Debugging under Unified Extensible Firmware Interface (UEFI): Addressing DXE Driver Challenges

Jeff Bobzin
Sr. Director
Insyde Software
Session ID EFIS003
The first computer bug, a moth removed from the Mark II in 1947, is now at the Smithsonian museum.

Grace Hopper, whose many accomplishments included writing the first compiler, popularized the term “Computer Bug.”
Agenda

• Benefits of the UEFI Architecture
• Tips for Project Planning
• Debugging Tools

…Debugging Now
UEFI is Architected for Modularity

- UEFI is an interface specification
- Allows Driver Encapsulation
- As supported devices migrate to new market segments, the driver can easily follow
  - Same driver for Mobile
  - For Desktop/Server
  - For Embedded Too!

**Benefits of the UEFI Architecture**
UEFI Has a Defined Control Flow

- UEFI/Framework boot process goes through defined stages with clear rules for each stage

![Diagram showing the boot process stages and DXE Debug]

**Benefits of the UEFI Architecture**

- Security (SEC)
- Pre EFI Initialization Environment (PEI)
- Driver Execution Environment (DXE)
- Boot Device Selection (BDS)
- Transient System Load (TSL)
- Run Time (RT)
- After Life (AL)

**Power on** → [Platform initialization...]
     → [OS boot...]
     → Shutdown
UEFI is Architected for Debugging

- Uses clear inter-module interfaces called protocols
- Drivers often both consume and produce protocols
- Driver writer should treat platform service protocols as a ‘black-box’
Agenda

• Benefits of the UEFI Architecture
• Tips for Project Planning
• Debugging Tools
Plan Early for Debug

- When designing your board consider:
  1. Firmware Storage, initial load
  2. Debugger tool Connections
  3. Serial Debug Output

- When starting driver or application project consider:
  1. Utilize UEFI Modular Architecture
  2. Possible to test in the Shell?
  3. CPU modes – IA32, x64, EBC
Agenda

• Benefits of the UEFI Architecture
• Tips for Project Planning

• Debugging Tools
  – Hardware (Jtag) Debuggers
  – Software Debugger Tools
  – Macros from Tianocore.org
Then…

Released in 1979, the SDK-86 was Intel’s first Reference Board. It offered a full-featured debugger in ROM with keypad and serial interface. All of its debugger commands are still used in debuggers today.
Standard Today - Source-level Debugging

- Source Level View of Target on developer desktop
- Set Breakpoints in target—
  - Execution - software and hardware
  - Hardware breakpoints for memory or I/O access
- Go/step control -> step into, step over, etc.
- Dump registers and memory, dump indexed registers, dump PCI registers, etc.
- Dump UEFI tables
High-level Debugger

American Arium
JTAG Source-level Debugger
ECM-XDP3
American Arium- ITP Jtag Connection
American Arium H/W Debugger Application Interface

SourcePoint - x86 Family F Model 2 (N/P/G) - C:\Program Files\American Arium\SourcePoint-1\DXE_debug.prj

Debugging Tools

Copyright 2008 Insyde Software
Hardware Debugger Useful in All Stages

Security (SEC)  Pre EFI Initialization Environment (PEI)  Driver Execution Environment (DXE)  Boot Device Selection (BDS)  Transient System Load (TSL)  Run Time (RT)  After Life (AL)

Power on ➔ [. . . Platform initialization . . .] ➔ [. . . OS boot . . .] ➔ Shutdown

Copyright 2008 Insyde Software
Platform Software Debugger

H2ODDT™ from Insyde Software
H2ODDT from Insyde Software

Connects to Target USB Port 0

Developer System

Firmware w/ H2ODDT Included

Target Board
H2O DDT Features and Requirements

• DDT drivers are built into the firmware image which is then flashed to the platform
• Versions for IA32 and x64
• Must use target USB Port 0 (which has special debug mode)
• Debugging capability from PEI init through DXE into boot phase
• Includes debug of SMM drivers and legacy option ROMs
H2ODDT Gives Full Source View
```c
#ifndef EFI_DXE_PERFORMANCE
UINT64
Tick;
#endif

mMobStart = MobStart;

//
// Initialize Memory Services
//
CoreInitializeMemoryServices ( MobStart, MemoryBaseAddress, ...

// Allocate the EFI System Table and EFI Runtime Service Table
// Use the templates to initialize the contents of the EFI System Table

 gst = CoreAllocateRuntimeCopyPool (offsetof (EFI_SYSTEM_TABLE), ,
 ASSERT (gst != NULL);

 gRT = CoreAllocateRuntimeCopyPool (offsetof (EFI_RUNTIME_SERVICE),
 ASSERT (gRT != NULL);

 gST->RuntimeServices = gRT;
```
Complete Step Control Over Target
More H2ODDT Features

Register viewer

Memory dump viewer

Local and Global Variables

Copyright 2008 Insyde Software
H2ODDT Platform Software
Debugger Useful Throughout the Boot

Security (SEC) Pre EFI Initialization Environment (PEI) Driver Execution Environment (DXE) Boot Device Selection (BDS) Transient System Load (TSL) Run Time (RT) After Life (AL)

Power on → [ . . . Platform initialization . . . ] → [ . . . OS boot . . . ] → Shutdown
New From Insyde: Shell-level Debugger

“DDT-AD” = Debugger for Applications and Drivers
“DDT-AD” Software Debugger for Shell-Level Debug

- Developer System
- Thumb Drive w/ DDT-AD Driver
- USB0
- Board needing driver
- Host Board

Copyright 2008 Insyde Software
“DDT-AD” Features and Requirements

• Debug your drivers using any UEFI-compatible platform
• DDT-AD is loaded from UEFI Shell Prompt using Flash drive or other media
• Capability to debug anything that can be loaded from the Shell
• Same cable and source-level debug features as standard Insyde H2O DDT
DDT-AT Lets You See Inside Your Driver

- IHV APP
  - TEST APP
- IHV DRIVER
- DDT-AD
- UEFI SHELL
- UEFI API INTERFACE
- UEFI-COMPATIBLE PLATFORM

Strengths of “DDT-AD”

1. Easy setup – no need to build debug firmware image
2. Simple connection using USB cable
3. Look inside the source of the driver or application under test
4. Debug problems on any UEFI board
5. Useful for many segments – OEM/ODM and IHV, driver and application
“DDT-AD” Status

- Insyde is beginning Beta-test
- Is this tool interesting to your company?
- Send email to debug@insydesw.com
DDT-AD Used in Shell Environment

Security (SEC) | Pre EFI Initialization Environment (PEI) | Driver Execution Environment (DXE) | Boot Device Selection (BDS) | Transient System Load (TSL) | Run Time (RT) | After Life (AL)

Power on → [Platform initialization] → [OS boot] → Shutdown

Previously exposed Tiano APIs now limited
DEMO: DDT-AD Debugger
Source Debuggers - Tips and Tricks

1. Compiler optimization can make source display hard to follow
   – Local variables often held in registers, code order may be rearranged...
   – Include this statement in your source to disable optimizer:
     \#pragma optimize ("", off)

2. Include this statement in your source to hard-code a breakpoint at any point
   __debugbreak() compiler intrinsic
More Source Debugger Tips...

3. Same source can be built for native CPU or EBC binary so usually recommended to **debug drivers in native** first and then retest in EBC before release

4. Problems in EBC drivers can be checked with EBC Debugger
   - [http://www.uefi.org/events/EBC_debugger.zip](http://www.uefi.org/events/EBC_debugger.zip)
Software Debug Macros

From www.tianocore.org

Welcome!

The EFI and Framework Open Source Community is the community surrounding the open source components of Intel's implementation of EFI, officially called the Platform Innovation Framework referred to as "the Framework". (To learn more about EFI, please click here.) These components comprise the main projects hosted here at the website (www.tianocore.org). The EFI and Framework open source community is growing by the day and could use more help in driving and developing the Framework and its capabilities. To learn more about getting involved in the community, click here.

About The Community

There is a lot to learn about the the EFI and Framework open source community. Without actually joining the community, you still have access to the documentation, source and binaries available on the EFI and Framework Open Source Community Website. However, your involvement will be limited to read-only access to the tools provided by the community. The real magic of any community is when you join and have full access to the tools that the community drivers use to evolve and make their application and its accompanying documentation and tools. As a community member, you will have access to the following:

- Forums - Used for a traceable dialog between community members.
- Issue Tracking - Helps keep track of feature requests, bugs and issues with the respective project.
- Mailing Lists - Like forums but handled via email instead of web forms.
- RSS Feeds - You can subscribe to RSS feeds for news, forums and mailing lists.

Word on the Street

- Addis to BIOS
- EFI - Wikipedia
- Evolution of BIOS
- Writing an end to the BIOS

Our Community at Work

- AMI Interface
- UEFI Evaluation Kit for EFI
- More info on Insyde Software's InsydeH2O Solution
- Solving Boot
EDK Debug Macros from tianocore.org

These Macros Output to serial port

- **DEBUG (ErrorLevel, String, ...)**
  - printf-like formatted print string if ErrorLevel is active
- **ASSERT (Expression)**
  - If Expression is FALSE, then print file name and line number and halt.
- **ASSERT_EFI_ERROR (Status)**
  - If Status is not EFI_SUCCESS, then print file name and line number and halt.
- **CR (Record, Type, Field, Signature)**
  - ASSERT()s if Data Structure Signature does not match
- **EFI_BREAKPOINT ()**
  - Generate a CPU break point instruction
## Review: Debugging Tools

<table>
<thead>
<tr>
<th></th>
<th>Board Connection</th>
<th>Strengths</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hardware Debugger</strong></td>
<td>ITP Jtag</td>
<td>Full Source view, break if hung</td>
</tr>
<tr>
<td><strong>H2ODDT Platform</strong></td>
<td>USB Port 0</td>
<td>Full Source view and breakpoints with no special connector</td>
</tr>
<tr>
<td><strong>Software Debugger</strong></td>
<td>USB Port 0</td>
<td>Easy to use tool for UEFI driver and application development</td>
</tr>
<tr>
<td><strong>DDT-AD Shell-level</strong></td>
<td>USB Port 0</td>
<td></td>
</tr>
<tr>
<td><strong>Debugger</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>In-source</strong></td>
<td>Serial</td>
<td>Free but still useful in project</td>
</tr>
<tr>
<td><strong>Macros</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Copyright 2008 Insyde Software
Summary

1. UEFI architecture is modular and designed for debugging

2. Plan early so board design includes required debugger-support features

3. Several debugger tools available: choose the best tool for your project
Call To Action

• The industry is moving to UEFI and conversion has many benefits.
• Plan early to include debugging and make your UEFI project a success.
• Select one of these easy to use tools to increase productivity!
Resources

- www.uefi.org
- www.tianocore.org
- www.arium.com
- www.insydesw.com
- Win a iPod touch at next UEFI session

Visit Insyde at Booth # 320
Or Booth # 153
Please Fill out the Session Evaluation Form
Place form in evaluation box at the back of session room

Thank You for your input,
we use it to improve future Intel Developer Forum events
Questions?