Driver Development with EDKII

UEFI 2020 Virtual Plugfest
September 15, 2020
Presented by Tomas Pilar, NUVIA Inc.
Meet the Presenter

Tomas Pilar
Firmware Engineer
Member Company: NUVIA Inc.
Agenda

• Starting Resources
• UEFI Fundamentals
• Common Issues
• Force Multipliers
• Interesting Bugs
So you want to write a driver ...

Starting Resources
The Spec

https://uefi.org/uefi
https://uefi.org/specifications

- “Complete” - 38 chapters, 12 appendices, 2500 pages
- Not actually complete - not all APIs, no libraries
- Read chapters 1-11 (except 5 & 6)
- “Code First” approach adopted only very recently
- Excellent reference (corner cases exist however)
Driver Writer’s Guide

- Good introductory material
- Despite age, content still relevant
- BOLO:
  - Obsolete protocols
  - Industry not following described practices
  - EFI ByteCode (EBC)

Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.11</td>
<td>Initial draft</td>
<td>4/10/12</td>
</tr>
<tr>
<td>0.10</td>
<td>Initial draft, fixed for formatting and grammar</td>
<td>6/6/09</td>
</tr>
<tr>
<td>0.09</td>
<td>Incorporated industry review comments</td>
<td>7/23/09</td>
</tr>
<tr>
<td></td>
<td>Updated the tooling conventions.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Updated for the 1.10.14/12 release of the EFI Sample Implementation.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Updated the supported versions of Microsoft Visual Studio and Windows.</td>
<td></td>
</tr>
<tr>
<td>0.91</td>
<td>Updated for UEFI 2.0</td>
<td>10/31/06</td>
</tr>
<tr>
<td>0.92</td>
<td>New formatting</td>
<td>11/10/06</td>
</tr>
<tr>
<td>0.92</td>
<td>Removed feedback incorporated</td>
<td>1/14/07</td>
</tr>
<tr>
<td>0.94</td>
<td>Additional formatting</td>
<td>3/27/07</td>
</tr>
<tr>
<td>0.95</td>
<td>Additional formatting</td>
<td>3/23/07</td>
</tr>
<tr>
<td>0.96</td>
<td>Additional formatting</td>
<td>4/25/09</td>
</tr>
<tr>
<td></td>
<td>Clarified role of EDK as being implementation specific and added definitions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Removed many references to the meaning of the implementation specific code examples (no longer treated as text from the EBT Specification)</td>
<td></td>
</tr>
<tr>
<td>0.87</td>
<td>Updated for UEFI 2.1 and EDK II</td>
<td>6/25/10</td>
</tr>
<tr>
<td>0.86</td>
<td>Review feedback incorporated, additional formatting</td>
<td>2/12/10</td>
</tr>
<tr>
<td>1.00</td>
<td>Review feedback incorporated, additional formatting</td>
<td>2/27/10</td>
</tr>
<tr>
<td>1.01</td>
<td>Removed feedback incorporated</td>
<td>3/6/10</td>
</tr>
<tr>
<td>1.1</td>
<td>Conversion to GitHub markdown format</td>
<td>4/16/10</td>
</tr>
</tbody>
</table>
Starting Resources III

Mailing List

- Where development and maintenance happens
- Plenty of people happy to answer questions
  … provided at least a rudimentary bit of research was done first

<table>
<thead>
<tr>
<th>Group Information</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://tianocore.org">http://tianocore.org</a></td>
</tr>
<tr>
<td>556 Members</td>
</tr>
<tr>
<td>28,304 Topics, Last Post: 10:38am</td>
</tr>
<tr>
<td>Started on 2016-07-29</td>
</tr>
<tr>
<td>Feed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
</tr>
</thead>
<tbody>
<tr>
<td>2020</td>
<td>988</td>
<td>1,499</td>
<td>1,674</td>
<td>1,688</td>
</tr>
<tr>
<td>2019</td>
<td>1,616</td>
<td>1,212</td>
<td>1,010</td>
<td>1,623</td>
</tr>
<tr>
<td>2018</td>
<td>1,091</td>
<td>1,285</td>
<td>1,309</td>
<td>1,023</td>
</tr>
<tr>
<td>2017</td>
<td>870</td>
<td>1,035</td>
<td>1,361</td>
<td>942</td>
</tr>
</tbody>
</table>
We are not in Kansas anymore ...

UEFI Fundamentals
Architecture Boot Flow

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

Power on → [ . . . Platform initialization . . . ] → [ . . . OS boot . . . ] → Shutdown
Driver Execution Context

**DXE_CORE**
- Dispatcher
- Performance Monitoring
- Core Memory Management

**DXE_DRIVER**
- Timers
- Consoles (Serial/VGA)
- HII Database
- Protocol Management
- ACPI Tables
- ESRT
- Network Stack (Core)
- ...

**UEFI_DRIVER**
- Network Cards
- Usb Peripherals
- Flash Update Drivers
- Input Devices
- Mass Storage Drivers
- Network Stack (Components)
- ...

**UEFI_APPLICATION**
- BMC Agent
- Boot Manager
- Boot Loaders (grub)
- Platform Key Manager
- Setup Utility
- Capsule Utility
- UEFI Shell
- ...

BootServices fully operational

ReadyToBoot signalled

ExitBootServices signalled

© Copyright 2020 NUVIA Inc.
No memory protection

- Zero page might be mapped - NULL pointers, Interrupt Vector Table
- Can scribble anywhere in memory - Service Table, Rootkits etc.
- (Optional) Stack and heap protection

No privilege differentiation

- Files have no privileges, write and execute anything
- Secure Boot is not automatic!
Environment II

Single Threaded Execution

- Single queue data transfers
- No need for parallelization techniques

No Interrupts

- Actually, no Interrupts for you specifically (why?)
- Drivers are pull-based
- Data transfer driven by regular poll driven by platform stack

Event System Pre-emption

- Events can pre-empt execution at any point to execute callbacks
- Shared resources can get trampled over
Protocols

“How do I do X? There is a protocol for that”

- All interfaces are implemented in form of protocols
- Protocol structs can contain data, methods, pointers ...
- Extensible - trivial to make new protocols
- Discoverable - Boot Services include protocol services (un/install, search)

```c
#define ISENGARD_PROTOCOL GUID
    { 0x...

EFI_GUID gIsengardProtocolGuid;

typedef struct _ISENGARD_PROTOCOL {
    EFI_STATUS (*BuildArmy)(...);
    // OtherStuff
} ISENGARD_PROTOCOL;
```

```
FooProtocol.h
```
Handles

• There are no naked protocols
  – Identify protocol instances
  – Semantically group protocols
• Automatic and transparent
• EFI_HII_HANDLE != EFI_HANDLE
• SHELL_HANDLE != EFI_HANDLE
Events

- Asynchronous support
- Signal/Wait
  - More dynamic than protocols
  - More anonymous than protocols
- Timers
  - Once (Timeouts)
  - Recurring (Polling)
- Can be grouped using a GUID and signalled together
  - System-wide “announcements”
Task Priority Levels

- Prioritisation of callbacks and events
- TPL_CALLBACK and TPL_NOTIFY have event queues
Task Priority Levels

- Prioritisation of callbacks and events
- TPL_CALLBACK and TPL_NOTIFY have event queues
Now we transpile the .i file to .iii and carefully add the eye of newt ...

Human Interaction Interfaces
Modularise the forms

- Look and feel (Platform)
- Strings (Static/Driver)
- Form logic (Static/Driver)
- Configuration (Driver or Platform)
Things that will eat your face in the dark ...

Common Issues
“Be conservative in what you send, be liberal in what you accept”

- No Platform Ever
**Inconsistent Nomenclature**

**BIOS**
- (Not!Uefi) Legacy 16 bit, real mode boot execution environment
- (Not!OptionROM) Monolithic platform firmware loaded from baseboard
- (Not!CPUFirmware) Later stage generic firmware loaded by earlier stage platform firmware
- (Not!OS) Pre-boot code

**UEFI**
- (Not!BIOS) Extensible Firmware Interface Specification
- (Not!OS) Pre-boot code
- (Not!Uboot) EDK II based platform firmware

**Option ROM**
- (Not!UEFI) Legacy 16-bit executable blob rather than UEFI Driver
- (Not!BIOS) Vendor code loaded from device flash
- (Not!PlatformFlash) Expansion ROM BAR or flash on peripheral devices
# HII Madness

<table>
<thead>
<tr>
<th>Normal Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Packed configuration blobs and string alignment</td>
</tr>
<tr>
<td>• UTF-16LE != UCS-2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hard Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Build errors of “Wagnerian Fierceness”</td>
</tr>
<tr>
<td>• Callbacks are unreliable</td>
</tr>
<tr>
<td>• It matters where you publish the package and the access protocol</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“2020” Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Values shared across formsets</td>
</tr>
<tr>
<td>• Patching IFRs dynamically</td>
</tr>
<tr>
<td>• RESET_REQUIRED does not ensure PCI reset</td>
</tr>
</tbody>
</table>
Enhanced BIOS Code

- Closed source
- Just close enough to EDK2 main tree to be maddening
Joys of OEM Deals

- VFR Content
- Strings Translations
- Proprietary Management Features
- Configuration Parity
- Expose the same information through a bazillion different interfaces
- “Thou shalt not write a runtime driver!”
Debugging On A Bad Day

Bad news - your platform does not have a crash handler.

Enjoy your invalid X64 instruction.

Type: 2
Sev: 80
Class: 3
SubClass: D
Op: 6
Debugging On A Good Day

Good news - Your platform has a crash handler, you get registers, stack trace and current task priority level.

Bad news - The platform crashed in closed-source “OEM Enhancement” or the BMC Agent after your driver fed it some garbage data.

Go back to square one. Do not pass go. Do not collect $200.
Driver Model

• Bunch of old protocols
  – Might crash platform if implemented
• Quite a few protocols are not useful
  – Might crash platform if not implemented
• Multiple Devices
  – Might load one driver, might replicate
  – Devices might load different versions of driver
  – Driver Override Protocols not always honored
  – Shared resources?
  – Inter-device communication?
Network Stack

> You want to write a network driver
> Spec/DWG says you should prefer SNP over UNDI
> NII is optional, used to expose PXE blob for UNDI
> BIOS not consider you a network card without NII
> If you install NII, BIOS will try and build its SNP on it
> PXE blob location in NII is a raw pointer
> You don’t have PXE blob as SNP device
> BIOS might ignore NII already open BY_DRIVER
> PXE Client will use NII version fields for PXE/BOOTP
> You are the only person writing an SNP driver
Self-Certification Tests

Mostly designed for IBVs or OEMs
- Test API, not functionality
- Small set runnable by IHVs

Now actually open-source (Good!)
ExitBootServices Delights

• No Timers
  – No timeouts
  – No polling
• No Memory Allocation
  – Hope you have no dynamic message boxes
• What if there is no ExitBootServices?
  – Never do device-driven periodic DMA
  – Limit extant descriptors
Life Pro Tips ...

Force Multipliers
The Plugfest

• People
  – You get to know the engineers on the other side
  – Those email addresses are people too
• Hardware
  – Much easier to patch pre-production BIOS
  – Ask Ard about Chucky
• It’s not just you
  – War stories are gold
Embrace the Print

- Print everywhere
  - Info at start of function
  - Print return codes
  - Print strings, device paths, stages of functions
- Filter by verbosity, make a library out of it
- Use TPL_HIGH_LEVEL
- Output to all kinds of places
  - Shunt down the serial port (bypass SERIAL_IO)
  - Write to consoles
  - Write to memory and dump once you can
Automate Everything

- Build Documentation
- Build Driver
- Host Based Tests (Fuzzing, Unit Tests …)
- Build Checks (size, strings, symbols …)
- Load Driver
- Bind Driver
- Device Passthrough
- Run empty VM
- Static Analysis
- Rebase upstream
- Commit
- Build OVMF
- Release

- Publish Promote
- System Level Tests
- HII Tests
- Protocol Tests
- Flash Update

© Copyright 2020 NUVIA Inc.
Code Intelligence

• Doxygen (Driver & Upstream)
• Static Analysis
• IDE with semantic understanding
• Modularise and encapsulate in Libraries (modules are cheap)
  – Debug information
  – Mailbox communication
  – Resources handling
• Store .uni files a UTF-8
• Build only your module (build -m foo.inf) in automation
Crash Handlers

• -b DEBUG
  – (.debug = .efi + symbols)
  – Load in GDB (disas /s)

• Learn (a bit of) architecture and calling conventions
  – Which registers correspond to parameters, return values ...
  – AFAFAFAF is poison

• Ask your new Plugfest friends to decode a crash for you
Create Protocols & Handles

• Protocols are your friends
  – Private structs
  – Mailbox communication
  – Version, Checksum and Magic!

• Handles are free
  – Separate logical parts of your device (PCI, NIC, HII, Flash)
Do not try this at home ....

Interesting Examples
## Interesting Examples

<table>
<thead>
<tr>
<th>Description</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNP.Mode.MaxPacketSize = 1514</td>
<td>No iSCSI disk in Windows</td>
</tr>
<tr>
<td>Improve SNP efficiency</td>
<td>DoS platform with ARP</td>
</tr>
<tr>
<td>SNP &amp; NII on different handles</td>
<td>iPXE scrubs the iBFT table</td>
</tr>
<tr>
<td>Periodically refresh HII_CONFIG_ACCESS</td>
<td>Setup Utility now blinks once a second</td>
</tr>
<tr>
<td>Forget EFIAPI function attribute</td>
<td>Crash in NBP which is still using your SNP</td>
</tr>
<tr>
<td>Use grub2.02 beta2</td>
<td>MNP steals data from grub</td>
</tr>
<tr>
<td>BIOS changes MNP poll timeout to 1s</td>
<td>Standard TFTP server retransmit is 1s</td>
</tr>
<tr>
<td>Publish FMPv3 descriptors</td>
<td>Platform crashes when gathering inventory</td>
</tr>
</tbody>
</table>
Questions?
Thanks for attending the UEFI 2020 Virtual Plugfest

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

presented by

NUVIA