

Bridging BIOS to UEFI



A White Paper By:
Dr. Gaurav Banga
SVP, Engineering & CTO, Phoenix Technologies

Copyright

© Copyright 2007 by Phoenix Technologies Ltd. All rights reserved. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the prior written permission of Phoenix Technologies Ltd.

Disclaimers

Phoenix Technologies Ltd. makes no representations or warranties with respect to the design and documentation herein described and especially disclaims any implied warranties of merchantability or fitness for a particular purpose. Further, Phoenix Technologies Ltd. reserves the right to revise this design and associated documentation and to make changes from time to time in the content without obligation of Phoenix Technologies Ltd. to notify any person of such revisions or changes.

Trademarks

Phoenix, Phoenix Technologies, Phoenix Technologies Logo are trademarks and/or registered trade-marks of Phoenix Technologies Ltd. All other marks and names are the property of their respective owners.

Contacting Phoenix

Corporate Address:

Phoenix Technologies, Ltd.

915 Murphy Ranch Road

Milpitas, California 95035

USA

Web site: <http://www.phoenix.com>

Document Date: July 15, 2007

Bridging BIOS to UEFI

New versions of the computer's operating system (OS) get all of the attention, but there is also an evolution occurring in the core systems firmware that is responsible for getting things started on a PC. Often referred to as the BIOS (basic input/output system), this firmware is becoming a modular and configurable system element based on the Unified Extensible Firmware Interface (UEFI) standard that will give computer system developers a simplified method for incorporating new hardware technology and customizing system functionality. By following a thoughtful migration from the existing BIOS to this "BIOS of the future," developers will be able to gain this enhanced flexibility and functionality in future designs without compromising designs already in development.

In the earliest personal computers (PCs), the BIOS served primarily to provide the operating system with a standard view of the system's core hardware, including keyboard, mass storage, display, and serial I/O. The BIOS firmware told the central processor (CPU) how to read and control these resources and provided the operating system with a standard interface that hid variations in detail among various chipsets and peripheral devices. This gave PC developers freedom of choice in hardware options as well as an ability to add innovative features while assuring compatibility with the operating system's standard initialization process.

Over time the BIOS developed additional capabilities. Additions included power-on self-test (POST), power management, automatic enumeration and resource allocation for add-in peripherals, and a variety of system control and configuration functions. These additions came as a result of innovation by BIOS vendors and provided additional design flexibility and control options to system developers.

Opening the BIOS

A desire began growing within the industry, however, to open BIOS innovation to others in the PC industry – especially silicon vendors. Opening the innovation process would provide greater opportunities for innovation within the PC industry by allowing more minds to work on the problem. It also would make it easier for system developers to differentiate their products by incorporating custom features in the BIOS.

Silicon vendors also desired a faster method for incorporating new system hardware options into the BIOS. The traditional approach was for silicon vendors to hand over to BIOS vendors the firmware used to test new silicon, then wait for those drivers to be worked into the BIOS before offering the silicon to system developers. To speed this process, silicon vendors wanted a standard interface on which to base their drivers. This standard interface would allow the drivers to "plug in" to the BIOS without additional software design effort, eliminating the delays associated with integrating them into the BIOS and speeding the evolution of PC hardware.

The silicon vendor "wish list" for the BIOS continued to grow. There was a desire to be able to use a high-level language such as C for driver development by standardizing on interfaces and coding practices, thereby easing driver development. The use of C would also help address a growing shortage of assembly language programmers. Another desire was to provide support of "pre-boot" environments that would give access to protected-mode memory and allow address space management for software to run without requiring the operating system to be loaded. This feature would be especially useful in embedded computing applications that may not need full PC functionality and in automating the manufacturing test of PC hardware by enabling services for built-in self test.

One of the first attempts to create such expanded system firmware, now evolving far beyond a

BIOS, came from microprocessor vendor Intel. The company developed the "Platform Innovation Framework," usually referred to simply as the Framework, as an architecture for system firmware development. Within that architecture Intel defined an Extensible Firmware Interface (EFI) as a standard for creating hardware drivers to work within the Framework. Working with Phoenix Technologies and other BIOS vendors, Intel incorporated EFI into the BIOS for its Itanium processor. The company further created a representative implementation it called Tiano to serve as a template for development and made the source code publicly available in order to promote the entire approach, which it referred to as "Green-H."

Intel's Green-H initiative met with objections from some quarters in the PC industry, however, because of several limitations. One was that the Framework and Intel's EFI drivers were specific to Intel hardware architectures; the industry wanted a compatibility path for adoption of alternative hardware from other silicon vendors. The approach also did not address newly emerging technologies such as multi-processor configurations, and was slow to evolve, limiting opportunities for innovation. Microsoft, as the primary OS vendor in the PC space, also wanted to have a say in how the firmware-to-OS interface looked like. Perhaps most importantly, the Intel specification could not be advanced without Intel's approval and Intel revised the EFI specification only once, three years after its 1999 introduction. Intel currently has no plans to introduce any new versions of EFI.

UEFI Arises

In response to these limitations and restrictions, the PC industry formed in 2005 the Unified EFI Forum to develop broader firmware standards based on Intel's EFI 1.10 specification, which it has licensed from Intel. In addition to broadening the hardware applicability of the concept behind Green-H, the Forum aimed to maintain the pace of innovation. UEFI 2.0 appeared in 2006 and UEFI 2.1 was introduced in January, 2007.

Firmware developed under the UEFI standard is a far cry from the traditional PC BIOS. Traditional BIOS is specific to the x86 architecture and needs to run under the 16-bit "real mode" of that architecture. It provides power-up hardware initialization and POST functions then serves as the interface between the hardware and the operating system. UEFI firmware serves some of the same functions, but is not a replacement for existing standards such as ACPI, SMBIOS, or the PCI firmware specifications. Instead, it enhances them by allowing simple incorporation of additional functionality.

UEFI-based firmware provides both boot services and run-time services. The boot services create a driver model for the devices used during boot, such as hard and floppy disk drives, network controllers, keyboards, mice and displays. The run-time services serve as the interface between the operating system and the hardware for low-level firmware functionality required during normal system operation. Rather than being hard coded and hand packed like the BIOS, however, UEFI-based firmware follows a modular approach that allows system developers several approaches for augmenting its functionality.

At the hardware level, the UEFI specification gives chip developers a standard interface so that they can create a firmware driver plug-in to handle their specific boot hardware. System developers can then take UEFI-based firmware and add the drivers for the hardware they selected without needing to do any additional program development. At the software level, the specification gives system developers a method for augmenting the firmware with their own code. Because the UEFI specification ensures that a simple, linear, protected memory addressing mode becomes available early in the boot process, developers have access to a well-defined pre-OS environment that they can use to run custom code for such things as manufacturing tests, system diagnostics, or provisioning functions.

Developers can also use the pre-OS environment to configure the system for embedded operation without requiring that it include normally essential PC features such as a keyboard, mouse, or monitor.

UEFI Advantages

As a result of the ease with which the UEFI-based firmware can be adapted, adopting the standard gives developers a number of advantages. One is to simplify the incorporation of new hardware technology as it becomes available. The only software impact of adding new hardware is adding a new driver to the firmware. Customization also becomes simpler. Vendors can create UEFI drivers to provide features and functions unique to their system and simply add them to the firmware.

This ease of customization is especially beneficial for systems that are to serve a low-volume market, such as a handheld computer for a shipping company that includes proprietary functionality and does not run a traditional OS. Creating such a system while using a traditional BIOS would require considerable software development. Under a UEFI system the development costs are considerably lower.

The adaptability of UEFI-based firmware also gives system developers more freedom of choice. They can create their customizations once then apply them to the UEFI firmware of whatever board design they choose. This simplifies switching between independent board vendors by virtually eliminating the firmware impact.

While the advantages of UEFI-based firmware are compelling, developers face a significant challenge to adopting the approach: time. Product development time for PCs average nearly 18 months, and server-class system often take longer. As a result, the development of many systems began well before the UEFI Forum released its specification to replace EFI. The development time for new silicon is also long, so the devices currently on the market were created before the UEFI specification became available. As a result, very few UEFI drivers exist for today's silicon. Similarly, mainstream OSes also lack support for UEFI firmware.

Building Bridges

As a result, the systems currently in development as well as those now starting will still require traditional BIOS support. Further, the need for a traditional BIOS will continue for a number of years while the industry is transitioning between the standards. Until all the necessary chips have UEFI driver support, operating systems are able to tap into UEFI run-time resources, and the market has discontinued the use of older, unsupported devices, developers will be caught in a "mix-and-match" situation where their systems will have a blend of supported and non-supported components. To meet their needs during the transition time, system developers will need firmware that bridges between the traditional BIOS and UEFI-based operation.

Phoenix Technology offers such a bridging solution: the SecureCore BIOS. This firmware is in full compliance with the UEFI 2.0 standard and is compatible with Intel's Tiano/Green H implementation and proprietary EFI drivers. It works with Intel's and AMD's upcoming chipset platforms, Windows Vista, XP, and earlier OSes, and classic silicon drivers. SecureCore guarantees support for today's silicon, eliminating the "mix-and-match" challenge. It also guarantees support for current OSes as well as forthcoming UEFI-aware OSes. Phoenix has already demonstrated SecureCore's compatibility with Microsoft's efforts at a recent Plugfest.

To help ease the shift that system developers must make between the traditional BIOS and UEFI-based firmware, SecureCore supports their existing assembly-language code-base and will accept many of their current firmware customizations. Developers can work with SecureCore using mixed-mode tools such as Visual Studio 2005 that understand both assembly and C code, allowing them to merge their existing firmware with SecureCore's code without re-coding. This gives developers time to move from a traditional to a UEFI-based approach gradually, in phases. Phoenix can provide substantial support to development teams during this transition.

The industry's move to UEFI is now becoming inevitable. Intel is targeting incorporation of UEFI compatible drivers into its reference platforms for the next-generation hardware chipsets and has discontinued major development on its proprietary EFI standards. AMD is also embracing UEFI 2.x, offering for its new CPU and chipsets UEFI drivers that Phoenix already supports. Microsoft has made Vista UEFI-aware and will support the standard in its forthcoming Longhorn server software. Major independent hardware vendors for plug-in I/O hardware have also adopted UEFI for driver development. The UEFI standard is thus being taken up by the major PC vendors, and others will have to follow in order to remain competitive.

With offerings like SecureCore, Phoenix Technology is helping developers prepare for full adoption of the UEFI standard in future generations of PCs. As a world leader in BIOS software development, Phoenix has in-depth understanding of the features and functions that will need to remain available during the transition. At the same time, its leadership position within the UEFI committees ensures that it has advanced understanding of where the industry is headed. With this combined background it has been able to develop SecureCore to be just what the industry needs just when it needs it.

Looking Forward

As the adoption of UEFI ramps up, it is important to keep in mind that the mere adoption of this new set of interfaces in the firmware layer is not a useful goal by itself. The PC industry must focus on how the adoption of these interfaces will speed innovation at lower costs.

Phoenix is in the process of utilizing the benefits of the new modular and more easily extensible codebase to rapidly implement support for a faster, more secure BIOS, the latest industry hardware standards, virtualization, a rich set of pre-OS applications, and tools that decrease BIOS enablement time and cost for new customer platforms by speeding up the debugging and validation processes. Beyond UEFI today, Phoenix is also looking at the use of advanced programming techniques, such as object oriented programming to further leverage the benefits of modern compiler and programming language technologies in firmware development.