Hardware Prototyping Using a Windows-Hosted UEFI environment

UEFI Summer Plugfest – July 6-9, 2011
Presented by Tim Lewis (Phoenix Technologies Ltd.)
Agenda

• Introduction
• SMBus and UEFI Overview
• Adding ASF, DASH & IPMI
• Debugging With Windows-Hosted Emulation
• Q & A
Introduction

• Problem: UEFI BIOS is a tough development environment.
  – Developed before hardware available. Often have to code blind.
  – Tough to debug. How to log output, set breakpoints.
  – Time consuming to debug. Long turn-around time to update.
  – How to unit test corner cases? Touching hardware crashes the system.
Introduction (2)

• Solution: Develop UEFI BIOS in an emulated hardware environment.
  – *WinHost* (Phoenix) or *Nt32Pkg* (TianoCore) offer an emulated UEFI environment running under Windows operating systems.
  – Provides UEFI abstractions for many Windows hardware resources (disk, serial, network)
  – Quick-turn around and excellent debug support using Visual Studio.

• Real-Life Solution: How Phoenix used WinHost to develop ASF, DASH and IPMI using an emulated SMBus Host Controller.
What Is SMBus?

- SMBus is a low-power 2-wire multi-master bus built on top of I²C.
  - Well-defined transport.
  - Addressable.
  - UDID gives vendor/device id similar to PCI.
- Used as the backbone of numerous platform management standards:
  - ASF, DASH, IPMI
How Does UEFI Support SMBus?

- The UEFI PI Specification’s *SMBus Host Controller* protocol abstracts differences in hardware interface.

```c
typedef struct _EFI_SMBUS_HC_PROTOCOL {
  EFI_SMBUS_HC_EXECUTE_OPERATION Execute;
  EFI_SMBUS_HC_PROTOCOL_ARP_DEVICE ArpDevice;
  EFI_SMBUS_HC_PROTOCOL_GET_ARP_MAP GetArpMap;
  EFI_SMBUS_HC_PROTOCOL_NOTIFY Notify;
} EFI_SMBUS_HC_PROTOCOL;
```
What Was Missing?

• The UEFI Driver Model!
What Was Missing?

• The UEFI Driver Model!
What Was Missing?

• The UEFI Driver Model!
What Was Missing?

• The UEFI Driver Model!

UEFI Plugfest – July 2011
What Was Missing?

• The UEFI Driver Model!
How To Integrate SMBus Host Controller Into UEFI Driver Model?

- Added Driver Binding Protocol.
- Added Device Path Protocol.
- Added Means To Connect Non-Boot Devices.
  - By default, Boot Manager only connected console input, output and boot devices.
Adding ASF, DASH and IPMI

• With SMBus in UEFI driver model, standards like ASF, DASH and IPMI fit in naturally.
Emulate Host Controller

• By emulating one driver, we could test and debug our ASF, DASH and IPMI stack w/o hardware.

• Easy to test setup pages, sensors, boot options, SMBIOS push and error conditions.
SMBus Host Controller Driver Architecture

- SMBusHost.c
- DriverBinding.c

Active Device List & Config
- Config.c

40+ Individual Devices
- SmbusDevices.c
- AsfDevices.c
- DashDevices.c
- IpmiDevices.c
- AsfAsd.c
- DashMc.c

Libraries / Utility Routines
- Smbuslo.c
- Asflo.c
- Dashlo.c
- Ipmlo.c
Common Unit Test Debug Scenarios

• Device Enumeration for Devices That:
  — Are not enumerable (ARP)
  — Don’t identify themselves well.

• Error Check For Devices That:
  — Return ill-formed packets.
  — Don’t support specific commands.

• Return Different SMBIOS Tables, Boot Options and Other Items.
Debug Using Messages or Breakpoints!

- Dump out enumeration information.
Debug Using Messages or Breakpoints!

- View how commands are translated from DASH to SMBus.
Debug Using Messages or Breakpoints!

- View how commands are translated from DASH to SMBus.
Conclusion

• Device Emulation Speeds Development.
• Better debug & log capabilities.
• Better unit test case.
  – Once a unique device found, model it in software for regression tests.
Q & A
Thanks for attending the UEFI Summer Plugfest 2011

For more information on the Unified EFI Forum and UEFI Specifications, visit http://www.uefi.org

presented by
But wait, there’s more ...

**Wed (July 6)**
- UEFI State of the Union (10:30am, Intel)
- Implementing a Secure Boot Path with UEFI 2.3.1 (1:00pm, Insyde)
- UEFI SCT Overview (2:30pm, HP/Intel)

**Thu (July 7)**
- Replacing VGA: GOP Implementation in UEFI (10:30am, AMD)
- UEFI prototyping using a Windows-hosted UEFI environment (1:00pm, Phoenix)
- EFI Shell Lab (2:00-4:00pm, “Thunder”, Intel)
- GOP Enabling & Testing Lab (4:30—5:30pm, “Thunder”, Intel)

**Fri (July 8)**
- Best Practices for UEFI Option ROM Developers (10:30am, AMI)

Download presentations after the plugfest at www.uefi.org
Backup
extern EFI_GUID gSctSmbusIoProtocolGuid;

typedef struct _SCT_SMBUS_IO_PROTOCOL {
    UINT32 Size;
    EFI_SMBUS_ADDRESS Addr;
    SCT_SMBUS_IO_IDENTIFY Identify;
    SCT_SMBUS_IO_EXECUTE Execute;
} SCT_SMBUS_IO_PROTOCOL, *PSCT_SMBUS_IO_PROTOCOL;
extern EFI_GUID gSctAsfIoProtocolGuid;

typedef struct _SCT_ASF_IO_PROTOCOL {
    UINT32 Size;
    SCT_ASF_IO_SEND_MSG SendMessage;
} SCT_ASF_IO_PROTOCOL, *PSCT_ASF_IO_PROTOCOL;
extern EFI_GUID gSctMctpIoProtocolGuid;

typedef struct _SCT_MCTP_IO_PROTOCOL {
    UINT32 Size;
    MCTP_EID Eid;
    MCTP_MEDIUM Medium;
    SCT_MCTP_SEND_MSG SendMessage;
} SCT_MCTP_IO_PROTOCOL, *PSCT_MCTP_IO_PROTOCOL;
extern EFI_GUID gSctIpmiProtocolGuid;

typedef struct _SCT_IPMI_IO_PROTOCOL {
    UINT32 Size;
    SCT_IPMI_SEND_REQUEST SendRequest;
} SCT_IPMI_IO_PROTOCOL, *PSCT_IPMI_IO_PROTOCOL;
References

• **System Management Bus (SMBus) Specification**, Version 2.0, August 3, 2000

• **UEFI Specification**, Version 2.3.1, April 6, 2011