Innovative Software Tools & Methods to Profile, Test and Optimize UEFI Firmware
Improving Test Coverage and Debug Results

Presented by Kevin Davis (Insyde Software)
Agenda

• Introduction
• Intel Processor Trace Buffer
• Driver Profiling
• Code Coverage Display
• Questions?
Debug Models

• Hardware Level Debuggers
  – Physical Access to board required
  – Complexity of connection to board
  – No or minor impact to software execution

• Software Level Debuggers
  – No need to have physical access to the board
  – Some impact to software execution
  – Complexity of sharing a processor
Define Software Terms

• **Host Machine** – runs the User Interface of the Software Debugger to the engineer

• **Target Platform** – runs the BIOS interface of the Software Debugger
Innovation of Branch Tracing

• Previously to trace, SW debuggers broke on either each instruction or at branches
  – On each break:
    • Transmit data to host or store locally
    • Check various debug control conditions
    • Restart next instruction
  – Very slow execution
• Now with Branch Tracing
  – SW debugger stops on a condition
  – Unwinds execution trace to previous IP or size of the Trace buffer
  – Target code executes much faster
Branch Records

32-bit Branch Trace Record Format

64-bit Branch Trace Record Format
Solving new problems

• More features can be created in a software debugger
  – Unravel call trees
  – Highlight areas called by Oss
  – Find long hardware I/O loops
  – Figure out where a crash started
  – Test sequence code coverage
  – Profile specific drivers
Profiling Drivers
Steps to profile a driver

**Update to Bios**
- Update debug engine to support Profiling
- Build bios with non-optimized and generating debug info

**Debugger**
- Setup Profiling buffer size from Project Setting dialog
- Enable/Disable Profiling from Profiling menu

**Log Analysis**
- Select Profiling Analyzer from Profiling menu to start profiling analysis
- The reports will be displayed on Report view after completion
Setup Trace Buffer sizes

Profiling Driver Options

- PEI Buffer Size (24K ~ 4M) 200000
- DXE Buffer Size (24K ~ 8M) 400000
Profiling Execution Flow Example

- Execution path displayed in tree view
- Executed code colored in green
Code Coverage
Need for Code Coverage

• Validate that all code is being tested
  – If code isn’t being tested during validation, code path is either:
    • Unnecessary – remove it!
    • Not being tested – probably buggy, might cause problems in the field
      – Fix the test & test the code
Code Coverage Example

- Executed
- Not executed
- undetermined

Coverage information by module/driver

Summary of functions in a module/driver
Call to action

• Contact your Debugger vendor
  – Ask about their support for tracing code execution
• Profile your drivers to understand their execution flow
• Verify that your code is executed during your testing
Thanks for attending the UEFI US Fall Plugfest 2016

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

References

Pointers to more information
References

- Material heavily borrowed from:
  - Intel 64, ia, 32 Architectures Software Developer Manual #325462
  - Insyde Software DDT Developer Manual
- Pointer
  - https://www.insyde.com/products/developertools