

Beyond BIOS:

Implementing the Unified Extensible Firmware Interface with Intel's Framework

Vincent Zimmer
Michael Rothman
Robert Hale

Intel
PRESS

Copyright © 2006 Intel Corporation. All rights reserved.

ISBN 0-9743649-0-8

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, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 750-4744. Requests to the Publisher for permission should be addressed to the Publisher, Intel Press, Intel Corporation, 2111 NE 25th Avenue, JF3-330, Hillsboro, OR 97124-5961. E-mail: intelpress@intel.com.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold with the understanding that the publisher is not engaged in professional services. If professional advice or other expert assistance is required, the services of a competent professional person should be sought.

Intel Corporation may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights that relate to the presented subject matter. The furnishing of documents and other materials and information does not provide any license, express or implied, by estoppel or otherwise, to any such patents, trademarks, copyrights, or other intellectual property rights.

Intel may make changes to specifications, product descriptions, and plans at any time, without notice.

Fictitious names of companies, products, people, characters, and/or data mentioned herein are not intended to represent any real individual, company, product, or event.

Intel products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

Intel, the Intel logo, Celeron, Intel Centrino, Intel NetBurst, Intel Xeon, Itanium, Pentium, MMX, and VTune are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

† Other names and brands may be claimed as the property of others.

This book is printed on acid-free paper. ∞

Publisher: Richard Bowles

Editor: David J. Clark

Program Management: Ashwood Group and Douglas Technology Group

Text Design & Composition: Horizon Interactive

Graphic Art: Kirsten Foote (illustrations), Ted Cyrek (cover)

Library of Congress Cataloging in Publication Data:

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

First printing

Contents

Preface xv

Chapter 1 Introduction 1

- History 2
 - Firmware as Hardware Abstraction 2
 - Option ROMs 4
 - Motherboard Hardware Initialization 4
 - BIOS as Differentiation 6
 - If One Abstraction Is Good... 6
 - Highest Cost Per Bit 7
 - How Does It Work At All? 8
 - Non-BIOS Alternatives 9
- EFI Goals 9
 - Operating System Neutrality 9
 - Crisply Defined and Extensible Interfaces 9
 - Modularity 10
 - Known Set of Intrinsic Services 10
 - Instruction Set Independent 10
 - High Level Language Friendly 10
 - Option ROMs as Full Partners 11
 - Scalability 11
 - Long-Lived Abstractions 11
 - Boot Manager 11

Rich “Pre-Boot” Environment, Limited Run-Time Environment	11
Framework Goals	12
Firmware for the Next 20 Years	12
Modularity Using an Object Model	13
A Framework, Not a Straight Jacket	13
Phasing	14
Support the Transition from Old to New	14
To OS or Not to OS	15
A Note on Ordering	16

Chapter 2 Basic EFI Architecture 17

Objects Managed by EFI-based Firmware	18
EFI System Table	18
Handle Database	19
Protocols	21
Working with Protocols	25
Multiple Protocol Instances	25
Tag GUID	25
EFI Images	26
Applications	30
OS Loader	31
Drivers	31
Events and Task Priority Levels	32

Chapter 3 EFI Driver Model 37

Why a Driver Model Prior to OS Booting?	38
Driver Initialization	38
Host Bus Controllers	40
Device Drivers	42
Bus Drivers	44
Platform Components	46
Hot Plug Events	47
Pseudo Code	49

Chapter 4 Protocols You Should Know 57

EFI OS Loaders	59
Device Path and Image Information of the OS Loader	62
Accessing Files in the Device Path of the OS Loader	63

Finding the OS Partition	64
Getting the Current System Configuration	65
Getting the Current Memory Map	66
Getting Environment Variables	67
Transitioning to an OS Kernel	68

Chapter 5 EFI Runtime 69

Isn't There Only One Kind of Memory?	71
How Are Runtime Services Exposed?	73
Time Services	75
Why Abstract Time?	76
Get Time	76
Set Time	77
Get Wakeup Time	77
Set Wakeup Time	78
Virtual Memory Services	78
Set Virtual Address Map	79
ConvertPointer	79
Variable Services	79
GetVariable	80
GetNextVariableName	81
SetVariable	81
Miscellaneous Services	84
Reset System	84
Get Next High Monotonic Count	85

Chapter 6 EFI Console Services 87

Simple Text Input Protocol	90
Simple Text Output Protocol	93
Remote Console Support	95
Console Splitter	99
Network Consoles	100

Chapter 7 Different Types of Platforms 103

Chapter 8 Volumes, Files, and Sections 121

Terminology	122
Design Constraints	124
The Boot Vector	124

- Alignment 124
- Easy Access 124
- Rare Writes 124
- Flash Headaches 125
- Bank Switching 126
- Fault Tolerance 127
- To File or Not to File 129
- File Types 130
- Section Types 130
 - Common Encapsulation Section Types 130
 - Common Leaf Section Types 131
 - Operations on Files 132
- How the Dispatcher Uses Volumes, Files, and Sections 132
- Internal Format 132
 - Basic Format 132
 - Managing the FV. 133
- Below the File System 134
 - The Driver Stack 134
 - Firmware Volume Block Interfaces 136
- Variables 136
- Capsules 137
 - The Structure of a Capsule 137
 - Passing through Reset 138
 - On the Other Side of Reset 139
 - The Capsule in DXE 140

Chapter 9 DXE Basics: Foundation, Dispatching, and Drivers 141

- DXE Foundation 143
 - Hand-Off Block (HOB) List 145
 - DXE Architectural Protocols 146
 - EFI System Table 149
 - EFI Boot Services Table 151
 - EFI Runtime Services Table 152
 - DXE Services Table 152
- Global Coherency Domain Services 153
 - GCD Memory Resources 153
 - GCD I/O Resources 155

- DXE Dispatcher 157
 - The *a priori* File 159
 - Dependency Grammar 160
- DXE Drivers 161
- Boot Device Selection (BDS) Phase 162
 - Console Devices 163
 - Boot Devices 164
 - Boot Services Terminate 164

Chapter 10 Some Common EFI Functions 167

- Architectural Protocol Examples 168
 - CPU Architectural Protocol 169
 - Real Time Clock Architectural Protocol 172
 - Timer Architectural Protocol 172
 - Reset Architectural Protocol 173
 - Boot Device Selection Architectural Protocol 174
 - Variable Architectural Protocol 175
 - Watchdog Timer Architectural Protocol 176
- PCI Protocols 177
 - PCI Host Bridge Resource Allocation Protocol 177
 - PCI Root Bridge I/O 183
 - PCI I/O 185
- Block I/O 188
- Disk I/O 190
- Simple File System 191
 - EFI File Protocol 193

Chapter 11 Information Passing 195

- Similarities and Differences 196
- Variables 198
 - Using Variables 198
 - Variable Volatility 198
 - Size Concerns 199
 - Common Variables 199
 - Summary 200
- Bridging the Phase Gap: Hand-Off Blocks (HOBs) – 200
 - Keeping PHIT 200
 - Using HOBs 201

- HOBs in DXE 201
- Summary 201
- The Human Interface Infrastructure (HII) 201
 - Terminology 202
 - The Configuration Model 203
 - Keyboards 204
 - Fonts 205
 - Strings 206
 - Forms 207
 - The HII Database Interface 211
 - Strings and Scripting 211
 - Driver Design Notes 211
 - The Data Hub 212
 - Message Ordering 212
 - Classes for Editing 214
 - Progress Codes 214

Chapter 12 Differences between DXE Drivers and EFI Drivers 215

- Global Coherency Domain Services 216
 - Dispatcher Services 223
 - Dependency Expression Reverse Polish Notation (RPN) 223
- DXE Dispatcher State Machine 224
 - Example Orderings 226

Chapter 13 Boot Drive Selection 229

- Firmware Boot Manager 232
 - Related Definitions 235
- Globally-Defined Variables 236
- Default Behavior for Boot Option Variables 239
- Boot Mechanisms 239
 - Boot via Simple File Protocol 239
 - Boot via LOAD_FILE Protocol 240

Chapter 14 Boot Flows 243

- Defined Boot Modes 244
- Priority of Boot Paths 245
- Reset Boot Paths 248

Intel® Itanium® Processor Reset	248
Non-Power-on Resets	248
Normal Boot Paths	249
Basic G0-to-S0 and S0 Variation Boot Paths	249
S-State Boot Paths	250
Recovery Paths	251
Discovery	251
General Recovery Architecture	252
Special Boot Path Topics	253
Special Boot Paths	253
Special Intel Itanium® Architecture Boot Paths	253
Intel Itanium® Architecture Access to the Boot Firmware Volume	254
Architectural Boot Mode PPIs	258
Recovery	258
Discovery	259

Chapter 15 Pre-EFI Initialization (PEI) 261

Scope	262
Rationale	262
Overview	263
Phase Prerequisites	267
Temporary RAM	267
Boot Firmware Volume	267
Security Primitives	267
Concepts	268
PEI Foundation	268
Pre-EFI Initialization Modules (PEIMs)	269
PEI Services	269
PEIM-to-PEIM Interfaces (PPIs)	270
Simple Heap	271
Hand-Off Blocks (HOBs)	271
Operation	272
Dependency Expressions	273
Verification/Authentication	274
PEIM Execution	274
Memory Discovery	274
Intel® Itanium® Processor MP Considerations	275

- Recovery 276
- S3 Resume 277
- Example System 277

Chapter 16 Putting It All Together—Firmware Emulation 281

- Virtual Platform 282
 - Emulation Firmware Phases 284
- Hardware Passthrough 290

Chapter 17 Compatibility Support Module 293

- CSM: A Bridge between Innovation and Tradition 294
- CSM Architecture 296
 - EfiCompatibility (CSM32) 297
 - Compatibility16BIOS (CSM16) 297
 - Compatibility16SMM 297
 - Thunk and ReverseThunk 297
 - Legacy Option ROM 298
- EfiCompatibility (CSM32) 298
 - Legacy Bios Driver 299
 - Legacy BIOS Protocol 300
 - Legacy BIOS Platform Protocol 301
 - Legacy Region Protocol 302
 - Legacy 8259 Protocol 302
 - Legacy Interrupt Protocol 303
- Compatibility16BIOS (CSM16) 303
 - Communication between CSM32 and CSM16 305
 - Compatibility16 Table 305
 - Compatibility16 Functions 308
 - Compatibility16 Data Structures 311
 - Compatibility16SMM 312
- Thunk and ReverseThunk 312
- Functional Visualization of CSM 314
 - Installing Legacy BIOS Environment 318
 - Booting to a Legacy Operating System 319

Appendix A Data Types	321
Appendix B Status Codes	325
Appendix C Quick Reference	329
Glossary	343
Index	347