Beyond DOS: The UEFI Shell – a Modern Pre-boot Application Environment

Mike Rothman, Sr. Staff BIOS Software Engineer, Intel
Jaben Carsey, Staff BIOS Engineer, Intel
Nathan C Skalsky, Staff Firmware Engineer, IBM
Jeff Bobzin, UEFI Architect, Insyde

EFIS005
Agenda

• Shells – History and Standardization
• Applications and Scripts
• UEFI Shell 2.0 Unique Features
• IBM Shell Innovations
• Insyde Shell Innovations
Agenda

- **Shells – History and Standardization**
- Applications and Scripts
- UEFI Shell 2.0 Unique Features
- IBM Shell Innovations
- Insyde Shell Innovations
Shells – History and Standardization

- History of Command-lines
  - Early 1970’s – Unix arrives
  - Early 1980’s – DOS arrives
  - Today – Most O/Ses expose a command-line

- Command-line uses
  - Scripting
  - Program Launching
  - Abstraction to underlying system
  - Bring-up Target

Modern version of DOS

UEFI Shell
The presence of standards enables interoperability
Shells – History and Standardization

• Reusable code regardless of UEFI implementation.
  – Due to scripting and programmable methods being standardized.

Shell Interfaces (EFI_SHELL_PROTOCOL)

UEFI / PI Interfaces

Platform/Hardware Interfaces

Shell Applications

UEFI Driver

UEFI Driver

UEFI Driver

UEFI Driver

UEFI Driver

CPU Modules

Chipset Modules

Hardware
Agenda

- Shells – History and Standardization
- Applications and Scripts
- UEFI Shell 2.0 Unique Features
- IBM Shell Innovations
- Insyde Shell Innovations
A UEFI Shell 2.0 Architecture

Shell Library
Shell Applications
Shell Scripts

Command-line Parser
Shell Protocol
Shell Console Parser
Command Launcher

Script Processor
Level 3 Command Set
Profile Command Set

Console Driver(s)
UEFI Driver
UEFI Driver
UEFI Driver
UEFI Driver
UEFI Driver
UEFI Driver

UEFI / PI Interfaces

CPU Modules
Chipset Modules

Hardware
Shell Applications

• UEFI Shell 2.0 applications are compiled C code binaries that:
  – Use a Shell protocol
    ▪ **EfiShellProtocol** – provides APIs for file IO and Shell Environment IO
    ▪ **EfiShellParametersProtocol** – provides Std I/O and Argc/Argv
  – Optionally use UEFI protocols
  – Are launched from command line, script, or in startup parameters to the shell itself

**Shell Applications replace EFI Shell Extensions**
Shell Scripts

• Shell Scripts (.nsh files) provide automated execution of sequences of shell commands, shell or UEFI applications, and other shell scripts
• Support complex logic via For, If, and Goto
• Route human readable commands to correct hardware
What’s Changed?

• EFI Shell scripts remain compatible
• UEFI 2.0 Scripts have additional capabilities
  – Query for command availability
  – Consistent Command feature sets

• Old Shell Protocols deprecated
• UEFI Shell Protocols added
  – EFI Shell extensions require porting
  – UEFI applications will work
• New UDK Shell Lib supports both Protocols
Agenda

• Shells – History and Standardization
• Applications and Scripts
• **UEFI Shell 2.0 Unique Features**
• IBM Shell Innovations
• Insyde Shell Innovations
Key UEFI Shell 2.0 Features

New features provided by UEFI Shell are:
• Configure command sets available to end users configured at built time
• Provide backwards compatibility with existing shell scripts
• Manage firmware image size

[PcdsFixedAtBuild]
gEfiShellPkgTokenSpaceGuid.PcdShellSupportLevel | 3
## bit 0 = Drivers1, bit 1 = Debug1, bit 2 = Install1, bit 3 = Network1
gEfiShellPkgTokenSpaceGuid.PcdShellProfileMask   | 0xF
UEFI Shell Command Sets

• Shell Levels manage main features
  Level 0 – Launching a single application
  Level 1 – Adds scripting
  Level 2 – Adds file manipulation
  Level 3 – Adds UI and information retrieval

• Shell Profiles manage additional commands
  Install – Adds OS loader configuration
  Debug – Adds debug
  Driver – Adds driver manipulation
  Network – Adds network configuration & test

Unique ability to balance required features and commands against desired binary size
Shell.EFI Image Size Management

- **Maximum Image Size**
  - Level 3 shell with all 4 defined profiles
  - Supports all standard commands and a UI for interaction with an user

- **Minimum Image Size**
  - Level 0 shell with no profiles
  - Supports launching a single application

- **Additional extra profiles possible**

64 size combinations available!
Agenda

- Shells – History and Standardization
- Applications and Scripts
- UEFI Shell 2.0 Unique Features
- **IBM Shell Innovations**
- Insyde Shell Innovations
IBM Experience
“Smarter Firmware for a Smarter Planet”

Nathan C. Skalsky
Advisory Firmware Engineer, IBM
September 13th, 2010

Agenda
• IBM’s UEFI Shell 2.0 roadmap
• Key Features of UEFI Shell
• Running the UEFI Shell on IBM System x
• Example Uses of UEFI Shell
  • Bring Up
  • Development
  • System Manufacturing
  • Deployment/Provisioning
  • Maintenance
  • Debug
• Conclusion
UEFI Shell 2.0 Roadmap

- **Compatibility:** All UEFI-compliant System x Servers and Blades.

- **Integrated Shell:** A built-in level 3 UEFI Shell 2.0 is planned to be available via x86 IBM eX5 firmware updates within the next year.
  - Available as a Boot Item
  - Launch-able via UEFI Shell

- **Tools/CLI Strategy:** Current direction is to continue to use OS-based pre-boot deployments environments for flashing/in-band configuration updates. Shell is considered a supplementary command-line environment.
Key Features UEFI Shell 2.0

- “Common-Denominator” Preboot CLI
  - No dependencies on OS load / deployment
  - Direct-hardware environment
- Embeddable and flexible foot-print
- Scripting
  - Automating Configuration/deployment tasks
  - Automating Testing (reset tests, verify OS interfaces, run UEFI standards compliance tool (SCT))
- Shell Libraries enable ease of development and portability of applications
- Execute UEFI binaries
- Load/Unload Pre-Boot UEFI/DXE Drivers
Key Application Areas of UEFI Shell 2.0

• Early Hardware “Bring-up” Milestone
• Development and Testing
• Manufacturing
• Deployment/Provisioning
• Maintenance
• Debug / Product Support Investigations
Launching and Using UEFI Shell 2.0

- **One-Time**
  - F1 Setup → Boot Manager → Boot From File

- **As a Boot Option**
  - F1 Setup → Boot Manager → Add Boot Option
Using UEFI Shell during Development

- Scripting: Startup.nsh (think “Autoexec.bat”)
- Commands: DumpVariable/PCI/IbmHiiParse
Conclusion

- UEFI Shell 2.0 is a powerful “common denominator” environment and set of IO/console libraries
- IBM and IBM vendors heavily leverage the shell for bring-up, development, and debug activities
- Future is bright for automated and interactive configuration, deployment and update use-cases
Agenda

• Shells – History and Standardization
• Applications and Scripts
• UEFI Shell 2.0 Unique Features
• IBM Shell Innovations
• **Insyde Shell Innovations**
Using the Pre-Boot Application Environment

• Network Browsing

• Complex Testing
Riding on top of a Network Capable Shell...

- Extends pre-boot space onto Internet
- Network Browsing Examples:
  - IT department support page
    - Help pages
    - Http download client
    - Access to OS recovery images
  - Remote assist system
    - System drivers download from OEM service site
    - Remote system diagnostic
    - Hardware support page
Demo of Network Browsing
Network Browsing in UEFI...coming soon!

Networking sets applications free in the pre-boot space
Complex Testing in a shell application

- Test hardware features not supported in OS
- Accelerate hardware feature development
  - Simpler debug environment than OS
  - More control for probing error conditions
- Enable efficient testing of features
  - Rapid test cycles booting just to UEFI Shell
  - Specific error cases can be validated
RAS Feature development and testing...

- OSes have limited support
  - Processor offline
  - Memory offline
- Need “live” ACPI environment
  - Methods and events supported
  - ACPI Component Architecture (ACPIICA) is candidate
    - Designed for OS integration
    - Open source code base
    - Used in Linux and other Oses
    - Excellent APIs for OS abstraction

- Why port ACPIICA subsystem to UEFI shell?
  - UEFI is good fit for rapid testing and prototyping
  - UEFI protocols suitable to provide needed APIs
ACPI Internals

- Internal Modules of the ACPI Core Subsystem

Diagram:
- Resource Management
- Event Management
- Namespace Management
- ACPI Table Management
- ACPI Hardware Management
- AML Interpreter
Porting ACPICA

- Use standard C libs
- Use UEFI API to provide hardware access
- Use UEFI periodic timer events to monitor

```c
ACPI_STATUS AcpiOsReadPciConfiguration (  
ACPI_PCI_ID   *PciId,      UINT32  Register,  
void          *Value,      UINT32  Width){

UINT64        Pciex_Address=0;

Pciex_Address = CALC_EFI_PCIEX_ADDRESS (PciId->Bus,  
                                      PciId->Device,PciId->Function, Register);
switch (Width)   {
  case 8:    
    Status = gRootBridgeIo->Pci.Read (gRootBridgeIo,  
                              EfiPciWidthUint8, Pciex_Address, 1, Value);
...
```
Demo ACPICA running on Intel 4-socket platform.
Summary

- Shell 2.0 implementation fully compliant to UEFI Shell Specification now available on tianocore.org
- Configure your shell to meet feature set and image size sweet spot
- Network profile sets applications free in the pre-boot space
- UEFI Application environment is great test harness
Additional sources of information on this topic:

• Other Sessions – Next Slide
• Demos in the showcase – EFI Booth, #160
• More web based info:
  – **UDK 2010** -  
  – **UEFI Specifications** -  [http://www.uefi.org](http://www.uefi.org)
• Book on topic:
  – Beyond BIOS 2\textsuperscript{nd} edition - Intel Press
• Get the UEFI Shell 2.0 specification – www.uefi.org.
• Get the UDK ShellPkg with all the source code from www.tianocore.org
Beyond BIOS 2nd Edition Promotion

2nd Edition - *Beyond BIOS* available Q4 2010

To receive a complementary copy of the book
Register at
http://www.intel.com/intelpress/register.htm

Enter “Beyond BIOS Offer” plus the serial number on the back of this voucher in the Book Title field. Your book will be shipped to you.

Offer not valid for Intel employees. Limited time offer.

Vouchers available in session room and UEFI Tech showcase booth #160
Intel® UDK2010 Available on tianocore.org

Intel® UDK2010
Open Source
UEFI Development Kit

http://www.tianocore.Sourceforge.net
UEFI PLUGFEST in Taiwan Oct 12-15, 2010

Visit www.uefi.org/events for Event Info and Registration
# IDF 2010 UEFI Fall Sessions

**Sept. 13, 2010 Moscone Room 2006**

<table>
<thead>
<tr>
<th>EFI#</th>
<th>Company</th>
<th>Description</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>S001</td>
<td>Intel, IBM, HP</td>
<td>Introducing the New Intel® UEFI Development Kit: Industry Foundation for Platform Innovation</td>
<td>11:00 AM</td>
</tr>
<tr>
<td>S002</td>
<td>Intel, LSI, Dell, Phoenix</td>
<td>UEFI Advancements for Independent Hardware Vendors</td>
<td>1:05 PM</td>
</tr>
<tr>
<td>S003</td>
<td>Intel, WindRiver</td>
<td>Boot Loader Solutions for Intel® Atom™ Processor Based Embedded Devices</td>
<td>2:10 PM</td>
</tr>
<tr>
<td>S004</td>
<td>Intel, Dell, AMI</td>
<td>Zero-Touch Platform Manageability with UEFI</td>
<td>3:15 PM</td>
</tr>
<tr>
<td>S005</td>
<td>Intel, IBM, Insyde</td>
<td>Beyond DOS: The UEFI Shell – a Modern Pre-boot Application Environment</td>
<td>4:20 PM</td>
</tr>
<tr>
<td>Q001</td>
<td>All</td>
<td>UEFI Q &amp; A session with all Speakers</td>
<td>5:25 PM</td>
</tr>
</tbody>
</table>

DONE
Session Presentations - PDFs

The PDF for this Session presentation is available from our IDF Content Catalog at the end of the day at:

intel.com/go/idfsessions

URL is on top of Session Agenda Pages in Pocket Guide
Please Fill out the Session Evaluation Form

Give the completed form to the room monitors as you exit!

Thank You for your input, we use it to improve future Intel Developer Forum events
Q&A

Tweet your questions and comments to @intel_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.

• Intel may make changes to specifications and product descriptions at any time, without notice.
• All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.
• Intel, processors, chipsets, and desktop boards 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.
• Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.
• Intel, Intel Sponsors of Tomorrow. and Intel Sponsors of Tomorrow. Logo and the Intel logo are trademarks of Intel Corporation in the United States and other countries.
• *Other names and brands may be claimed as the property of others.
• Copyright ©2010 Intel Corporation.
Risk Factors

The above statements and any others in this document that refer to plans and expectations for the second quarter, the year and the future are forward-looking statements that involve a number of risks and uncertainties. Many factors could affect Intel's actual results, and variances from Intel's current expectations regarding such factors could cause actual results to differ materially from those expressed in these forward-looking statements. Intel presently considers the following to be the important factors that could cause actual results to differ materially from the corporation's expectations. Demand could be different from Intel's expectations due to factors including changes in business and economic conditions; customer acceptance of Intel’s and competitors’ products; changes in customer order patterns including order cancellations; and changes in the level of inventory at customers. Intel operates in intensely competitive industries that are characterized by a high percentage of costs that are fixed or difficult to reduce in the short term and product demand that is highly variable and difficult to forecast. Additionally, Intel is in the process of transitioning to its next generation of products on 32nm process technology, and there could be execution issues associated with these changes, including product defects and errata along with lower than anticipated manufacturing yields. Revenue and the gross margin percentage are affected by the timing of new Intel product introductions and the demand for and market acceptance of Intel's products; actions taken by Intel’s competitors, including product offerings and introductions, marketing programs and pricing pressures and Intel’s response to such actions; defects or disruptions in the supply of materials or resources; and Intel’s ability to respond quickly to technological developments and to incorporate new features into its products. The gross margin percentage could vary significantly from expectations based on changes in revenue levels; product mix and pricing; start-up costs, including costs associated with the new 32nm process technology; variations in inventory valuation, including variations related to the timing of qualifying products for sale; excess or obsolete inventory; manufacturing yields; changes in unit costs; impairments of long-lived assets, including manufacturing, assembly/test and intangible assets; the timing and execution of the manufacturing ramp and associated costs; and capacity utilization. Expenses, particularly certain marketing and compensation expenses, as well as restructuring and asset impairment charges, vary depending on the level of demand for Intel's products and the level of revenue and profits. The majority of our non-marketable equity investment portfolio balance is concentrated in the flash memory market segment, and declines in this market segment or changes in management’s plans with respect to our investment in this market segment could result in significant impairment charges, impacting restructuring charges as well as gains/losses on equity investments and interest and other. Intel's results could be impacted by adverse economic, social, political and physical/infrastructure conditions in countries where Intel, its customers or its suppliers operate, including military conflict and other security risks, natural disasters, infrastructure disruptions, health concerns and fluctuations in currency exchange rates. Intel’s results could be affected by the timing of closing of acquisitions and divestitures. Intel’s results could be affected by adverse effects associated with product defects and errata (deviations from published specifications), and by litigation or regulatory matters involving intellectual property, stockholder, consumer, antitrust and other issues, such as the litigation and regulatory matters described in Intel's SEC reports. An unfavorable ruling could include monetary damages or an injunction prohibiting us from manufacturing or selling one or more products, precluding particular business practices, impacting our ability to design our products, or requiring other remedies such as compulsory licensing of intellectual property. A detailed discussion of these and other factors that could affect Intel’s results is included in Intel’s SEC filings, including the report on Form 10-Q for the quarter ended March 27, 2010.

Rev. 5/7/10
Backup Slides
1) This item illustrates what the standard entry point for any UEFI compatible binary application or driver looks like. This is the fundamental starting point for all UEFI compatible programs which exposes the underlying UEFI firmware services.

2) During the initialization of a UEFI program, the standard entry point would be used to access the standard runtime and boot services that the UEFI compatible firmware provides.

3) In most shell-aware applications, there would be either a library or macro which would be used to provide access to the underlying shell protocol interfaces. This library/macro isn’t required by the UEFI shell specification, but would commonly be found in many of the available shell-aware programs.

4) In shell-aware applications, the availability of the functions defined in the EFI_SHELL_PROTOCOL can be leveraged.