presented by

BYOSOFT™
百 敖 软 件

UEFI and IoT: Best Practices in Developing IoT Firmware Solutions

Spring 2017 UEFI Seminar and Plugfest

March 27 - 31, 2017

Presented by Hawk Chen (Byosoft)

Agenda

- Introduction
- Background
- Boot Performance Tuning
- Easy Customization with Intel FSP
- Summary / Questions

Introduction

Introduction

Why we are talking about this?

- IoT devices for vertical segment bring many different requirements and challenges to the UEFI Firmware solution
- Easy customization is important for scaling out of IoT devices
- To share our best practices in an effort to simplify the implementation of the UEFI Firmware solution on IoT devices

Background

Background

IoT vertical segments have different focuses and needs tailored firmware solutions

Industrial & Energy

Transportation

Retail

User experience (Fast boot / Touch Screen)

Security (Protected boot & storage / Device Identification / Trusted Execution Environment)

Easy Customization and Differentiation

Power/Performance Optimization

Boot Performance Tuning

Boot Performance Tuning

Performance Measurement

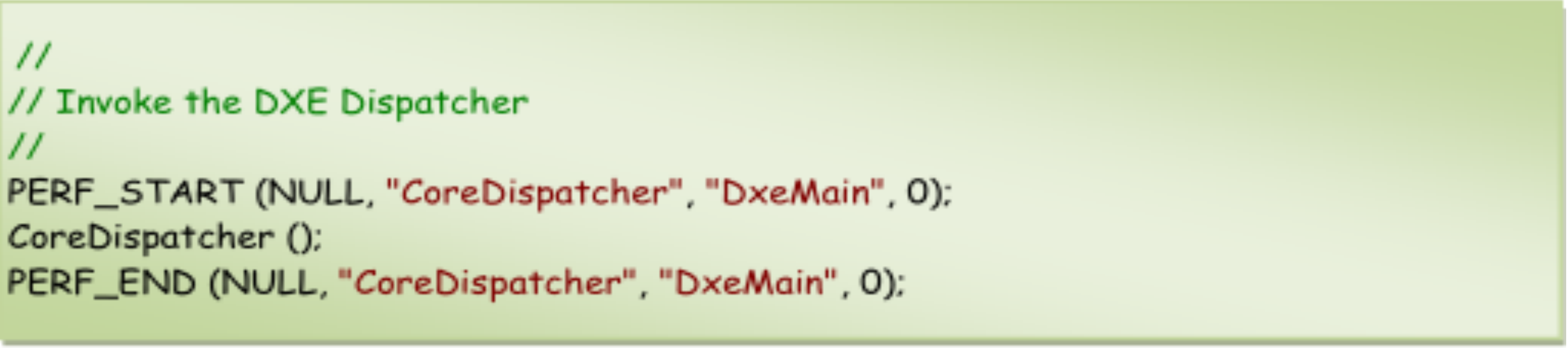
Performance Optimization

Performance Measurement

Intel UDK core provides one infrastructure to measure performance in pre-OS phase

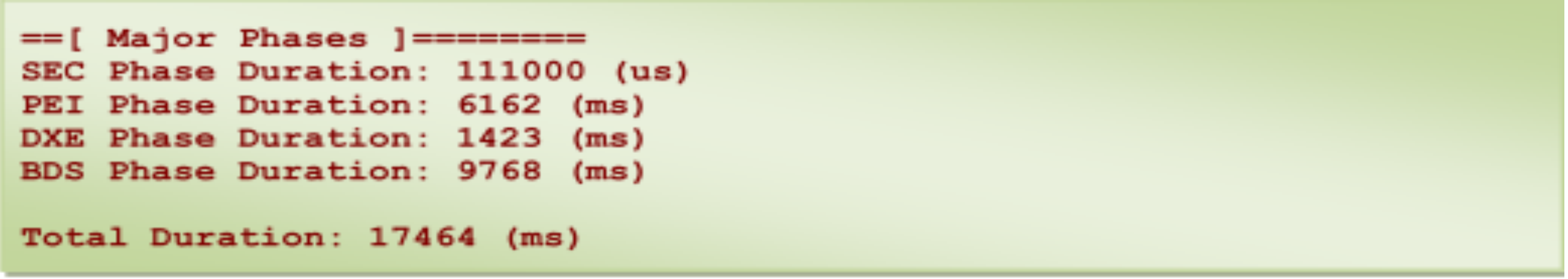
- Uses PERF_START & PERF_END macro to measure the time and cost during the traced execution range, and then creates a named tracing record
- Uses DP under UEFI shell to view all of the tracking records

Performance Measurement



```
//
// Invoke the DXE Dispatcher
//
PERF_START (NULL, "CoreDispatcher", "DxeMain", 0);
CoreDispatcher ();
PERF_END (NULL, "CoreDispatcher", "DxeMain", 0);
```

Tracing Macro Sample

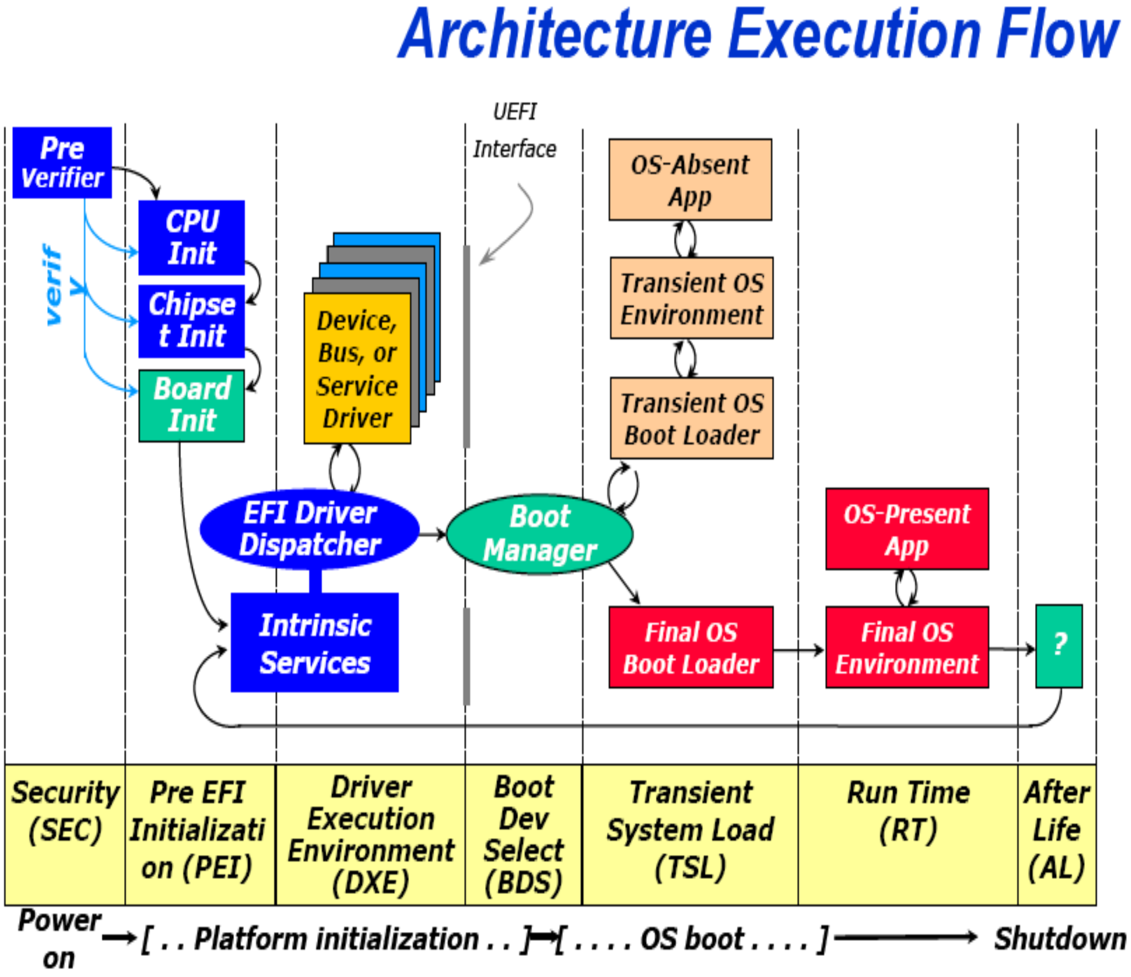


```
==[ Major Phases ]=========
SEC Phase Duration: 111000 (us)
PEI Phase Duration: 6162 (ms)
DXE Phase Duration: 1423 (ms)
BDS Phase Duration: 9768 (ms)

Total Duration: 17464 (ms)
```

DP Output Sample

Performance Optimization



- Applicable to different boot phases
- Applicable to different kinds of platform hardware and UEFI Firmware implementation, without impacting UEFI compatibility
- Consider boot performance as early as platform design phase

Performance Optimization - SEC

Phase	Optimization
SEC	Configure BFV Flash area as WP (Write Protect), after enabling CAR as temporary memory
	Enable SPI prefetching and configure with maximum clock
	Initialize the BSP with maximum speed

Performance Optimization - PEI

Phase	Optimization
PEI	Initialize SATA to spin up HDD as early as possible
	Light up display panel as early as possible
	Configure Firmware flash area as WP (Write Protect) after complete memory sizing
	Organize the FLASH layout effectively: ■ Only report FvMainCompact FV through EFI_PEI_FIRMWARE_VOLUME_INFO_PPI to have Pei Core process this single one ■ Only build FV HOBs with FVs that contain DXE drivers such as FvMain

Performance Optimization - DXE

Phase	Optimization
DXE	Only report variable FLASH area when initialize FVB services

Performance Optimization - BDS

Phase	Optimization
BDS	Use GOP driver instead of VBIOS for UEFI boot only
	Utilize non-blocking storage IO for SATA device
	Avoid clearing the 1st 640KB memory for UEFI only boot, or use hardware based memory clearing if applicable
	Enhance boot path to only initialize and configure the associate device for selected boot option
	Minimize USB timing for USB related drivers if Intel USB controller is available
	Minimize device hardware reset timing by staged initialization
	Use cached data to reduce device training time, such as memory, CPU BIST and so on

Example: IoT IVI After tuned

Boot time: 6s -> 2s after tuned

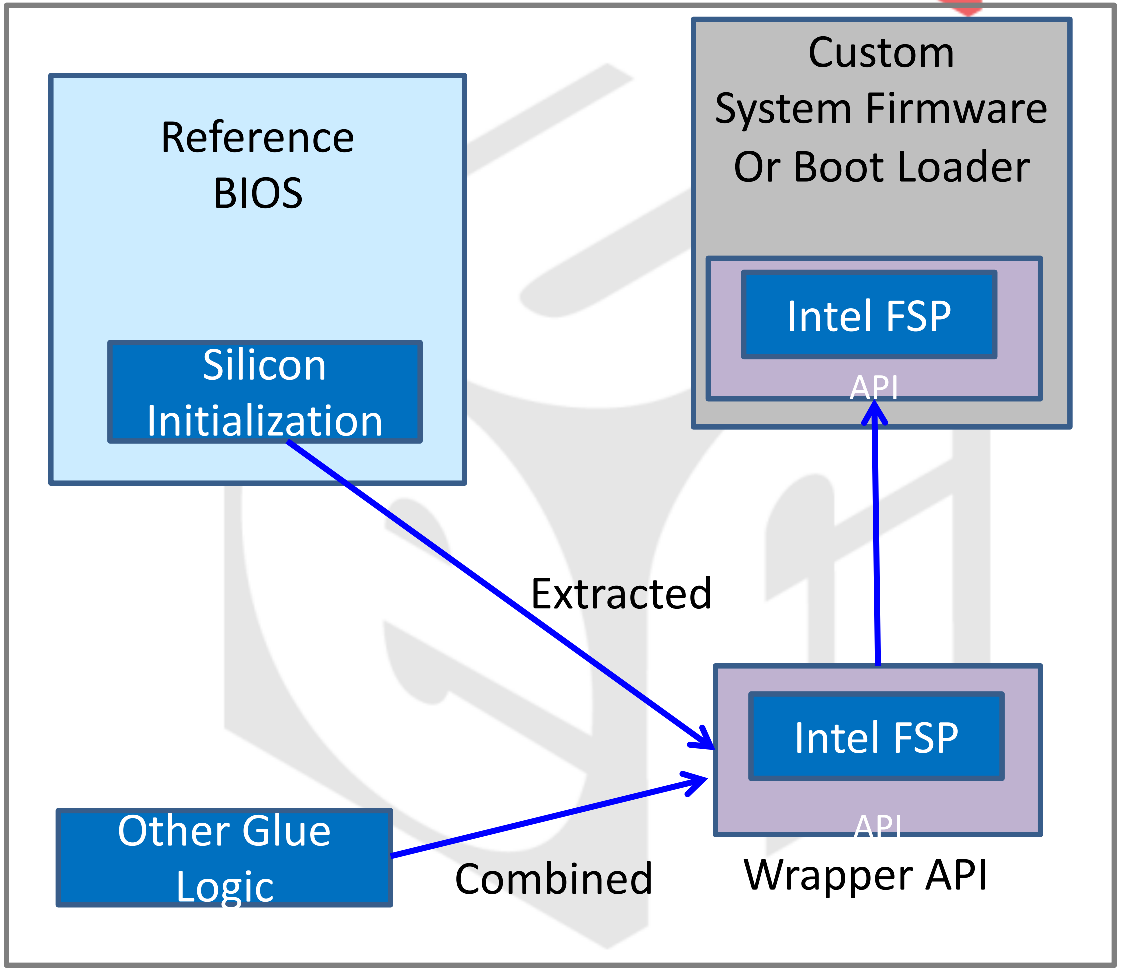
Component	Configuration
Processor	Intel® Atom™ Processor E3827 2C/2T
Memory	2G/1 Channel (DDR3-1066/1333), Memory down
Flash	8MB 50MHZ SPI Flash
Storage	eMMC 8G / 16G for primary OS and applications
Graphics	Intel ® Integrated Graphics (HDMI to RGB,HDMI to CVBS, HDMI to YPbPr, HDMI to AV)
Operation System	Android 4.2 (UEFI Boot)

Easy Customization with Intel FSP

What’s Intel FSP ?

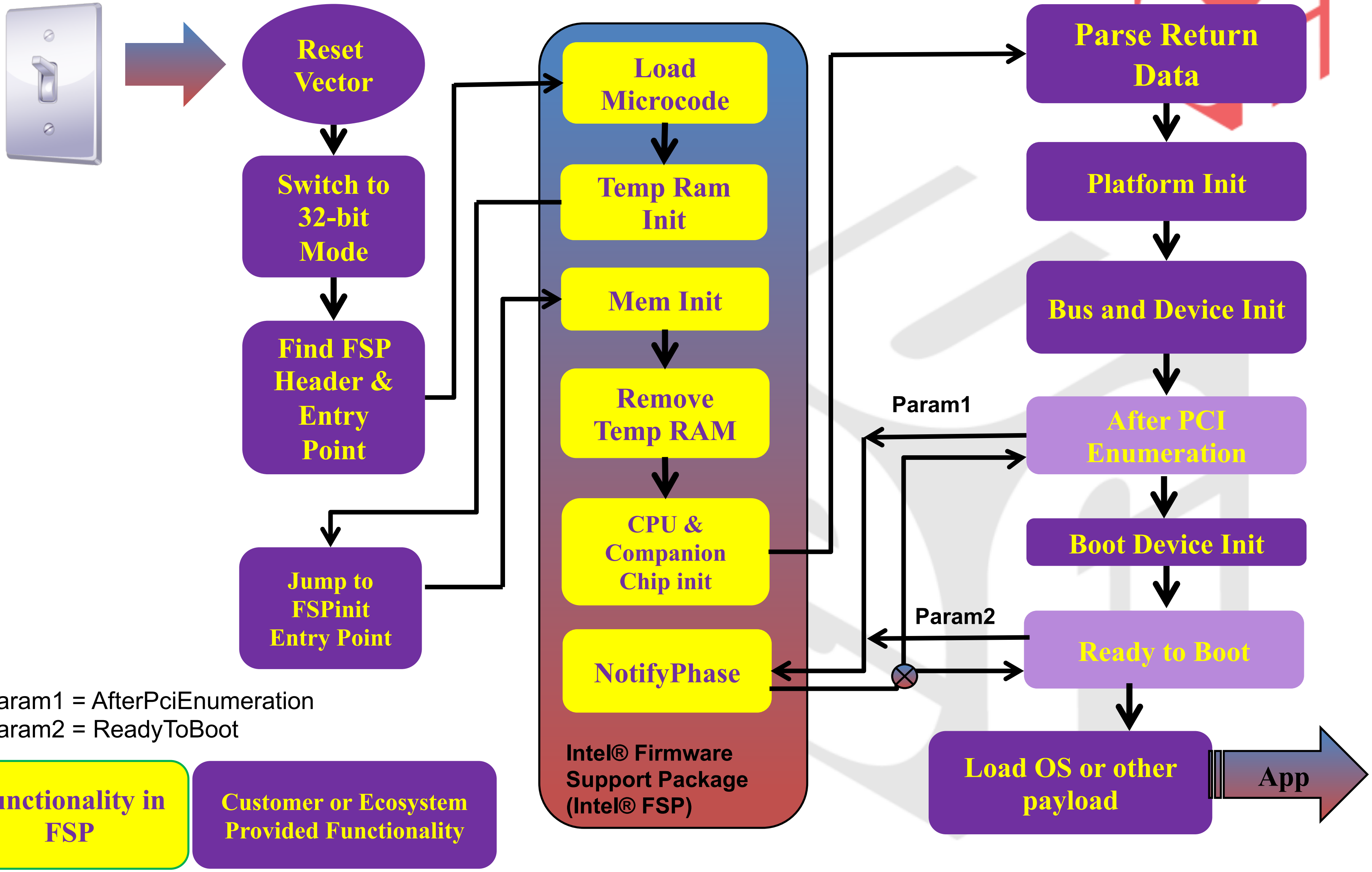
- FSP = **F**irmware **S**upport **P**ackage
- FSP’s small subset of UEFI Firmware solution
- Not a boot loader, but simply CPU, Chipset, and memory initialization code
- Released as binary with API wrapper
- Only abstracts the firmware IP
- Customers require some system firmware infrastructure used with FSP

FSP is NOT a boot loader, and requires integration with a custom system firmware infrastructure



Intel FSP Usage Model

A Sample Boot Flow Involving FSP v1.0



Param1 = AfterPciEnumeration
Param2 = ReadyToBoot

Technique Highlights for UEFI + FSP

- Easy integration of FSP into UEFI firmware solution
- Small footprint with fast execution
- Easy platform customization via a stand-alone configuration tool
- Meets diverse requirement of IoT devices with non-PC design, such as DSS, IVI, network gateway, storage, etc.,
- Keep UEFI compatibility as traditional UEFI firmware solution

Summary and QA’s

- Boot Performance tuning is conducted on platform level and can leverage many common practices in different boot phases.
- Platform customization with Intel FSP can help to improve efficiency and flexibility for IoT device firmware enablement
- Questions?

Reference Information

Document	Location
Reducing Platform Boot Time UDK 2010 Based Performance Optimization	http://www.intel.cn/content/dam/www/public/us/en/documents/white-papers/reducing-platform-boot-time-paper.pdf
PI (Platform Initialization) Specification	http://www.uefi.org/specs/ This is where the terms SEC, PEI, DXE, and BDS are defined and referenced.
UEFI Specifications	http://www.uefi.org/specs/ This is the OS interface and runtime EFI stuff.
UEFI Firmware	https://technet.microsoft.com/en-us/library/hh824898.aspx
Intel FSP	http://www.intel.com/content/www/us/en/intelligent-systems/intel-firmware-support-package/intel-fsp-overview.html

Thanks for attending the Spring 2017 UEFI Seminar and Plugfest

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

presented by

BYOSOFT™
百敖软件