



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.

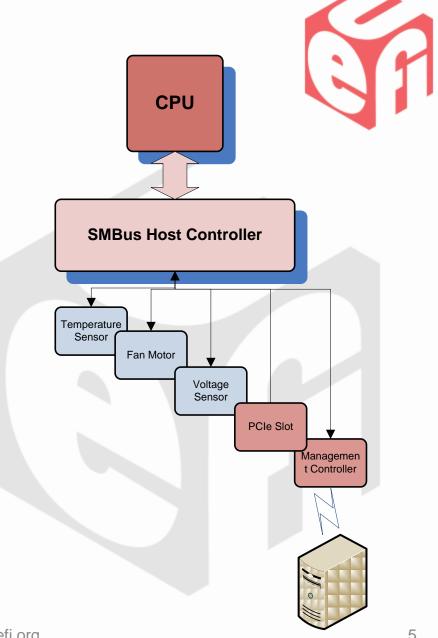
Introduction (2)



- 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 2wire 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?

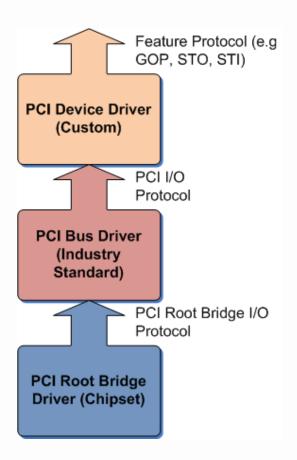


 The UEFI PI Specification's SMBus Host Controller protocol abstracts differences in hardware interface.

THE STATE OF THE S

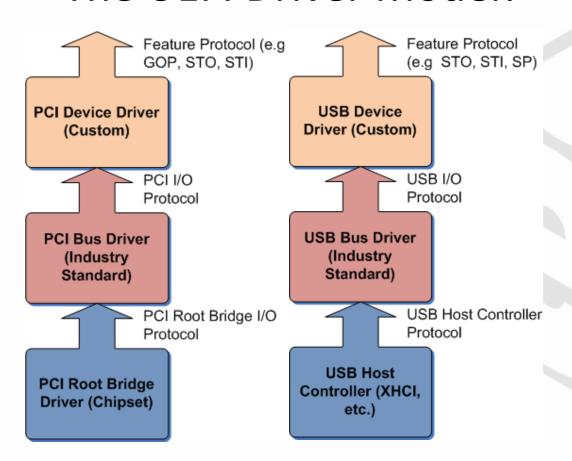




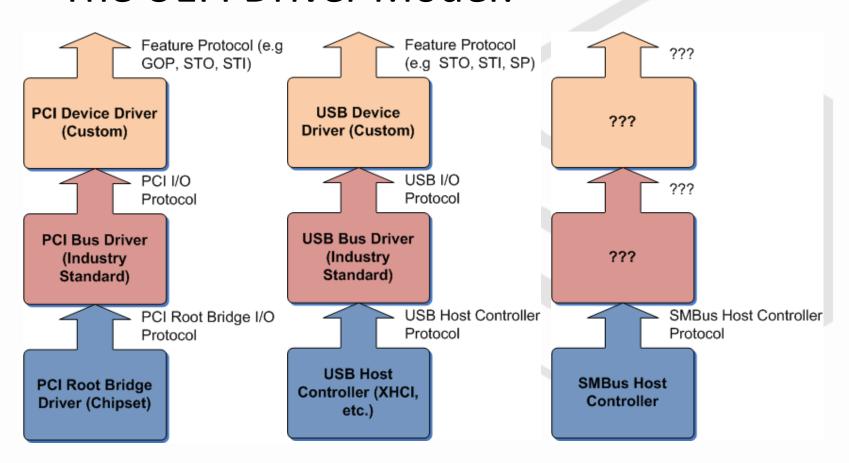




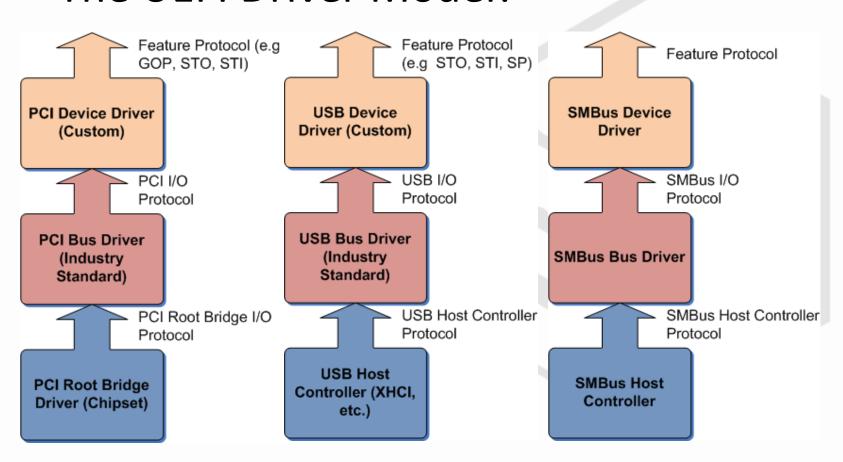










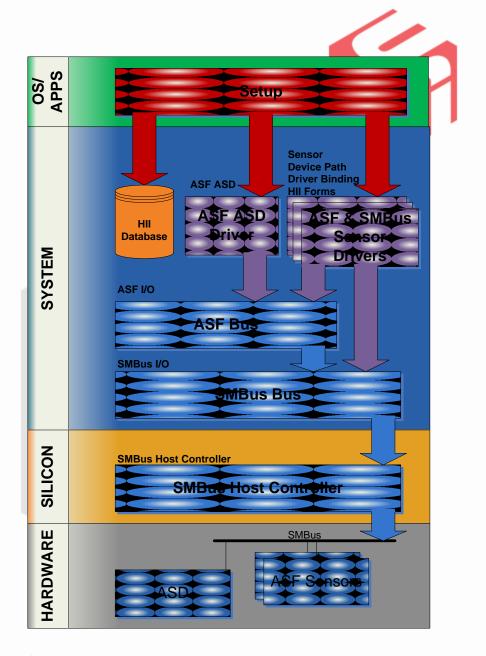


How To Integrate SMBus Host Controller Into UEFI Driver Model?

- 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.

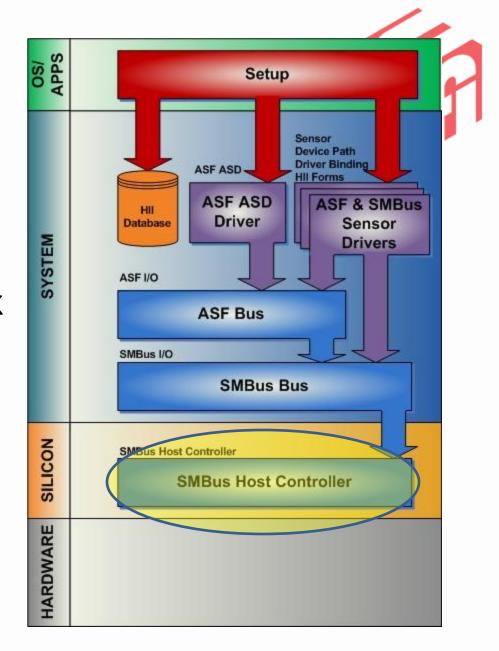
Adding ASF, DASH and IPMI

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

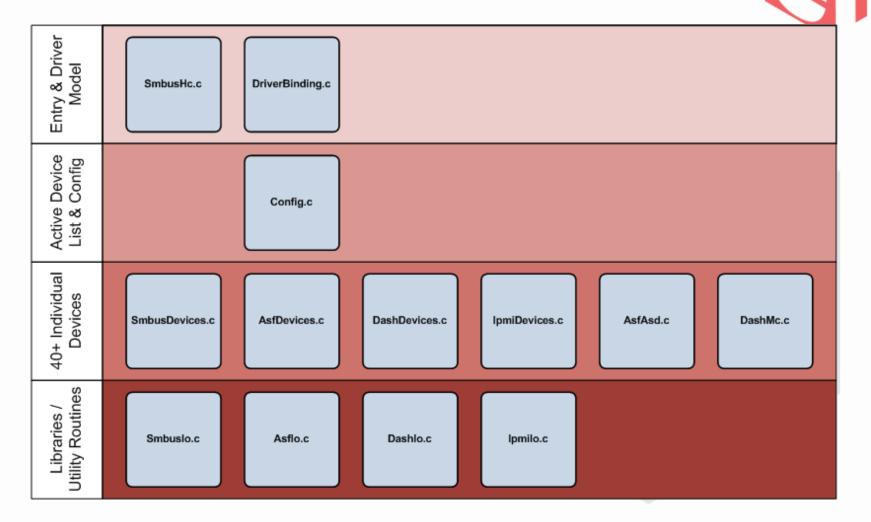


Emulate Host Controller

- 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



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!



```
c:\Sct\21a\Projects\WinHost\000\temp\ia32\SystemWinHostSec.exe
SmbusWinHostSmbusHcDxe.SmbusHcEnum: Enumerating WinHost SMBus Device
SmbusWinHostSmbusHcDxe.SmbusHcEnum:
                                       SMBus Host Controller: 5B54079D-B817-4106-
90C0-4809ECB3A1C4
SmbusWinHostSmbusHcDxe.SmbusHcEnum:
                                       SMBus Device
                                                             : USID=0xFFFFFFFF,SSD
I=0xFFFF,SSVI=0xFFFF,I=0x0004,DI=0x5555,VI=0x0001,UR=0x08,DC=0x81
SmbusWinHostSmbusHcDxe.SmbusHcEnum:
SMBus Device
 Support ARP
 Support Bold
                   : TRUE
 Address
                   : FF
SMBus UDID
Vendor Specific Id
                         : 0×FFFFFFFF
Subsystem Device Id
 Subsystem Vendor Id
                         : 0xFFFF
 Interface
                          0×0004
    Version
Device Id
                          0x5555
 Vendor Id
 Uandor Revision
 Device Capabilities
                        : 0×81
 PEC
                         : Enabled
 Addressing
                         : Dynamic Volatile
```

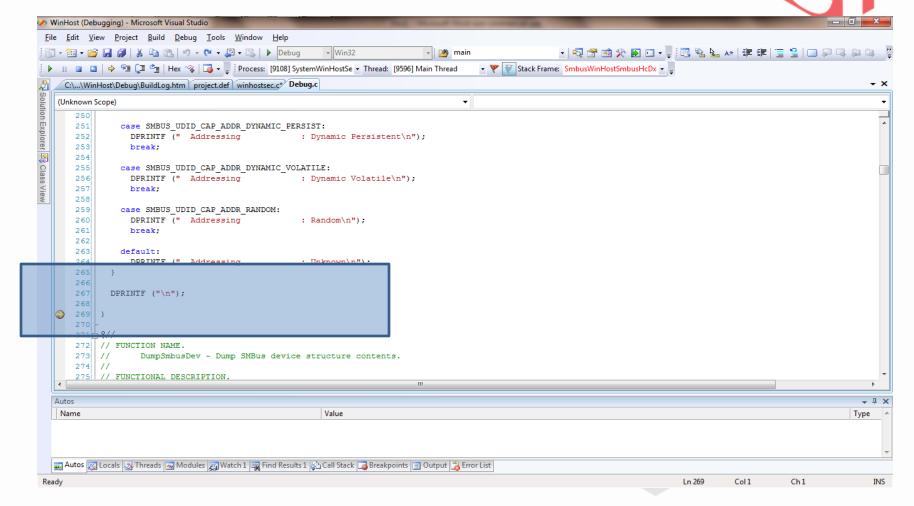
Dump out enumeration information.

Debug Using Messages or Breakpoints!

 View how commands are translated from DASH to SMBus.

```
- -
C:\windows\system32\cmd.exe - wh.bat
    Message Type:
                                                    01 00 00 00 01 00 03 01 05
    Raw Data:
DASH PLDM MESSAGE
Instance:
                                                    0 (Response)
Version:
                                                   0x03 - BIOS Control/Config
0x01 - Get BIOS Table
0x05 - Unsupported PLDM Command
00 03 01 05
Type:
Command Code:
Completion Code:
   Raw Data:
SmbusWinHostSmbusHcDxe.SmbusDevExecDashMc: Exit (Success)
SmbusWinHostSmbusHcDxe.SmbusHcExecute: Exit (Success)
SmbusBusDxe.SmbusIoExecute: Exit (Success)
AsfBusDxe.SendMessage: Exit (Success)
AsfManagerDxe.StartWatchdogTimer: Exit (Success)
DashManagerDxe.SetSmbiosTableCmd: Entry
DashManagerDxe.PldmSendMessage: Entry
DASH PLDM MESSAGE
                                                    0 (Request)
0
Instance:
Version:
                                                    0x01 - SMBIOS
0x04 - Set SMBIOS Struct Table
80 01 04 00 00 00 05 7F 04 FF FE 00 00 00 00
 Type:
 Command Code:
   Raw Data:
                                                      5C EØ 69 DF
MctpSmbusDxe.SendMessage: Entry
MctpSmbusDxe.SendMessage: Device Address = 16, Length = 25
SmbusWinHostSmbusHcDxe.SmbusHcExecute: Entry
SMBus Address:
SMBus Command:
SMBus Operation:
SMBus Packet Error Check:
                                                    0x09 - Write Block
                                                    TRUE
                                                    71 01 00 00 C0 01 80 01 04 00 00 00 00 05 7F 04 FF FE 00 00 00 00 5C E0 69 DF
Buffer:
SmbusWinHostSmbusHcDxe.SmbusDevExecDashMc: Entry
SmbusWinHostSmbusHcDxe.SmbusDevExecGetUdidGeneral: Entry
SmbusWinHostSmbusHcDxe.SmbusDevExecGetUdidGeneral: Exit (FALSE)
SmbusWinHostSmbusHcDxe.SmbusDevExecDashProcessRequest: Start Of Single-Packet Me
Smage
SmbusWinHostSmbusHcDxe.SmbusDevExecDashPldmSetDateTime: Entry
SmbusWinHostSmbusHcDxe.SmbusDevExecDashPldmSetDateTime: Exit (FALSE)
SmbusWinHostSmbusHcDxe.SmbusDevExecDashPldmSetBiosTable: Entry
SmbusWinHostSmbusHcDxe.SmbusDevExecDashPldmSetBiosTable: Exit (FALSE)
SmbusWinHostSmbusHcDxe.SmbusDevExecDashPldmSetBiosTableTags:                 Entry
SmbusWinHostSmbusHcDxe.SmbusDevExecDashPldmSetBiosTableTags:                Exit (FALSE)
```

Debug Using Messages or Breakpoints!



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.

THE STATE OF THE S

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

SMBus I/O Protocol



ASF I/O Protocol



DASH MCTP I/O Protocol



27

IPMI I/O Protocol



References



- System Management Bus (SMBus)
 Specification, Version 2.0, August 3, 2000
- UEFI Specification, Version 2.3.1, April 6,
 2011
- Platform Initialization (PI) Specification,
 Volume 5, Version 1.2, May 13, 2009