1-512-256-0197
www.mindshare.com
training@mindshare.com

Hands-On UEFI Architecture

Training

Let MindShare Bring “Hands-On UEFI Architecture” to Life for You

Microsoft's announcement that all systems shipping with a desktop version of Windows 8 have to use UEFI (Unified Extensible Firmware Interface) and Secure Boot is speeding up the industry's transition from legacy BIOS to firmware producing the UEFI.

This change will impact you, whether you run Windows or not: Depending on the security policies implemented by the OEM, you may have difficulty installing the OS of your choice, even if it is UEFI based.

Although most new x86-based systems implement UEFI firmware, legacy BIOS is far from dead. Legacy BIOS is still used in embedded systems and will have to be supported on older computers.

This course will give you a thorough understanding of how both legacy BIOS and UEFI firmware take control of the system and prepare it to hand control to an OS boot loader, starting from the reset vector. On the first day of training we will give you a quick overview of not only legacy BIOS and UEFI, but also other boot firmware options that are available. The other two days are dedicated to walking you through the UEFI Forum's PI (Platform Initialization) spec's seven phases and explain how modules are discovered and dependencies are resolved as system hardware is initialized. Secure Boot, the UEFI compliant interface, CSM - the legacy BIOS compatibility module, the pre-OS execution environment, and the hand-off to an operating system boot loader are all covered.

You Will Learn:

- Components of Legacy BIOS
- The order and relevance of UEFI PI/Tiano's phases
- When and how key subsystems and devices are detected, initialized, and configured
- How backward compatibility can be implemented, making it possible to boot legacy, MBR-based, operating systems
- Where device configuration and management information is collected, stored, and communicated between the phases and to an operating system, including ACPI info
- Process of handing over system control from Firmware to OS
- The firmware's ongoing participation in running the system

Course Length: 4 Days

Who Should Attend?

Firmware/Driver/Software engineers and managers looking to understand the computer boot process from power-on until OS load.

Course Contents:

Firmware fundamentals

- What, why, and where is “firmware?”
- BIOS – The x86 legacy firmware architecture
- Limitations Legacy BIOS places on modern systems
- UEFI Forum’s *Platform Initialization*, Intel’s *Framework*, and UEFI (*Unified Extensible Firmware Interface*)
- Compatibility with Legacy BIOS
- x86 platform essentials
  - CPU capabilities in real- and protected-mode
  - Real- and protected mode memory maps

The POST Process

- What happens before the POST (Power-On Self Test)?
- Accessing the POST code
- Reading configuration settings from NV-RAM
- Chipset initialization
- How video is initialized
- How memory is tested
- "Walking" the PCI buses to identify and initialize installed PCI devices and functions
- Initialization of key legacy functions and devices
- USB initialization
- Memory map after POST
  - The IVT (Interrupt Vector Table)
  - BDA and EBDA content and layout
- Classic hand-off to operating system
- ACPI system initialization – The RSDP and RSDT

The UEFI PI and *Tiano*’s phases

- SEC – *Security* phase
  - Available system resources
  - Authenticating firmware boot code
  - Hand-off information
- PEI – *Pre-EFI Initialization* phase
  - Key PEI tasks
  - PEI Foundation code
  - PEIM – *PEI Modules*
    - Detection
    - Authentication
    - Dependency resolution
    - Execution
- PPI – The *PEIM-to-PEIM Interface*
- Creating HOBs – *Hand-Off Blocks*
- DXE – *Driver Execution Environment* phase
  - Key DXE tasks
  - DXE Foundation code
  - DXE Dispatcher
    - The *A Priori* file
    - Driver authentication
  - DXE and EFI drivers
    - DXE driver model
    - EFI driver model
  - Creation of the *EFI System Table* and other descriptor tables

- BDS – *Boot Device Selection* phase
    - BDS and UEFI
    - Console services
    - Running EFI applications
    - Boot paths
    - Selecting a boot option
        - Custom system configuration for selected operating system
        - Operating system hand-off
- TSL – *Transient System Load* phase
    - Transient OS boot loader
    - Transient OS environment
    - BDS re-entry
    - Terminating boot services
- RT – *Runtime* phase
    - Firmware services available to the operating system
    - Invoking DXE/EFI services
    - SMM – *System Management Mode*
    - *Fallback* mode
- AL – *Afterlife* phase
    - Purpose of the AL phase
    - Entering and exiting the AL phase
        - ACPI state transitions
        - Reset
        - OS panic or OS hang
    - Firmware update capsules

Secure Boot

- Exactly what is *Secure Boot*?
- How *Secure Boot* works
- OS vendors and *Secure Boot*
- Industry concerns

ACPI Implementation in *Tiano*

- ACPI Support driver
- ACPI Platform driver
- ACPI table storage

CSM – The BIOS Compatibility Support Module

- Creating BIOS-equivalent data structures and tables
- Required remnants of BIOS functionality
- Detecting and executing legacy option ROMs
- Implementation of legacy INT19 boot loader
- Hand-off to MBR based operating system loader
- Limitations of the *Compatibility Support Module* – CSM
- The *EfiCompatibility* and *Compatibility16* components

Firmware structure

- Firmware devices
- Firmware volumes
- FFS – the *Firmware File System*
- Different firmware file types
- Firmware file components
    - Header
    - Different sections

EFI disk organization

- GUID partition advantages
- MBR preservation
- GPT (GUID Partition Table) disk architecture
- The *System* partition

1-800-633-1440
www.mindshare.com
training@mindshare.com

Recommended Prerequisites:

Attendees are expected to have a technical background. Basic knowledge of assembly language programming, microprocessor technology, memory, and standard peripherals architecture is expected.

Course Material:

1) Presentation PDF handout
2) Lab exercises
3) Hands-on virtual environment