Unified Extensible Firmware Interface (UEFI) Framework

**UEFI Overview**

Intel SSG/SSD/UEFI
Legal Disclaimer

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

This document contains information on products in the design phase of development.

All products, computer systems, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.

Intel, Intel Atom, Intel Core, and the Intel logo 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.

Copyright © 2011, Intel Corporation. All rights reserved.
Agenda

- BIOS Background
- UEFI Overview
- Platform Initialization (PI) Overview
What’s Legacy BIOS

- **Basic Input - Output System** for original IBM PC/XT and PC/AT
- Originated in 1980s
- Based on 8086 architecture
- A group of clearly defined OS-independent interface for hardware
  - Int10 for Video service
  - Int13 disk service
  - Int16 keyboard service
  - Int18 BIOS ROM loader
  - Int19 bootstrap loader
- Availability of MS-DOS outside of IBM allowed applications to run equally well across different brands of box "PC clones".
EFI / UEFI History Timeline

Specifications
- PCI Spec
- Intel® Itanium® Platforms
- EFI 1.02
- EFI Dev Kit (EDK)
- framework 0.9 Spec

Implementation
- IBM 16 Bit BIOS
- EFI only way to boot Itanium® Platforms
- EFI Sample Implementation 1.10.14.6x
- EFI Dev Kit (EDK)

Open Source EFI Developer Kit (EDK) [http://www.tianocore.sourceforge.net](http://www.tianocore.sourceforge.net)
UEFI Specifications - [http://www.uefi.org](http://www.uefi.org)
UEFI Specification Timeline

- **UEFI 2.0**
  - PI 1.0
  - SCT UEFI 2.0
  - EDK 1.01: UEFI 2.0
  - SCT PI 1.0

- **UEFI 2.1**
  - PI 1.1
  - SCT UEFI 2.1
  - EDK 1.04: UEFI 2.1 + PI 1.0

- **UEFI 2.2**
  - PI 1.2
  - Shell 2.0
  - Packaging 1.0
  - EDK 1.05: UEFI 2.1 + PI 1.0

- **UEFI 2.3**
  - PI 1.2
  - EDK II: UEFI 2.3 + PI 1.2+

- **UEFI 2.0+**
  - EDK II: UEFI 2.1+ + PI 1.0

- **UEFI 2.3+**
  - EDK II: UEFI 2.3+ + PI 1.2+

- **SCT UEFI 2.1**

- **SCT UEFI 2.0**

- **EDK 1.04**
  - UEFI 2.1 + PI 1.0

- **EDK 1.05**
  - UEFI 2.1+ + PI 1.0

- **EDK 1.01**
  - UEFI 2.0

- **EDK 1.04**
  - UEFI 2.1 + PI 1.0

- **EDK 1.05**
  - UEFI 2.1+ + PI 1.0

- **Open Source**

- **http://uefi.org**

- **BIOS background**

All products, dates, and programs are based on current expectations and subject to change without notice.
Agenda

- BIOS Background
- UEFI Overview
- Platform Initialization (PI) Overview
What is UEFI?

- **Unified (EFI) / Extensible Firmware Interface (EFI)**
- UEFI is an interface specification
- Abstracts BIOS from OS
  - Decouples development
- Compatible by design
  - Evolution, not revolution
- Modular and extensible
  - OS-Neutral value add
- Provide efficient Option ROM Replacement
  - Common source for multiple CPU architectures
- Complements existing interfaces
UEFI Overview
Unified EFI (UEFI) Forum – www.uefi.org

• Promoters
  – OEMs: Dell, HP, IBM, Lenovo
  – IBVs: AMI, Insyde, Phoenix
  – AMD, Apple, Intel, Microsoft

• UEFI Specification
  – EFI 1.10 specification contributed to the Forum by Intel and Microsoft to be used as a starting draft
  – UEFI 2.0 - 2.3 specification released.
  – Forum will evolve, extend, and add any new functionality as required
  – Intel contributed EFI 1.10 SCT being used as starting base for UEFI

Purpose: Worldwide adoption and promotion of UEFI specifications.
UEFI Membership

Adopters:
• Any entity wanting to implement the specification

Contributors:
• Corporations, groups or individuals wanting to participate in UEFI
• Chance to join work groups and contribute to spec or test development
• Early access to drafts and work in progress

Promoters:
• Board and Corporate Officers
Boot device support

- Hard disk
- Removable media
  - CD-ROM, DVD-ROM
    - El Torito 1.0 "No emulation"
  - Floppy, USB Storage, etc.
- Network
  - PXE BIOS support specification (Wire for Management)
  - iSCSI
- Future media via extensibility methods
New Partition Structure

LBA0  LBA1  LBA\text{n}

MBR  Partition  Table HDR  0  1  \ldots  n  Partition 1

First useable block  Start partition  End partition

Primary Partition Table

Backup Partition Table

Last useable block

See Section-5 UEFI 2.X Spec.
GPT Advantages over MBR Partition Table

- 64-bit Logical Block Addressing.
- Supports unlimited number of partitions.
- Uses a primary and backup table for redundancy.
- Uses version number and size fields for future expansion.
- Uses CRC32 fields for improved data integrity.
- Defines a GUID for uniquely identifying each partition.
- Uses a GUID and attributes to define partition content type.
- Each partition contains a 36 Unicode character human readable name.
- No magic code must execute as part of booting.
- Fixes the 2.2 Terabyte Problem.
UEFI Terminology

UEFI Specification - Key Concepts

- **Objects** - manage system state, including I/O devices, memory, and events
- **The UEFI System Table** - data structure with data in-formation tables to interface with the systems
- **Handle database and protocols** - callable interfaces that are registered
- **UEFI images** - the executable content format
- **Events** - the software can be signaled in response to some other activity
- **Device paths** - a data structure that describes the hardware location of an entity
UEFI Data Structures - UEFI System Table

- EFI System Table
  - EFI Runtime Services Table
    - Variable Services
    - Real Time Clock Services
    - Reset Services
    - Status Code Services
    - Virtual Memory Services
  - Version Information
    - EFI Specification Version
    - Firmware Vendor
    - Firmware Revision
  - Handle Database
  - Protocol Interface

UEFI Terminology

Boot Service Data Structures

Runtime Data Structures
GUID

• “Globally” Unique Identity
  – 128-bit quantity defined by Wired for Management WfM 2.0 specification **
• Used to identify protocols
  – 1:1 with interfaces
• Regulate extension mechanism
  – Documented in the spec
  – Added through drivers

Safe co-existence of 3rd party extensions

** http://www.intel.com/design/archives/wfm/index.htm
Legacy BIOS vs UEFI

Legacy BIOS
- INT 10h
- INT 13h
  - Chaining
- INT 16h
- INT 15h

? Ralf Brown’s Interrupt List

Legacy BIOS
- INT 10h
- INT 13h
  - Chaining
- INT 16h
- INT 15h

UEFI Means the Pieces all Fit and Work!

- UEFI
- GUID1 UEFI Specification
- GUID2 PI Specification
- GUID3 ODM defined
- GUID4 OEM defined
- GUID5 IBV defined
Handles

• All protocols have a handle which is associated with the protocol
• Every device and executable image in UEFI has a handle protocol in the handle database
• Every boot device must have a device path protocol to describe it
Protocols (API)

GUID, Interface Structure, Services

DEVICE_PATH, DEVICE_IO, BLOCK_IO, DISK_IO, FILE_SYSTEM,
SIMPLE_INPUT, SIMPLE_TEXT_OUTPUT, SERIAL_IO, PXE_BC,
SIMPLE_NETWORK, LOAD_FILE, UNICODE_COLLATION

BlkIo->ReadBlocks(BlkIo, ...)

Access Device or Services Produced by Other UEFI Drivers
Handle Protocol Database

UEFI Terminology
Device Path Protocol

• A data structure description of where a device is in the platform
• All boot devices, logical devices and images must be described by a device path
• 6 types of device paths:
  – Hardware
  – ACPI – UID/HID of device in AML
  – Messaging – i.e. LAN, Fiber Channel, ATAPI, SCSI, USB
  – Media – i.e. Hard Drive, Floppy or CD-ROM
  – EDD 3.0 boot device – see EDD 3.0 spec int13 48
  – End of hardware – marks end of device path

See Section-9 UEFI 2.X Spec.
Why UEFI Device Path? –
An UEFI Device Path describes a **boot target**.
Binary description of the physical location of a specific target.

Acpi(PNP0A03,0) /Pci(1F|1) /Ata(Primary,Master) /HD(Part3, Sig010…) \EFI\Boot”/”OSLoader.efi”

**UEFI Terminology**

- **Note:** Boot Sequence is part of the PI Spec.
Boot Services

- Events and notifications
  - Polled devices, no interrupts
- Watchdog timer
  - Elegant recovery
- Memory allocation
- Handle location – for finding protocols
- Image loading
  - Drivers, applications, OS loader

Runtime Services

Available at both Boot time and Runtime

- Timer, Wakeup alarm
  - Allows system to wake up or power on at a set time.
- Variables
  - Boot manager handshake
- System reset

UEFI Aware OS

RT Services are Minimal set to meet OSV needs
Typical System

CPU

PCI Host Bus

PCI Bus

PCI-PCMCIA

USB

IDE

IDE Bus

PCI-ISA Bridge

PCI Bus

VGA

ISA Bus

ISA FDC

ISD

CD-ROM

Hard Drive

Mouse

Keyboard

Floppy Drive

Bus Controller

Device Controller

Other

See Section-2.5 UEFI 2.X Spec.
Driver Initialization

- UEFI Driver Handoff State
- Not Allowed to Touch Hardware Resources
- Installs Driver Binding on Driver Image Handle

*Created by LoadImage()*

*Installed in Driver Initialization*
*Implemented by Driver Writer*

Registers Driver for Later Use
Driver Binding Protocol

Driver Image Handle

LOADED_IMAGE

DRIVER_BINDING

DRIVER_BINDING

Supported()
Start()
Stop()
Version
Example: UEFI ATAPI Driver Stack

- HD Handle
- File System Protocol (FAT)
- Disk IO Protocol
- Block IO Protocol
- UEFI ATAPI Driver
- IDE ATAPI disk drive

ATAPI Device Path
ACPI(pnp0604,0)/PCI(0,1)/ATA(primary, master)

Image Handle

Device Path Protocol

UEFI system partition

UEFI boot services
Agenda

- BIOS Background
- UEFI Overview
- Platform Initialization (PI) Overview
Technology not addressed by UEFI

- Memory Initialization
- Recovery
- FLASH update
- ACPI S3
- Platform Initialization
- System Management Mode (SMM)
- Setup

UEFI Separates BIOS and OS
UEFI and PI Specifications

USWG/PIWG Relationship

- UEFI Spec is about interfaces between OS, add-in driver and system firmware
  - A new model for the interface between the Operating systems and other high-level software and the platform firmware

- PI Specs relate to making UEFI implementations
  - Promote interoperability between firmware components providers
  - Modular components like silicon drivers (e.g. PCI) and value-add drivers (security)

UEFI and PI are Independent Interfaces
Intel® Platform Innovation Framework for UEFI and Platform Initialization (PI)

- Base Core Foundation ("Green H")
- Foundation lets different teams share code
- Developers can easily move between projects
- Chipset code enabled by Silicon vendor
- Standardization benefits the industry
- IBV provides value add
- Glue code "Big H" is Open Source on [www.tianocore.org](http://www.tianocore.org)
- The framework¹ Start point for Platform Initialization (PI) Specification on [www.UEFI.org](http://www.UEFI.org)

¹Intel® Platform Innovation Framework for UEFI
Intel® Platform Innovation Framework for UEFI and Platform Initialization (PI) Overview

Specification time line

- **2001**: framework 0.9 Spec
- **2002**: EFI 1.02, EFI 1.10
- **2003**: PI 1.0
- **2004**: UEFI 2.0
- **2005**: PI 1.1
- **2006**: UEFI 2.1
- **2007**: PI 1.2
- **2008**: UEFI 2.2
- **2009**: UEFI 2.3
- **2010**: Packaging 1.0, Shell 2.0

1Intel® Platform Innovation Framework for UEFI
The framework\(^1\) and PI Design Strategy

- High level design based on the framework\(^1\) plus modular components
- Generalize the framework\(^1\) Maximize reuse of infrastructure
  - High degree of independence from platform and market segment specifics
- Specifics encapsulated in the drivers
  - Drivers map to software visible hardware
  - Isolate hardware/platform specifics to support component-based firmware construction

\(^1\)Intel® Platform Innovation Framework for UEFI
Get to “C” Code Quickly

• Commercial “C” compilers use stack model
  – Requires some memory initialized for a stack
• Split the framework\(^1\) and PI infrastructure in two
  – Pre-EFI Initialization (PEI), preamble to get memory
  – Driver Execution Environment (DXE), infrastructure to support “C” coded EFI drivers
• First part of the framework\(^1\)/PI finds memory by using special stack
  – Infrastructure code plus PEI Modules
• The framework\(^1\)/PI uses modules for CPU, chipset and board
  – Minimum initialization to get memory working
• Architecture only requires “enough” memory
  – PEI limited so defer to rich DXE “C” environment
Architecture Execution Flow

Boot Execution Flow

Pre EFI Initialization (PEI)
- CPU Init
- Chipset Init
- Board Init

UEFI Interface
- Device, Bus, or Service Driver

EFI Driver Dispatcher
- Intrinsic Services

Boot Manager
- OS-Absent App
- Transient OS Environment
- Transient OS Boot Loader
- Final OS Boot Loader
- Final OS Environment

OS-Present App

Power on
[... Platform initialization ...]

[... OS boot ....]

Run Time (RT)

After Life (AL)

Shutdown
POST Execution Flow – High Level

1. **Reset**
   - **CPU Init**
   - **Memory Init**
   - **CS Init**

2. **POST Dispatch**
   - **Console Init**
   - **Device Init**
   - **Bus Init**

3. **Boot Dev Select**
   - **Legacy OS Load**
   - **EFI Pre-boot Application**

4. **Normal Boot**
   - **S3 Resume**
   - **Recovery**

5. **Boot Mode**

6. **OS Runtime**
Execution Flow – High Level

Boot Execution Flow

- **PEI**
  - Reset
  - CPU Init
  - Memory Init CS Init
  - Boot Mode
    - S3 Resume
    - Recovery

- **POST**
  - Firmware Volumes
  - Cache as RAM
  - PEIM
  - Handoff Blocks HOB
  - NVRAM
  - Capsules

- **EFI Pre-boot**
  - Application
  - Legacy OS Load
  - EFI Pre-boot Application

- **OS Runtime**
  - GUID
POST Execution Flow – High Level

- Reset
- CPU Init
- Memory Init
- CS Init

Boot Mode

- S3 Resume
- Recovery

POST Dispatch

Compatibility Support Module (CSM)

EFI Preboot

Normal Boot

 EFI Runtime Services

Legacy OS Load

EFI Pre-boot Application

TSL

OS Runtime