought of as standing function calls. An example is the BIOS Keyboard Service Get Key Function.

Cyclic Redundancy Check (CRC)

A method of redundancy check where the check key is produced by a cyclic or repeating algorithm. A common means of error checking.

Default

A value, setting, or option that is assigned by the program or system.

Device Block

A permanent work area for each BIOS device, containing hardware port addresses, interrupt levels, and device status information.

Device ID

Each type of device is identified to the system by a device ID.

Direct Memory Access (DMA)

Direct Memory Access is a means for I/O devices to transfer data directly to and from system memory without the intervention of the microprocessor. This significantly decreases I/O processing by the microprocessor.

Discrete Multistaged Requests

An BIOS function that requires a significant amount of time while waiting for a hardware interrupt or time interval to occur returns control to the processor between stages of servicing a request. An example is the BIOS Diskette Service Diskette Read Function.

DMA Controller

A DMA controller is a device which gives addresses and control signals to the device that has won the bus through arbitration. The controller does not enter into the arbitration itself.

DMA Device

A DMA device enters into arbitration for the channel. If it wins, it receives addresses and control signals from the DMA controller so it can read or write data.

continued

Glossary, Continued

DOS

Acronym for Disk Operating System and short for PC-DOS and MS-DOS. DOS, like other operating systems, organizes the files and memory for other programs.

Error Handler

An invisible program on the Reference Diskette that reads the POST error log. If an error is found, a cause and solution type message about the error is displayed.

Expanded Memory

For AT-compatible systems, up to 32 MB of additional "paged" memory above the DOS 640K limit. Application programs written according to LIM EMS or AST EEMS specifications can use this type of memory. Examples of such programs are Lotus 1-2-3, Symphony and Framework.

Expanded Memory Specification (EMS)

For AT-compatible systems, a specification and protocol established by a consortium of computer manufacturers, principally Lotus, Intel, and Microsoft (LIM), which establishes a set of rules for organizing and accessing expanded memory.

Extended Expanded Memory Specification (EEMS)

A specification and protocol established by a consortium of computer and software manufacturers, principally AST, Quadram, and Ashton-Tate, which establishes a set of rules for organizing and accessing expanded memory.

Extended Memory

The memory above 1 MB. XENIX and IBM's VDISK can use this memory, but DOS and almost all application programs cannot, since use of the protected mode of the Intel 80286 or 80386 microprocessor is required.

Fixed Disk (Hard Disk)

A magnetic storage device consisting of a drive mechanism with permanently installed metallic disks; a "filing cabinet" for the computer.

continued

Function Transfer Table

A data structure that contains a list of pointers to the entry routines and function start addresses of one BIOS service. There is a Function Transfer Table for each BIOS service.

Hardware

The physical equipment and components in the computer system.

Initialization Table

An BIOS common data structure that defines initialization data for each device in the system. It is referred to when initializing each Device Block and Function Transfer Table.

Interrupt

The suspending of microprocessor program execution by a demand for attention coming from a peripheral device. After the interrupt has been serviced, the suspended microprocessor task can be resumed where it was broken off by the interrupt.

Kilobytes (K)

1024 bytes.

Known State

When a device is initialized or reset, and then set to a particular pre-established condition, it is said to be in a known state.

Logical Device

A conceptual, as opposed to physical, identification of a hardware device by an operating system, so as to allow the latter a greater degree of device independence. Operating systems commonly identify physical devices with an operating system-assigned Logical ID number. Contrast with Physical Device.

Logical ID

An identifier for a device controller used by BIOS. There may be several devices (units) attached to a Logical ID. The Logical Device ID (Logical ID/LID) is used by the operating system as an index into the Common Data Area to locate the Device Block pointer and Function Transfer Table pointer pair with which each BIOS service is associated.

continued

Glossary, Continued

Low Level Format

Electronic equivalent of drawing a detailed street map on the fixed disk. The electronic markings tell the system at what points to start and end reads and writes.

Main Memory

The memory between 0 and 1 MB.

Megabyte (MB)

One million bytes.

Memory

A device that can store data recorded in it and from which the data can be retrieved.

Micro Channel

Information is exchanged between the computer system board and the adapter cards which are plugged into it by means of the "bus." Micro Channel refers to the particular bus design in a PS/2-compatible computer. It is also referred to as MCA, for Micro Channel Architecture.

Microprocessor

Central processing unit, or "brain" of the computer.

Multitasking

Multitasking programs execute multiple program modules simultaneously. Information input into one module does not need to be processed completely before information can be input into another module. BIOS supports multiple concurrent requests.

Offset

A method of addressing that defines an address as relative to the beginning of a memory segment.

Operating System

Generic systems software which controls the execution of applications software.

continued

Option Diskette

The diskette provided by manufacturers of adapter cards that contains adapter description files (ADFs), which are written to the Reference Diskette and used by it to configure the system.

Parameter

Value, option, or setting that can be set in two or more ways.

Physical Device

A hardware device that physically exists in a system configuration. Physical devices are identified by a device ID. Contrast with logical device.

Power-on Self Test (POST)

A program that tests all parts of the computer every time you turn on the computer.

Private Data

The BIOS data structures contain areas (private data) which have information that is not available to BIOS callers. This information includes hardware device support levels, BIOS internal routines, and BIOS internal parameters.

Program

A set of instructions defining the operations of a computer in order to achieve the desired results.

Programmable Option Select (POS)

A way of setting up peripheral devices on a PS/2 machine via the power-on self test (POST), in which values are placed in registers on the devices. This software set up routine replaces the traditional switches and jumpers on devices.

continued

Protected Virtual Address Mode (Protected Mode)

One of two 80286 or three 80386 memory addressing modes. In protected virtual address mode, the 80386/80286 uses all address lines. This allows addressing of up to 16 megabytes of physical memory in an 80286 and 4 GB in an 80386. The 80286 processor's internal memory management allows addressing of an additional 1 gigabyte of virtual memory in protected mode (the 80386 up to 64 Terabytes). Protected mode addresses are specified in selector:offset format. BIOS data structures support protected mode access of all BIOS services.

PS/2-Compatible Computer

Any computer that can run software programs written for an IBM PS/2 computer. IBM PS/2 systems come in two basic varieties: Models 25 and 30 versus Models 50, 60, and 80. The former systems do not implement the Micro Channel Architecture (MCA).

Public Data

That part of an BIOS data structure which is accessible by the calling program (or the operating system).

RAM Extension

An extension to BIOS that exists as files that will be located and initialized into system RAM during the overall BIOS ROM initialization process. It can add or replace whole BIOS services or individual functions.

Real Address Mode

One of two 80286 memory addressing modes (the 80386 has three). In real address mode, the 80286 and 80386 microprocessors use 20 address lines, thus allowing memory addressing of up to 1 megabyte of physical memory (2^{20}). Real address mode does not support virtual memory addressing. Real mode addresses are specified in segment:offset format. BIOS data structures support real mode access of all BIOS services.

Reentrant Code

Reentrant code allows one copy of a given routine to be entered multiple times. Input #1 need not be completely processed before input #2 is allowed. Input #2 need not be completely processed before input #3 is allowed, and so on. All BIOS code is reentrant.

continued

Reference Diskette

In PS/2-compatible MCA-based systems, POST error recovery, access to system utilities, and system configuration are all controlled by a utility diskette, the Reference Diskette. The Reference Diskette can automatically configure a system, resolving conflicts between adapter card settings, and optimally configuring the system.

Revision Level

The BIOS ROM contains a ROM revision number that identifies the revision level of the BIOS device services contained in the ROM. By convention, BIOS extensions must examine the Device ID/Secondary Device ID pair associated with the service they are extending. It is the extension's responsibility to insure that only the service with the highest revision number is initialized.

ROM Extension

An extension to BIOS that exists in the same peripheral card ROM that contains its CBIOS counterparts. It contains manufacturer-specific BIOS device services, and is located and initialized as part of the overall BIOS ROM initialization process.

Secondary Device

An additional field used by the BIOS to make sure that the selected device's hardware level is supported.

Secondary Device ID

When more than one physical device is associated with a device ID, the two physical devices can be differentiated with a Secondary Device ID. When a unique BIOS Service is required for each physical device, the two services must be differentiated via their Secondary Device ID.

Segment

A unit of contiguous, one-dimensional address space. In real mode, these address blocks are 64K in size, referenced by one byte. In protected mode, programs can allocate segments of any size they require up to 64K.

continued

Selector

A value contained in a segment register (such as the CS, DS, SS, or ES segment registers) when in protected mode. This value determines what segment is currently being used; e.g., with CS, what segment is being used for executing code.

Single-Tasking

Single-tasking programs can only execute one program module or routine at a time. Information input into a module or routine must be processed completely before information can be input into another module.

Software

A comprehensive term used to identify all of the nonhardware components of a computer. Software includes computer programs and data.

System Board

A large circuit board that holds most of the main electronic parts of the computer.

System Board ROM

Read-only memory chips that reside on the system board and provide control information for various system components.

System Parameter Table

A common BIOS data structure 20h bytes long that describes the number of devices available in the system, the BIOS common entry points, and system stack requirements.

Task

In an 80386, a task is the execution of a single process or set of instructions to perform a particular function. It is not the same as an operating system task.

Time-Out

When the interval of time expected for a certain process (an interrupt) to occur is exceeded.

Virtual 8086 Mode (80386 only)

A way of emulating the 8086 or 8088 microprocessors on the 80386. The 8086 program runs in protected mode as a task that can run with multiple 8086 virtual tasks, as well as alongside other multiprogrammed 80386 tasks.

Additional Resources

The following books provide additional material related to the BIOS:

Dettman, Terry R. *DOS Programmer's Reference*. Carmel, IN: Que® Corporation, 1988.

Duncan, Ray. *Advanced MS-DOS Programming*. Redmond, WA: Microsoft Press, 1986.

Duncan, Ray. *IBM® ROM BIOS*. Redmond, WA: Microsoft Press, 1988.

International Business Machines Corporation. *IBM® Personal System/2™ and Personal Computer BIOS Interface Technical Reference*. Boca Raton, FL: IBM, 1987.

International Business Machines Corporation. *IBM® Personal System/2™ Model 30 Technical Reference*. Boca Raton, FL: IBM, 1987.

International Business Machines Corporation. *IBM Personal System/2™ Model 50 and 60 Technical Reference*. Boca Raton, FL: IBM, 1987.

Norton, Peter and Wilton, Richard. *The New Peter Norton Programmer's Guide to the IBM® PC and PS/2®*. Redmond, WA: Microsoft Press, 1988.

Wadlow, Thomas A. *Memory Resident Programming on the IBM PC*. Reading, MA: Addison-Wesley Publishing Co., Inc., 1987.

Wilton, Richard. *Programmer's Guide to PC® and PS/2™ Video Systems*. Redmond, WA: Microsoft Press, 1987.

Index

A

- Abbreviations, xxiii–xxiv
- ABIOService, 2, 19, 23, 131, 151, 155
- ABIOServiceCommonEntry, 2, 19–22, 131, 151–154
- ABIOService.SYS file, 27, 160
- Abort DMA Operation service, 561–562
- Access. *See* Program access
- ACK byte from keyboard controller, 300
- Acronyms, xxiii–xxiv
- Action, return codes requiring, 618
- Adapters
 - card identification for, 55
 - CMOS RAM for, 527–530
 - description files for, 34, 55, 570–571
 - and POS data, 69, 579–582
 - ROM extension for, 108
 - slots for, 55
- ADD extension class, 163, 165, 174
- Address fields for diskettes, 214
- Address line 20
 - disabling of, 482
 - enabling of, 481
- Address modes. *See* Protected address mode; Real address mode
- Addresses
 - for CMOS RAM, 443–445
 - changing
 - Global Descriptor Table and protected mode, 478
 - for fixed disk systems, 233
 - for I/O ports, 61–79
 - parallel, 39, 70–71, 74–75, 417
 - POS, 67, 69–70, 572
 - serial, 40, 71–73, 77–79, 361
 - for PICs, 56
 - for system control functions, 58–59
 - for video RAM, 309
- ADF (adapter description files) for POS data, 34, 55, 570–571
- Alarm interrupts, 441
 - canceling of, 457
 - setting of, 454–456
- Allocate Arbitration Level DMA service, 556–557
- Alt key, 266–267, 587
- Alternate Reset Fixed Disk System function, 232
- Analog monitor support, 37, 308
- Anchor pointer
 - to CDA, 85–86
 - and program access, 19–20, 151
 - relationship of, to other structures, 14
- Arbitration levels
 - allocation of, 556–557
 - and bus sharing, 545
 - deallocation of, 558
 - disabling of, 559
 - for DMA, 51–53, 548–550
 - for multiple instances of devices, 115–116
 - register for, 67
- Arbus register in DMA controller, 49, 53, 551

A, cont'd

- Arithmetic calculations, coprocessor for, 32
- Assignment of interrupt requests, 57
- Attention register, fixed disk adapter, 74
- Attention return code, 25, 141, 618
- Attribute bytes, video, 314
 - controller for, 36, 307
 - and palette register, 312, 345, 347
 - registers for, 75
- Autoinitialization of DMA controller, 49, 547
- Auxiliary device, enabling and disabling of, 296

B

- Banks, RAM, 37
- Base memory, CMOS RAM for, 526
- Battery-backed CMOS RAM. *See* CMOS RAM
- Battery Bad return code, 532
- Baud rate
 - generator of, 41, 362
 - setting of, 378-379
- Bimodal support for interrupt handlers, 146
- BIOS
 - extensions, 157
 - ROM, 1
- Blocks, print
 - printing of, 427-431
- Blocks, video
 - of color registers, reading of, 354-355
- Booting and initialization, 12, 95
- Break Interrupt for serial ports, 392
- Break key, 267, 589

- Buffers, serial port
 - receive, 389-391, 393-396
 - transmit, 379-382, 384-386, 399
- Buffers, video
 - video, size parameters for, 321
- Build Initialization Table function, 13, 96, 100-102
 - for RAM extensions, 178-179
 - for ROM extensions, 177
- Build System Parameters Table function, 13, 96, 98-99
 - for RAM extensions, 179
 - for ROM extensions, 177
- Burst mode DMA transfer, 51, 548
- Bus sharing for DMA, 47, 545
- BUSY signal, math coprocessor, 32-33

C

- CALL for control transfer, 17, 134
- Calling of functions, 127
 - and ABIOSCall, 155
 - and ABIOSCommonEntry, 152-154
 - control transfer, 134-135
 - Default Interrupt Handler for, 147-148
 - by handlers, 9
 - hardware interrupt handlers for, 142-146
 - processing model for, 128-131
 - and program access, 151
 - and Request Block initialization, 132-133
 - return code handling in, 139-141
 - time-out handlers for, 149-150
 - and transfer conventions, 136-138
- Cancel Alarm Interrupt Real Time Clock Service, 457
- Cancel Periodic Interrupt Real Time Clock Service, 461
- Cancel Print Block Parallel Port Service, 430-431

C, cont'd

- Cancel Serial Communications Service, 407–408
- Cancel Update–Ended Interrupt Real Time Clock Service, 464
- Capacity, diskette, 193
- Card selected feedback register, DMA, 67
- CBIOS
 - accessibility to, 2
 - and compatibility, 1
 - and Diskette Service, 189–190
 - extensions for, 158
 - and Fixed Disk Service, 232, 245
 - and serial ports, 367
 - structures of, compared to A BIOS, 11
- CDA. See Common data area
- Central Arbitration Control
 - for DMA controller, 549
 - with Micro Channel, 33, 51
- CGA (Color Graphics Adapter)
 - compatibility with, 309–310, 325
 - and palette register, 312
- Change Line Status signal, 191
 - emulation of, 186–188
 - reading of, 226–227
 - testing of, 204
- Channel extension connectors, 33
- Channel position select register, POS, 67
- Channels, DMA, 54, 551
 - flags for, 53, 551
- Character Block Specifier field, 319–320
- Character block to load field, 327
- Character block to select field, 328
- Characters
 - blocks of, selection of, 337–338
 - and fonts, 314
 - generators of, 313
 - serial support for number of, 40, 361
- Checksum
 - power–on, CMOS RAM for, 528
 - recomputation of, 540–541
- Checksum Invalid return code, 532
- Clear Byte Pointer, DMA, 61, 69
- Clear Mask Register, DMA, 62, 69
- Clear video buffer flag, 326
- Clock generator chip, 42
- Clock interrupts, 441
- CMOS RAM
 - ADF file data stored on, 34, 55
 - areas for, 46
 - device ID for, 4, 153
 - data for, 525–527
 - error handling for, 532
 - for Error Log Service, 607–608
 - extended, 528–531
 - hardware environment for, 524
 - and NMIs, 449
 - port addresses for, 66
 - and POS, 569, 575–582
 - for power–on password, 60, 583
 - reading of, 45
 - for real time clock, 45, 443, 524–525
 - return codes for, 620
 - services for, 523
 - Read CMOS RAM, 536–537
 - Read Device Parameters, 534
 - Recompute Checksum, 540–541
 - Return Logical ID Parameters, 533–534
 - Write to CMOS RAM, 538
 - writing to, 45
- Color Graphics Adapter
 - compatibility with, 309–310, 325
 - and palette register, 312

C, cont'd

- Color register
 - reading of, 349–350, 354–355
 - writing to, 351–353, 356–358
- Colors, video, 309–311, 325
- default, 313
- Common Data Area, 10, 84–86
 - and Anchor Pointer, 19
 - building of, 103–105
 - and control transfer, 18, 135–136
 - data pointers in, 85–86, 103, 107, 109, 113, 475
 - in DB and FTT initialization routines, 108
 - device data pointers in, 107
 - and extensions, 173–174
 - and initialization, 11–13, 95
 - and Initialization Table, 100–101
 - Logical IDs in, 15, 152
 - null entries in, 85, 112
 - and program access, 151
 - and protected mode, 110–112
 - relationship of, to other structures, 14, 82
 - and REPLACE extensions, 164
 - for stack frame loading, 137
- Common Entry Routines, 9 17–18
 - and control transfer, 18, 134–135
 - device service for, 109
 - with function calling, 131
 - logical ID for, 104–105
- Common Interrupt Routine, 9
 - address of, 113
 - with function calling, 131
 - pointer for, 98
 - for time-out handlers
- Common Start Routine, 9
 - address for, 113
 - and control transfer, 18
 - with function calling, 131
 - pointer for, 98
- Common Time-out Routine, 9
 - address of, 113
 - with function calling, 131
 - pointer for, 98
- Compatibility, 1–2
 - diskette, 194–195
 - video display, 36
 - video RAM, 308
- Configuration
 - for CMOS RAM, 525–531
 - diskette drive, 35
 - system, reading of, 473–474
- Configuration control register, diskette controller, 77
- Conforming Bit for pointers, 111
- Connectors for Micro Channel, 34
- Continuous Keyboard Read service, 280–281
- Continuous multistaged functions, 6
 - processing model of, 128
- Continuous read, NMI, 492–493
- Control. See Transfer conventions
- Control addresses for parallel ports, 39, 417
- Control keys, 266, 588
- Control Port, address for, 71, 74–75
- Control register, fixed disk adapter, 73
- Control word register, PIT, 65
- Controllers
 - CRT, 36, 74, 76, 307
 - diskette, 35, 193
 - DMA, 545–546
 - fixed disk, 35, 57, 235
 - keyboard, 38, 261
 - serial port, 361
- Coprocessors, 32–33
 - interrupt request for, 57
 - port address for, 69
- Count control for DMA controller, 50, 546–547

C, cont'd

- CPU
 - clock generator for, 42
 - and functions, 5
- CRT controller, 36, 307
 - index register for, 76
 - port addresses for, 74, 76
- Current Serial Port Status field
 - for receive, 389–390, 392–396
 - for transmit, 380, 382–386
- Cursor
 - in graphics modes, 310, 326
 - keys for, 266, 588
 - size of, and fonts, 342
- Cylinders, fixed disk, 236–237

D

- DAC. *See* Digital-to-analog converter
- Data addresses for parallel ports, 39, 70, 74–75, 417
- Data length, changing of, 202
- Data pointers
 - in CDA, 85–86, 103, 107, 109, 113, 475
 - for CMOS RAM, 536, 538
 - in Initialization Table, 101
 - relationship of, to other structures, 14
 - in Request Block structure table, 123–124
- Data registers, diskette controller, 76
- Data string length for keyboard controller, 295
- Data structures, 10–11, 81–83, 94
 - Common Data Area, 84–86
 - Device Block, 89–93
 - Function Transfer Table, 87–88
- Data transfer
 - diskette rates for, 193
 - with DMA devices, 50
 - for fixed disk systems, 235
- Date
 - reading of, 465–466
 - writing of, 466–467
- Daylight Savings Update field, 452
- DB. *See* Device Blocks
- Deallocate Arbitration Level DMA Service, 558
- Default colors, 313
- Default fonts, 310, 314, 325, 327–328
- Default Interrupt Handlers, 4, 144, 147–148
 - for Diskette Service, 196
 - for Fixed Disk Service, 239
 - for Keyboard Service, 269
 - for Parallel Port Service, 419
 - for Pointing Device Service, 498
 - for Real Time Clock Service, 446
 - for Serial Communications Service, 364
 - for System Timer Service, 438
 - for Video Service, 316
- Default sector size, changing of, 202
- Defect map, fixed disk, 236–237
- Delay, typematic, 286–288
- Descriptors, selector, 111
- DevHlp services, 2, 19–23, 131, 151
- Device Blocks, 10, 82, 89–93
 - and ADD extensions, 165
 - and control transfer, 18, 135–136
 - for extensions, 26, 159, 173–175
 - initialization of, 11–13, 96, 106–109, 113
 - and Initialization Table, 100–101
 - and MODIFY extensions, 167
 - for multiple instances of devices, 115–116
 - pointers to, 84–86
 - characteristics of, 111
 - initialization of, 103

D, cont'd

- and Logical IDs, 15
- protected mode, 110
- relationship of, to other structures, 14
- for REPLACE extensions, 164
- Device Control Flag, 326, 356
- Device Error return code, 25, 141
- Device Error With Time-out return code, 25, 141
- Device IDs, 15, 21
 - and BIOSCommonEntry, 152
 - CDA for, 85
 - in Device Block, 90, 93
 - for extensions, 169
 - in Initialization Table, 100
 - in Return Logical ID Parameters function, 20
- Device In Use return code, 25, 141
- Devices
 - bit size of, and DMA controller, 50
 - CDA data pointers for, 85
 - data area for, in Device Block, 92
 - extensions for, 162
 - memory for, 14
 - multiple instances of, 115-116
 - return codes for, 25, 139-141
 - supported, 4, 21
 - See *a/s/o* Device Blocks; Device IDs; Read Device Parameters functions
- Diagnostic status for CMOS RAM, 525
- Digital input register, diskette controller, 77
- Digital output register, diskette controller, 76
- Digital-to-analog converter, 36-37, 307-308
 - and palette register, 312, 345, 347
 - port addresses for, 75
 - reading of color register in, 349-350
 - writing to color register in, 351-353, 356-358
- Direct Memory Access Service, 47, 50, 543-544
 - device ID for, 4, 153
 - error handling for, 552
 - for fixed disks, 231
 - hardware environment for, 545-552
 - return codes for, 619-620, 622
 - services for, 553-568
 - Abort DMA Operation, 561-562
 - Allocate Arbitration Level, 556-557
 - Deallocate Arbitration Level, 558
 - Disable Arbitration Level, 559
 - DMA Transfer from Memory to I/O, 563-564
 - DMA Transfer Status, 560
 - Load DMA Controller Parameters, 567-568
 - Read Device Parameters, 555
 - Read from I/O and Write to Memory, 565-566
 - Return Logical ID Parameters, 553-554
 - See *a/s/o* DMA controller
- Disable Address Line 20 System Services, 482
- Disable Arbitration Level DMA Service, 559
- Disable Diskette Service, 206-207
- Disable Keyboard Service, 278-279
- Disable NMI Service, 491
- Disable Pointing Device Service, 507-508
- Disable standard function, 4
- Discrete multistaged functions, 6, 128

D, cont'd

- Diskette Change Line signal, 191
 - emulation of, 186–188
 - reading of, 226–227
 - testing of, 204
 - Diskettes
 - change line emulation for, 186–188
 - CMOS RAM for, 526
 - controller registers for, 76–77
 - device ID for, 4, 153 hardware for, 35, 193
 - error handling for, 195
 - interrupt request for, 57
 - return codes for, 619–626
 - services for, 189–192
 - Default Interrupt Handler, 196
 - Disable Diskette, 206–207
 - Format Diskette, 214–217
 - Interrupt Status, 229–230
 - Read Change Line Signal Status, 226–227
 - Read Device Parameters, 199–201
 - Read Diskette, 208–210
 - Read Media Parameters, 221–222
 - Reset/Initialize Diskette Subsystem, 203–205
 - Return Logical ID Parameters, 197–198
 - Set Device Parameters, 202–203
 - Set Media Type for Format, 223–225
 - Turn Diskette Motor Off, 228–229
 - Verify Diskette Sectors, 218–220
 - Write to Diskette, 211–213
 - support for, 116
 - Divisor latch, serial port, 71, 77
 - DMA controller, 47–54, 545–546
 - arbitration levels for, 33, 52
 - direct accessing of, 50, 547
 - port addresses for, 61, 66, 68–69
 - See also Direct memory access
 - DMA Transfer from Memory to I/O, 563–564
 - DMA Transfer Status, 560
 - Door, diskette, and Diskette Change Line signal, 191
 - DOS, ROM BIOS interrupt vectors for, 157
 - DOS compatibility box, 22–23, 154–155
 - Double density diskettes, 194
 - Drive types, diskette, 194
 - Dynamic reassignment of arbitration levels, 53, 551
- ## E
- EGA (Enhanced Graphics Adapter)
 - compatibility with, 309–310, 325
 - and palette register, 312
 - Emulation
 - of Change Line Status signal, 186–188
 - of video modes, 310, 325
 - Enable Address Line 20 System Services, 481
 - Enable FIFO Control Serial Communications Service, 412–413
 - Enable Keyboard Service, 276–277
 - Enable Keyboard Security Service, 593–594
 - Enable NMI Service, 490
 - Enable Pointing Device Service, 505–506
 - Enable Speaker System Services, 483–484
 - Enable standard function, 4

E, cont'd

- Enabling of transmit interrupts, 379
- End-of-interrupt processing, 144
- Enhanced Graphics Adapter
 - compatibility with, 309-310, 325
 - and palette register, 312
- Enhanced Load Text Mode Font Video Service, 342-344
- Entry pointers for control transfer, 17, 134
- Entry points
 - for extensions, 168-169
 - vectors to, 87-88
- Entry routines for transferring control, 9, 17, 131
- Environment. *See* Hardware environment
- EOI (end-of-interrupt) processing, 144
- Equipment installed, CMOS RAM for, 526
- ERRNUM pointer, 607
- Error handling
 - for CMOS RAM Service, 532
 - of DBs and FTTs, 108
 - for Direct Memory Access Service, 552
 - for Diskette Service, 195
 - for Error Log Service, 609
 - for Fixed Disk Service, 238
 - for Keyboard Security Service, 589
 - for Keyboard Service, 268
 - for math coprocessor, 32-33
 - for Nonmaskable Interrupt Service, 487
 - and operating system transfer convention, 138
 - for Parallel Port Service, 418
 - for Pointing Device Service, 497
 - for POS Service, 573
 - for Real Time Clock Service, 445
 - for Serial Communications Service, 363
 - for switching to protected mode, 480
 - for switching to real mode, 477
 - for System Services, 470
 - for System Timer Service, 437
 - for Video Service, 315
- Error Log Service
 - CMOS RAM for, 531
 - device ID for, 4, 153
 - error handling for, 609
 - extended CMOS RAM for, 607-608
 - services for, 605-606
 - Read Error Log, 612-613
 - Return Logical ID Parameters, 610-611
 - Write to Error Log, 614-615
- ESDI fixed disk controller, 35
- Expansion memory, CMOS RAM for, 526, 528
- Extended CMOS RAM
 - data area for, 46, 524
 - for Error Log Service, 607-608
 - port address for, 66
 - for POS Service, 569
 - services for, 528-531
- Extended function execute register, DMA, 62
- Extended function register, DMA, 62
- Extended mode
 - DMA, 54, 552
 - parallel port, 39, 417
- Extensions, 26-27, 157-161
 - Device Block and FTT routines for, 173-175
 - examples, 180-188
 - headers for, 168-170
 - and initialization, 108-109, 117
 - Initialization Table entry routine for, 171-172
 - non-intrusive interception, 181-183

E, cont'd

nonstaged function, redirection of, 184–185
RAM, 178–179
recommendations for, 162–167
ROM, 176–177
service code for, 176
staged function, redirection of, 186–187

F

Feature Configuration Table, 473
Feature control register, VGA, 74–76
FIFO control, 412–413
 serial port register for, 72, 78
File headers for extensions, 26, 159, 168–169
Filter bytes, writing of, 601–604
Fixed Disk Interrupt Status service, 257–258
Fixed disks
 adapters for, 73–74
 CMOS RAM for, 526
 controller for, 35, 57, 235
 device ID for, 4, 153
 DMA arbitration level for, 549
 error handling for, 238
 extensions for, 163
 hardware environment for, 235
 parameters table for, 236–237
 port address for, 70
 return codes for, 620–626
 services for, 231–234
 Default Interrupt Handler, 239
 Fixed Disk Interrupt Status, 257–258
 Read Device Parameters, 242–244
 Read Fixed Disk, 249–250

Reset/Initialize Fixed Disk, 244–248
Return Logical ID Parameters, 240
Verify Fixed Disk Data, 255–256
Write and Verify Fixed Disk, 253–254
Write to Fixed Disk, 251–252

Flags

Device Control, 326
in DMA controller, 48, 53, 546, 551
for DOS compatibility box, 22, 154
for Start Routines, 143

Floating point numbers, coprocessor for, 32–33

Fonts

default, 310, 314, 325, 327–328
extension for, 184–185
retrieval of information about, 331–332
ROM-resident, 313–314, 327
text mode, 325, 339–344

Format Diskette Service, 214–217

Framing errors for serial ports, 392

FTT. *See* Function Transfer Tables

Function count in FTT, 88

Function Pointers, relationship of, 14

Function Transfer Tables, 10, 82, 87–88

 and control transfer, 135
 for extensions, 26, 173–175
 initialization of, 11–13, 103, 106–109, 113
 and Logical IDs, 15
 pointers to, 17–18, 84–85
 and ADD extensions, 165
 characteristics of, 111
 and control transfer, 18, 136
 for extensions, 159, 162, 173–175
 and initialization, 95

F, cont'd

- and Initialization Table, 100-101
- Logical ID 2 entry in, 114
- and MODIFY extensions, 167
- and protected mode, 111-112
- relationship of, to other structures, 14
- for REPLACE extensions, 164
- relationship of, to other structures, 14

Functional parameters, Request Block, 121-123

Functions

- address pointers to, 82
- entry points for, extensions control of, 26
- relationship of, to other structures, 14
- and request blocks, 15, 122
- standard, 4
- suspension of, 5, 22-23, 154
- transferring control to, 17-18
- See *also* Calling of functions

G

- Game port address, 70
- Gap length, changing of, 202
- Graphics controller, 36, 307
- Graphics registers, VGA, 75
- Gray scale summing flag, 326, 356

H

- Handlers, calling by, 9
- Hardware devices
 - as physical devices, 15
 - relationship of, to other structures, 14
- Hardware environment, 29-30

- for CMOS RAM Service, 46, 524
- for Direct Memory Access Service, 545-552
- for Diskette Service, 193-195
- DMA controller, 47-54
- for Fixed Disk Service, 235
- I/O devices, 35-42
- I/O port list, 61-79
- for Keyboard Security Service, 585
- for Keyboard Service, 261
- math coprocessors, 32-33
- Micro Channel, 33-34
- microprocessors, 31
- NMI masks, 60
- for Parallel Port Service, 39, 417-418
- for Pointing Device Service, 497
- for POS Service, 571-572
- power-on passwords, 60
- Programmable Interrupt Controller, 56-57
- Programmable Option Select, 55
- for Real Time Clock Service, 443
- for Serial Communications Service, 40-41, 361-363
- system control port definitions, 58-59
- system time-related devices, 42-46
- for System Timer Service, 436-437
- for Video Service, 36-37, 307-308

Hardware interrupts

- for Diskette Service, 193
- for fixed disks, 35
- handlers for, 142-146
- maskable, PICs for, 56
- for multistaged functions, 7-8, 129-130
- for parallel ports, 39
- by PIT, 43
- sharing of, 144-145

Hardware time-out

- for function driving, 130
- vs. time period stages, 8

H, cont'd

Headers for extensions, 26, 159, 168–169

Heads

fixed disk, 236–237

number of, in disk formatting, 214

High density diskettes, 194

I

Identification code

for keyboard, 272

for pointing devices, 519–521

Identity BIOS service, CBIOS vs. ABIOS, 11

Idle CPU time, 5

IDs. *See* Device IDs; Logical IDs

Initialization, 95–97, 117

ABIOS, 11–16

of Common Data Area, 96, 103–105

of DAC color registers, 349, 351

of Device Blocks and FTTs, 11–13, 96, 106–109, 113

of Initialization Table, 96, 100–102

of logical ID 2, 113–115

and multiple instances of devices, 115–116

of palette register, 345

of Pointing Device Service, 503

of protected mode tables, 96, 110–112

of RAM extensions, 178–179

of Request Block, 11, 132–133

of return code fields, 143

of ROM extensions, 176–177

of serial ports, 361

of stack frames, 20, 23

of System Parameters Table, 96, 98–99

of video hardware, 324–330

Initialization Command Words, PIC, 63

Initialization Table

building of, 13, 96, 100–102
for extensions, 26, 159, 168, 171–172

first Device ID entry for, 114

Initialize DAC to default values flag, 326

INMOS G171 DAC chip, 36, 307

Input rules for Request Blocks, 16, 133

Input status register, VGA, 74, 76

Intel 8042 keyboard controller, 38, 261

for keyboard security, 585

for Pointing Device Service, 497

port address for, 66

Intel 8237 DMA Controller, 47, 545–546

Intel 8254A Programmable Interval Timer, 42–44, 436

Intel 8259A Programmable Interrupt Controller, 56–57

Intel 80286/80386/80386SX microprocessors, 31

Intel 80287/80387 math coprocessor, 32

Intel 82284 clock generator, 42

Internal calls

device ID for, 4, 153

and Logical ID 2, 113

logical IDs for, 104

Interrupt Status Diskette Service, 229–230

Interrupt status field, fixed disk, 257

Interrupt Status Register, fixed disk adapter, 74

Interrupts, 9, 131

assignment of requests for, 57

diskette handlers for, 196

enable register for, 71, 77

I, cont'd

- and extensions, 162
- identification register for, 71, 77
- levels of, 33, 39–40, 56, 261, 361
- mask register for, 63–64
- for multiple instances of devices, 115
- operating system handling of, 138
- pointer to, 87–88, 98
- for real time clock, 441
- for serial port, 361
- and service registers, PIC, 62
- and staged function redirection, 186
- update-ended, 462–464
- See also Hardware interrupts;
Nonmaskable interrupts

Invalid Function return code, 25, 141

Invalid Logical ID return code, 25, 141

Invalid Request Block Length return code, 25, 141

Invalid Service-Specific Parameter return code, 25, 141

Invalid Unit Number return code, 25, 141

Invocation byte, writing of, 597–598

I/O address width with Micro Channel, 33

I/O ports

- for CMOS RAM, 45, 66
- for diskette, 76–77
- for DMA controller, 47, 66, 68–69
- for error logs, 608
- for fixed disk, 70, 73–74
- for game port, 70
- for keyboard, 65–66
- for manufacturing checkpoint port, 79
- for math coprocessor, 69
- for parallel port, 70–71, 74, 75
- for POS, 67, 69, 572
- for PIC, 62–64, 68, 572

- for PIT, 64–65
- for serial port, 71–73, 77–79
- for video, 74–76

See also Parallel ports; Serial ports

I/O privileges, 112

Isolation, BIOS for, 1

Italicizing fonts, extension for, 184–185

K

Keyboard

- device ID for, 4, 153
- error handling for Keyboard Service, 268
- error handling for Keyboard Security Service, 589
- hardware environment for Keyboard Security Service, 585
- hardware for, 38, 261
- hardware environment for Keyboard Service, 261
- interrupt request for, 57
- layout of, 262
- return codes for, 619–622, 625
- services for Keyboard Security Service, 583–584
- Enable Keyboard Security, 593–594
- Read Device Parameters, 592
- Return Logical ID Parameters, 590–591
- Write Filter Byte 1, 601–602
- Write Filter Byte 2, 603–604
- Write Invocation Byte, 597–598
- Write Match Byte, 599–600
- Write Password, 595–596
- services for Keyboard Service, 259–260
- Continuous Keyboard Read, 280–281
- Default Interrupt Handler, 269
- Disable Keyboard, 278–279

K, cont'd

- Enable Keyboard, 276–277
 - Read Keyboard ID Bytes, 272–273
 - Read Keyboard LED Status, 282–283
 - Read Keyboard Scan Mode, 289–291
 - Reset/Initialize Keyboard, 274–275
 - Return Logical ID Parameters, 270
 - Set Keyboard LED Status, 284–285
 - Set Keyboard Scan Code Mode, 292–295
 - Set Typematic Rate and Delay, 286–288
 - Write Command(s) and Data to Keyboard, 300–303
 - Write Command(s) to Keyboard Controller, 295–299
 - scan codes for Keyboard Service, 263–267, 586–587
 - system scan codes for Keyboard Security Service, 586–588
 - and time-out handling, 149
- Keyboard/auxiliary data port, 65
- ## L
- Landing zone, fixed disk, 236–237
 - LED, keyboard, status of, 282–285
 - Length
 - data string, for keyboard controller, 295
 - of extensions, 168
 - in Request Block structure, 122
 - Level-sensitive interrupts
 - with Micro Channel, 33
 - for parallel port, 39
 - and PICs, 56
 - and serial port, 40, 361
 - Levels of priority for DMA controller, 549
 - Lifespan of Request Blocks, 15, 132
 - Line control
 - register for, 72, 78
 - setting of, 375–377
 - Line status register, 73, 79
 - Load DMA Controller Parameters service, 567–568
 - Load Text Mode Font Video Service, 339–341
 - Local Descriptor Table and protected mode, 478
 - Locate BIOS function and service, CBIOS vs. A BIOS, 11
 - Logical devices vs. physical devices, 15
 - Logical ID 2, initialization of, 109, 113–114
 - Logical IDs, 15
 - and A BIOSCommonEntry, 20, 152
 - and ADD extensions, 165
 - assignment of, 103
 - in bimodal environment, 146
 - and CDA, 85–86, 104–105
 - and control transfer, 135
 - and Default Interrupt Handlers, 147
 - in Device Blocks, 90–93
 - for diskette drives, 190, 197–198
 - for fixed disk systems, 232
 - FTT pointer for, 87
 - and hardware interrupt handlers, 142
 - in Initialization Table, 100
 - interrupts for, 144–145
 - and MODIFY extensions, 167
 - for multiple instances of devices, 115
 - relationship of, to other structures, 14

L, cont'd

- in Request Block structure table, 122
- for serial ports, 40
- See *also* Device IDs; Return Logical ID Parameters functions

M

- Manufacturers, fixed disk, 236-237
- Manufacturing checkpoint port, 79
- Mask register, DMA, 61, 68
- Maskable hardware interrupts, PICs for, 56
- Masks, NMI, 60
- Master Clear, DMA, 61, 69
- Match byte, writing of, 599-600
- Math coprocessors, 32-33
 - interrupt request for, 57
 - port address for, 69
- MDA (Monochrome Display Adapter), compatibility with, 309-310, 325
- Memory
 - for CDA, 103, 110
 - device, 14, 84
 - for Initialization Table, 102
 - transferring data from, 563-564
 - transferring data to, 565-566
 - video, 36-37, 307-308, 314
- Memory address registers, DMA, 61, 68, 546
- Memory refresh, arbitration level for, 52
- Messages, error. See Return Codes, Appendix A
- Micro Channel, 33-34
 - arbitration levels for, 51
- Microprocessor mode and hardware interrupts, 146
- Microprocessors, 31
 - address line 20, 481-482
 - and DMA, 50, 547
 - system, arbitration level for, 52
- Miscellaneous output register, VGA, 75-76
- Mode control field for DMA controller, 49, 547
- Mode Register, DMA, 61, 69
- Model byte for extensions, 169
- Models, PS/2, identification of, 473
- Modem Status Serial Communications Service, 404-406
- Modems
 - control register for, 72, 78
 - controlling of, 374-375
 - line status of, 409
 - status register for, 73, 79
 - See *also* Serial ports
- Modes
 - keyboard scan, 289-294
 - real time clock, 452
 - system timer, 43
 - video, 309-313, 324-330
 - See *also* Protected address mode; Real address mode
- MODIFY extension class, 163, 166-167, 174
- Monitor support, 37, 308, 311-313
- Monochrome Display Adapter, compatibility with, 309-310, 325
- Motherboard IDs, reading of, 473
 - See *also* system board
- Motor, diskette, 204, 228-229
- Motorola MC146818A Real Time Clock, 45-46, 443, 524-525
- Mouse, 495
 - interrupt request for, 57
 - return codes for, 619, 622
- Multi-byte scan codes, 264-265, 586
- Multiple instances of devices, 115-116

M, cont'd

- Multiscan monitor support, 37, 308
- Multistaged functions, 7–8
 - driving of, 129
 - time–period handling of, 130
- Multistaged processing/multitasking operating systems, 5–6
- Multitasking operating system, 31

N

- National Semiconductor
 - 8250/16640/16650 serial port controller, 40–41, 361
- NEC 765 diskette controller, 35, 193
- NMI. *See* Nonmaskable interrupts
- NMI Continuous Read service, 492–493
- Non–intrusive interception by extensions, 181–183
- Nonmaskable interrupts
 - arbitration level for, 52
 - and CMOS RAM access, 449
 - device ID for, 4, 153
 - error handling for, 487
 - mask for, 60
 - services for, 485–486
 - Disable NMI, 491
 - Enable NMI, 490
 - NMI Continuous Read, 492–493
 - Return Logical ID Parameters, 488
- Nonstaged function, redirection of, 184–185
- Not My Interrupt return code, 25, 141, 143–144, 618
- Not set state for Diskette Change Line signal, 191
- Null CDA entries, 85, 112
- Null stripping serial port mode, 392

- Number of Initialization Table Entries
 - for extensions, 168–169
 - in Systems Parameters Table, 98–99
- Numbering of logical IDs, 104
- Numeric exception error handling interrupt, 33
- Numeric keypad, 267, 588

O

- Operating system
 - CDA building by, 103–104
 - and DBs and FTTs, 106, 109
 - isolation of, by BIOS, 1
 - transfer convention of, 18, 137–138
- Out of paper error, Parallel Printer Service, 427
- Output rules for Request Blocks, 16, 133
- Overrun errors for serial ports, 392

P

- Page table address registers, 66
- Palette register
 - and DAC, 312
 - reading of, 345–346
 - writing to, 347–348
- Parallel ports
 - addresses for, 39, 70–71, 74–75, 417
 - device ID for, 4, 153
 - error handling, 153
 - hardware environment for, 39, 417–418
 - interrupt request for, 57
 - return codes for, 619–620, 622, 626
 - services for, 415–416
 - Cancel Print Block, 430–431

P, cont'd

- Default Interrupt Handler, 419
 - Print Block, 427-429
 - Read Device Parameters, 421-422
 - Reset/Initialize Parallel Port, 425
 - Return Logical ID Parameters, 420-421
 - Return Print Status, 432-433
 - Set Device Parameters, 423-424
- Parallel-to-serial conversions, 41, 362
- Parameters
 - errors in, 140
 - Request Block, 121
 - table of, for Fixed Disk Service, 236-237
- Parity, serial support for, 40, 361, 392
- Pass parameters, CBIOS vs. A BIOS, 11
- Passwords
 - CMOS RAM for, 526, 528
 - with keyboard, 296, 583-584
 - power-on, 60
 - writing of, 595-596
- Patching, recommendations for, 162
- Pause key, 267, 589
- PC-type adapters and Micro Channel, 33
- Periodic Interrupt Rate field, 449
- Periodic interrupts, clock timer, 441
 - canceling of, 458-460
 - setting of, 461
- Phoenix 8042 Advanced Keyboard/Mouse Controller Firmware, 289, 292
- Physical addresses, conversion of, 109
- Physical devices
 - vs. logical devices, 15
 - supported, 153
- Physical DMA channels, 53, 551
- PhysToVirt call, 22
- PIC. *See* Programmable Interrupt Controller, 56-57
- PIO (programmed I/O) for DMA controller, 49, 547
- PIT. *See* Programmable Interval Timer
- Pointer field definitions, 379, 389
- Pointers
 - and CDA, 84-86, 103, 107, 109, 113, 475
 - for data area, 103
 - to functions, 82
 - pushing of, onto stack, 134
 - receive, 389-391, 393-396
 - transmit, 379-382, 384-386, 391, 399
 - See also* Anchor pointer; Device Blocks, pointers to; Function Transfer Tables, pointers to
- Pointing Device Continuous Read, 509-511
- Pointing devices
 - controller for, 38
 - device ID for, 4, 153
 - error handling for, 497
 - hardware environment for, 497
 - services for, 495-496
 - Continuous Read, 509-511
 - Default Interrupt Handler, 498
 - Disable Pointing Device, 507-508
 - Enable Pointing Device, 505-506
 - Read Device Parameters, 500-501
 - Read Pointing Device Identification Code, 519-521

P, cont'd

- Reset/Initialize Pointing Device, 503–504
- Return Logical ID Parameters, 499–500
- Set Resolution, 514–515
- Set Sample Rate, 512–513
- Set Scaling Factor, 516–518
- Polling of Request Block, 142–143
- Port pairs, logical ID, in device block, 91
- Portability, interface for, 30
- Ports
 - for CMOS RAM, 45, 66
 - for diskette, 76–77
 - for DMA controller, 47, 66, 68–69
 - for error logs, 608
 - for fixed disk, 70, 73–74
 - for game port, 70
 - for keyboard, 65–66
 - for manufacturing checkpoint port, 79
 - for math coprocessor, 69
 - for parallel port, 70–71, 74, 75
 - for POS, 67, 69, 572
 - for PIC, 62–64, 68, 572
 - for PIT, 64–65
 - for serial port, 71–73, 77–79
 - for video, 74–76
 - See also Parallel ports; Serial ports
- POS. See Programmable Option Select
- Power-on passwords, 60
 - for keyboard security, 583
- Power-on self test (POST)
 - ADF files read during, 34, 55
 - password access during, 60
- Print Block Parallel Port Service, 427–429
- Print screen key, 267, 589
- Printer, resetting of, 425
- Printer Interrupt Time-out field, 421
- Printer Status, 425, 427, 430
 - returning of, 432–433
- Private data in Device Block, 90
- Privileges, I/O, 112
- Processing model, 5–9, 128–131
- Program access
 - with ABIOSCALL, 2, 19, 23, 131, 151, 155
 - with ABIOCOMMONENTRY, 19–22, 131, 151–154
- Programmable baud rate generator, 41, 362
- Programmable Interrupt Controller, 56–57
 - port addresses for, 62–64, 67–68
- Programmable Interval Timer, 42–44, 436
 - port addresses for, 64–65
- Programmable Option Select, 55
 - CMOS RAM for, 527–530
 - device ID for, 4, 153
 - error handling of, 573
 - hardware environment for, 571–572
 - with Micro Channel, 33–34
 - parallel port extended mode, 39, 417
 - port addresses for, 67, 69–70
 - services for, 569–570
 - Read POS Data from an Adapter, 579–580
 - Read Stored POS Data from CMOS RAM, 575–576
 - Return Logical ID Parameters, 574–575
 - Write Dynamic POS Data from an Adapter, 581–582
 - Write Stored POS Data from CMOS RAM, 577–578
- Programmed I/O for DMA controller, 49, 547

P, cont'd

Protected address mode, 31
and CDA, 84
environments for, building of, 13
and math coprocessor, 32
switching to, 478-480
tables for, building of, 110-112

Public data in Device Block, 89

PUSH for control transfer, 17, 134

R

RAM extensions, 27, 160
area for, 180
header for, 169
initialization of, 108, 178-179
See also CMOS RAM; Memory

Raw system scan codes, 280

Read Block of Color Registers Video Service, 354-355

Read Change Line Signal Status Diskette Service, 226-227

Read CMOS RAM Service, 536-537

Read DAC Color Register Video Service, 349-350

Read Device Parameters functions, 4
for CMOS RAM Service, 534
for Direct Memory Access Service, 555
for Diskette Change Line signal, 191
for Diskette Service, 199-201
for Fixed Disk Service, 242-244
for Keyboard Security Service, 592
for Parallel Port Service, 421-422
for Pointing Device Service, 500-502
for Real Time Clock Service, 449-451
for Serial Communications Service, 367-369
for Video Service, 319-322

Read, Diskette Service, 208-210
and Diskette Change Line signal, 191

Read, Error Log Service, 612-613

Read, Fixed Disk Service, 249-250

Read from I/O and Write to Memory, DMA Service, 565-566

Read Keyboard ID Bytes, Keyboard Service, 272-273

Read Keyboard LED Status, Keyboard Service, 282-283

Read Keyboard Scan Mode, Keyboard Service, 289-291

Read Media Parameters Diskette Service, 221-222

Read Palette Register Video Service, 345-346

Read Pointing Device Identification Code, 519-521

Read POS Data from an Adapter, 579-580

Read standard function, 4

Read Stored POS Data from CMOS RAM, 575-576

Read System Configuration, System Services, 473-474

Read Time and Date, Real Time Clock Service, 465-466

Read/write counters, PIT, 64-65

Reading
from CMOS RAM, 45
of system control ports, 58

Real address mode, 31
and CDA, 84
for initialization, 12, 95
and math coprocessor, 32
switching to, 475-477

R, cont'd

- Real time clock, 45–46, 524–525
 - data for, 444–445
 - device ID for, 4, 153
 - error handling for, 445
 - hardware environment for, 443
 - interrupt request for, 57
 - return codes for, 619
 - services for, 495–496
 - Cancel Alarm Interrupt, 457
 - Cancel Periodic Interrupt, 461
 - Cancel Update–Ended Interrupt, 464
 - Default Interrupt Handler, 446
 - Read Device Parameters, 449–451
 - Read Time and Date, 465–466
 - Return Logical ID Parameters, 447–448
 - Set Alarm Interrupt, 454–456
 - Set Device Parameters, 452
 - Set Periodic Interrupt, 458–460
 - Set Update–Ended Interrupt, 462–463
 - Write Time and Date, 466–467
- Receive Serial Communications Service, 389–398
- Recompute Checksum CMOS RAM Service, 540–541
- Redirection
 - of cascade, interrupt request for, 57
 - of nonstaged functions, 184–185
 - of staged functions, 186–188
- Reference diskette
 - ADF files read by, 34, 55
 - for passwords, 583–584
- Relative Block Addresses for fixed disk systems, 233
- REPLACE extension class, 163–164, 174
- Request Blocks, 10, 94, 119–120, 125, 128
 - and ABIOSTCommonEntry, 20, 152
 - and control transfer, 18, 136
 - and Default Interrupt Handlers, 147–148
 - and function requests, 132–133
 - for hardware interrupt handling, 142–145
 - and initialization, 11, 15–16, 97, 117, 132–133
 - in Initialization Table, 101
 - lifespan of, 15, 132
 - parameters for, 121
 - for Receive function, 390, 393–396
 - return codes in, 24, 129, 140
 - structure of, 122–124
 - and time–out handlers, 149–150
 - for Transmit function, 380, 382–386
 - use of, 16
- Reserved data pointers, initialization of, 113
- Reserved fields in Request Blocks, 16
- Reset Diskette System function, 190
- Reset/Initialize functions, 4
 - for Diskette Service, 190, 203–205
 - for Fixed Disk Service, 244–248
 - for Keyboard Service, 274–275
 - for Parallel Port Service, 425
 - for Pointing Device Service, 503–504
 - for Serial Communications Service, 370–372
- Resolution
 - and default fonts, 314
 - for pointing devices, 514–515
 - video, 309–310, 325
- Restore Video Environment, Video Service, 335–336
- Resume Stage after Interrupt return code, 25, 141, 618
- Resume Stage after Time Delay return code, 25, 141, 150, 618

R, cont'd

- Retryable Device Error return code, 25, 141
- Retryable Device Error With Time-out return code, 25, 141
- Retryable diskette errors, 195
- Retryable errors, 25, 140-141, 238
- Retryable fixed disk errors, 238
- Retryable Time-out Error return code, 25, 141
- Return Code Field Not Valid return code, 25, 141
- Return codes, 24-25, 129, 139-142, 617
 - action-required, 618
 - for Default Interrupt Handler, 148
 - for Diskette Service, 195
 - for Fixed Disk Service, 238
 - initialization of, 143
 - for Keyboard Service, 268
 - for Parallel Port Service, 418
 - for Real Time Clock Service, 445
 - for Request Blocks, 16, 123, 125
 - for Serial Communications Service, 363
 - for System Timer Service, 437
 - termination-indicating, 619-626
 - for Video Service, 315
- Return Line Status, Serial Communications Service, 409
- Return Logical ID Parameters functions, 4, 20, 152
 - for CMOS RAM Service, 533-534 and Default Interrupt Handlers, 147
 - for Direct Memory Access Service, 553
 - for Diskette Service, 197-198
 - for Error Log Service, 610-611
 - for Fixed Disk Service, 240
 - for Keyboard Security Service, 590-591
 - for Keyboard Service, 270
 - for Nonmaskable Interrupt Service, 488
 - for Parallel Port Service, 420-421
 - for Pointing Device Service, 499-500
 - for POS Service, 574-575
 - for Real Time Clock Service, 447-448
 - for Serial Communications Service, 365-366
 - for System Services, 471-472
 - for System Timer Service, 439
 - for Video Service, 317
- Return Modem Status Serial Communications Service, 410-411
- Return Print Status Parallel Port Service, 432-433
- Return ROM Fonts Information Video Service, 331-332
- RGB data format, 356
- RLL fixed disk controller, 35
- Rogue interrupts, 144
- ROM
 - ABIOS vs. CBIOS, 31
 - fonts resident in, 313-314
- ROM extensions, 27, 160
 - header for, 168
 - initialization of, 176-177
 - signature for, 168
- ROMCriticalSection and DOS compatibility box, 22-23, 154
- Rows, video, and fonts, 342

S

- Sample rate for pointing devices, 512-513
- Save Video Environment, Video Service, 333-334
- Scaling factor for pointing devices, 516-517

S, cont'd

- Scan codes for Keyboard Service, 263–267, 586–589
 - reading of, 280–281
- Scratch register, serial port, 73, 79
- Secondary device ID in Initialization Table, 100–101
- Sector number and size in diskette formatting, 214, 223
- Sectors per track, fixed disk, 236–237
- Security. *See* Keyboard, security services for
- Select Character Generator Block Video Service, 337–338
- Selector descriptors, 111
- Sequencer, video, 36, 307
- Serial buffer head pointer
 - receive, 389
 - transmit, 379, 382
 - port address for, 75
- Serial buffer tail pointer
 - receive, 389–390
 - transmit, 379–380, 382
- Serial Port Service, support by, 115
- Serial ports
 - addresses for, 40, 71–73, 77–79, 361
 - Device Block for, 367
 - device ID for, 4, 153
 - error handling for, 363
 - hardware environment for, 361–363
 - interrupt request for, 57
 - and keyboard controller, 261
 - return codes for, 620
 - services for, 359–360
 - Cancel, 407–408
 - Default Interrupt Handler, 364
 - Enable FIFO Control, 412–413
 - Modem Status, 404–406
 - Read Device Parameters, 367–369
 - Receive, 389–398
 - Reset/Initialize Serial Port, 370–372
 - Return Line Status, 409
 - Return Logical ID Parameters, 365–366
 - Return Modem Status, 410–411
 - Set Baud Rate, 378
 - Set Line Control, 375–377
 - Set Modem Control, 374
 - Transmit, 379–388
 - Transmit and Receive, 399–403
- Serial Receive Buffer, 71, 77, 389–391, 393–396
- Serial Receive Head Pointer, 389–391, 393–396
- Serial Receive Tail Pointer, 389–391, 393–396
- Serial-to-parallel conversions, 41, 362
- Service codes for extensions, 26, 159, 176
- Service-specific Entry Routines, 17
 - for control transfer, 134–135
- Service-Specific Interrupt Routine, 149
- Service-specific parameters, Request Block, 121, 123–124
- Service-Specific Unsuccessful Operation return code, 25, 141
- Services, relationship of, to other structures, 14
- Set Alarm Interrupt Real Time Clock Service, 454–456
- Set Baud Rate Serial Communications Service, 378

S, cont'd

- Set Device Parameters function, 4
 - for Diskette Service, 202–203
 - for Parallel Port Service, 423–424
 - for Real Time Clock Service, 452
- Set Keyboard LED Status service, 284–285
- Set Keyboard Scan Mode Code, Keyboard Service, 292–295
- Set Line Control, Serial Communications Service, 375–377
- Set Media Type for Format, Diskette Service, 223–225
- Set Modem Control, Serial Communications Service, 374
- Set Periodic Interrupt, Real Time Clock Service, 458–460
- Set Resolution, Pointing Device Service, 514–515
- Set Sample Rate, Pointing Device Service, 512–513
- Set Scaling Factor, Pointing Device Service, 516–518
- Set state for Diskette Change Line signal, 191
- Set Typematic Rate and Delay, Keyboard Service, 286–288
- Set Update–Ended Interrupt, Real Time Clock Service, 462–463
- Set Video Mode, Video Service, 324–330
- Sharing of hardware interrupts, 144–145
- Shift key, 266–267, 587
- Shutdown, CMOS RAM, 525
- Signatures for extensions, 168
- Single–staged functions, processing model of, 128
- Single–staged processing/single–tasking operating systems, 5–6
- Single–tasking operating system, 31
- Slashes, 267, 588
- Speaker, enabling of, 483–484
- Speeds, microprocessor, 31
- ST506 fixed disk adapter, 35, 235
- Stacks and stack frame
 - and ABIOSCall, 23, 155
 - and ABIOSCommonEntry, 20, 152
 - and extensions, 162
 - loading of, 137
 - pushing of pointers onto, 17, 134
 - for System Parameter Table, 98
- Staged functions, redirection of, 186–188
- Standard functions, 4
- Start Routines, 9, 131
 - flags for, 143
 - pointer for, 87–88, 98, 605
 - and return codes, 24
 - and staged function redirection, 186
- Status
 - of function calls, 25–26
 - modem, 404–406, 410
 - of parallel ports, 39, 71, 74–75, 417
 - printer, 432–433
- Status registers
 - diskette controller, 76
 - DMA, 61, 68
 - fixed disk adapter, 73
- Stop bits, serial support for, 40, 361
- Stored pointer method of frame loading, 137
- Structures. *See* Data structures
- Subaddress extension, POS, 70
- Successful return code, 25, 141
- Suspension of functions, 5, 22–23, 154–155

S, cont'd

- Switch to Protected Mode, System Services, 478–480
- Switch to Real Mode System Services, 475–477
- Sys Req key, 267, 589
- System board ROMs, initialization of, 108
- System board setup enable register, 67
- System configuration, reading of, 47–474
- System control port
 - definitions for, 5–59
 - port addresses or, 65–67
- System microprocessor, arbitration level for, 52
- System Parameters Table, building of, 13, 96, 98–99
- System scan codes, 280
 - for Keyboard Security Service, 586–589
 - for Keyboard Service, 264–267
- System Services, 469
 - device ID for, 4, 153
 - error handling for, 470
 - for Watchdog Timer, 44
 - services for, 471–484
 - Disable Address Line 20, 482
 - Enable Address Line 20, 481
 - Enable Speaker, 483–484
 - Read System Configuration, 473–474
 - Return Logical ID Parameters, 471–472
 - Switch to Protected Mode, 478–480
 - Switch to Real Mode, 475–477
- System Timer Service and devices, 42–46, 435
 - Default Interrupt Handler, 438
 - device ID for, 4, 153

- hardware environment for, 436–437
- Return Logical ID Parameters, 439

T

- Termination, return codes indicating, 619–626
- Text modes
 - and character blocks, 314
 - fonts for, 325, 339–344
 - resolution of, 310, 325
 - scan lines for, 327
- Time
 - reading of, 465–466
 - writing of, 466–467
 - See *also* System Timer Service and devices
- Time-out errors, 140
 - and keyboard, 268
 - return code for, 25, 141
- Time-out parameter in Request Block structure table, 123
- Time-out routines, 8–9, 149–150, 162
 - for function calling, 131
 - pointer for, 87–88, 98
 - and staged function redirection, 186
- Time-period handlers, 7–8, 130
- Time-related devices, 42
- Time to Wait Before Continuing Request field, 421, 423
- Time to Wait for Printer Initialization field, 421, 423
- Timer channels, 43–44
- Timer services. See System Timer Service and devices
- Timer tick, interrupt request for, 57
- Touchpads, 495
- Track number in diskette formatting, 214, 223

T, cont'd

Track switching, diskette, 204
Trackballs, 495
Transfer Control bytes field for DMA controller, 50, 547
Transfer conventions, 17-18, 94
 ABIOS 136
 for function calling, 131, 134-136
 and initialization, 117
 operating system, 18, 137-138
 and Request Block, 125
 with time-out handlers, 149
Transfer count registers, DMA, 61, 68
Transfer Data Pointer Mode for CMOS RAM, 536, 538
Transfer rates, diskette, 193
Transfer status, DMA, 546, 560
Transmit and Receive, Serial Communications Service, 399-403
Transmit Buffer, 379-382, 384-386, 399
Transmit Head Pointer, 379-382, 384-386, 391, 399
Transmit Holding Register, 71, 77, 383, 385
Transmit, Serial Communications Service, 379-388
Transmit Tail Pointer, 379-382, 384-386, 391, 399
Turn Diskette Motor Off, Diskette Service, 228-229
200 scan-line video, 310-311, 325
Typematic rate and delay, 286-288

U

Unexpected Reset return code, 25, 141, 618

Unit in Request Block structure table, 122
Unit-unique areas in Device Block, 92
Update-ended interrupts, 441
 canceling of, 464
 setting of, 462-463

V

Verify Diskette Data, Diskette Service, 191
Verify Diskette Sectors, Diskette Service, 218-220
Verify Fixed Disk Data, Fixed Disk Service, 255-256
VGA
 adapter for, 305
 chip for, 307-308
 and palette register, 312
 port addresses for, 74-76
 support for, 36

Video

device ID for, 4, 153
error handling for, 315
extension for, 184-185
hardware for, 36-37
hardware environment for, 307-308
modes for, 309-313, 324-330
monitor support for, 311-312
ROM-resident fonts for, 313-314
services for, 305-306
 Default Interrupt Handler, 316
 Enhanced Load Text Mode Font, 342-344
 Load Text Mode Font, 339-341
 Read Block of Color Registers, 354-355
 Read DAC Color Register, 349-350
 Read Device Parameters, 319-322
 Read Palette Register, 345-346

V, cont'd

Restore Video Environment, 335–336
Return Logical ID Parameters, 317
Return ROM Fonts Information, 331–332
Save Video Environment, 333–334
Select Character Generator Block, 337–338
Set Video Mode, 324–330
Write Block of DAC Color Registers, 356–358
Write DAC Color Register, 351–353
Write Palette Register, 347–348
Video Mode field, 326
Video subsystem register, VGA, 75
Virtual DMA channels, 53, 551

W

WAIT signal, math coprocessor, 32
Wait states, 5, 31
Watchdog Timer, 44
Work area in Request Block structure table, 124
Write and Verify, Fixed Disk Service, 253–254
Write Block of DAC Color Registers, Video Service, 356–358
Write Command(s) and Data to Keyboard, Keyboard Service, 300–303
Write Command(s) to Keyboard Controller, Keyboard Service, 295–299
Write DAC Color Register, Video Service, 351–353

Write Dynamic POS Data from an Adapter, 581–582
Write Filter Byte 1, Keyboard Security Service, 601–602
Write Filter Byte 2, Keyboard Security Service, 603–604
Write Invocation Byte, Keyboard Security Service, 597–598
Write mask register, DMA, 62, 69
Write Match Byte, Keyboard Security Service, 599–600
Write Palette Register, Video Service, 347–348
Write Password, Keyboard Security Service, 595–596
Write precompensation, fixed disk, 236–237
Write standard function, 4
Write Stored POS Data from CMOS RAM, POS Service, 577–578
Write Time and Date, Real Time Clock Service, 466–467
Write to CMOS RAM, CMOS RAM Service, 538
Write to Diskette, Diskette Service, 211–213
and Diskette Change Line signal, 191
Write to Error Log, Error Log Service, 614–615
Write to Fixed Disk, Fixed Disk Service, 251–252
Writing
to CMOS RAM, 45
to system control ports, 58–59

The Phoenix Technical Reference Series

Phoenix Technologies Ltd.

ABIOS for IBM® PS/2® Computers and Compatibles is an indispensable guide to the ABIOS, the part of the PS/2 BIOS that supports multitasking operating systems such as OS/2®. It is written for anyone needing a detailed understanding of the system software resident in their IBM PS/2 or compatible computer—applications programmers, operating system developers, hardware manufacturers, and students of PC architecture. This reference is the most complete source of information available on ROM-based system software for OS/2.

Key information includes:

- an ABIOS theory of operation section, that includes information on initialization, data structures and standard calling procedures
- a complete description of procedures for modifying ABIOS, with several examples
- descriptions of undocumented IBM BIOS features
- a description of how to access ABIOS services through OS/2
- programming tips for using ABIOS functions more productively
- descriptions of key hardware commands, including DMA, POS, and the Micro Channel.™

The information in this book is applicable to all Micro Channel Architecture-based IBM PS/2 and compatible computers.

Phoenix Technologies Ltd. of Norwood, Massachusetts is the industry leader in providing ROM BIOS products for IBM PC and PS/2 compatibles.

The Phoenix Technical Reference Series includes two other technical reference volumes on BIOS software—*CBIOS for IBM PS/2 Computers and Compatibles: The Complete Guide to ROM-Based System Software for DOS* and *System BIOS for IBM PC/XT/AT Computers and Compatibles: The Complete Guide to ROM-Based System Software*.

Applications		
OS/2	DOS	
ABIOS	CBIOS	System BIOS
PS/2 Hardware		PC/XT/AT Hardware



ISBN 0-201-51805-8

Addison-Wesley Publishing Company, Inc.