Intel® Firmware Support Package (Intel® FSP)

A Rapid, Competitive, and Scalable Firmware Solution for Intelligent Systems Ecosystem

Intel® Intelligent Systems Summit

Intel® Intelligent Systems: A new era in embedded computing

328095-001

Legal Disclaimer

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL® PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. INTEL PRODUCTS ARE NOT INTENDED FOR USE IN MEDICAL, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS.

Intel may make changes to specifications and product descriptions at any time, without notice. All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.

Intel, processors, chipsets, and desktop boards may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

All code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands may be claimed as the property of others.

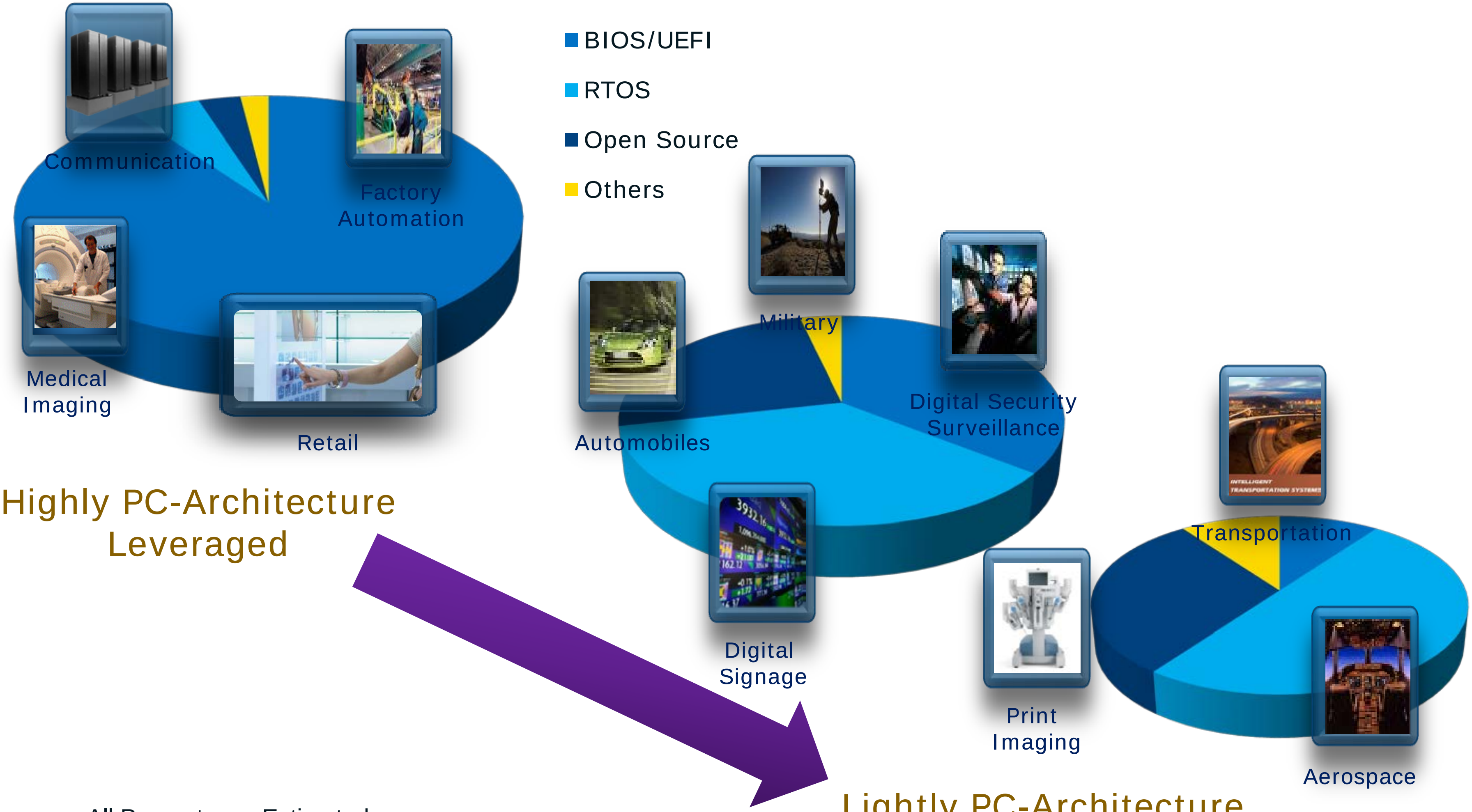
Agenda

- Introduction
- Philosophy and Principles
- Programming Interface and Data Structure
- Development Options
- Summary

Agenda

- **Introduction**
- Philosophy and Principles
- Programming Interface and Data Structure
- Development Options
- Summary

Firmware Solutions in the Intelligent System & Embedded Space



■ BIOS/UEFI

■ RTOS

■ Open Source

■ Others

Highly PC-Architecture Leveraged

Lightly PC-Architecture Leveraged

All Percentages Estimated

Comparing Intel Intelligent Systems Group (ISG) Firmware Solutions

BIOS

Full solution, all the features, ready to use, with licensing and royalties from 3rd parties

Intel® Boot Loader Development Kit (Intel® BLDK)

No frills, kit for customization, plus parts and tools, and prebuilt kit

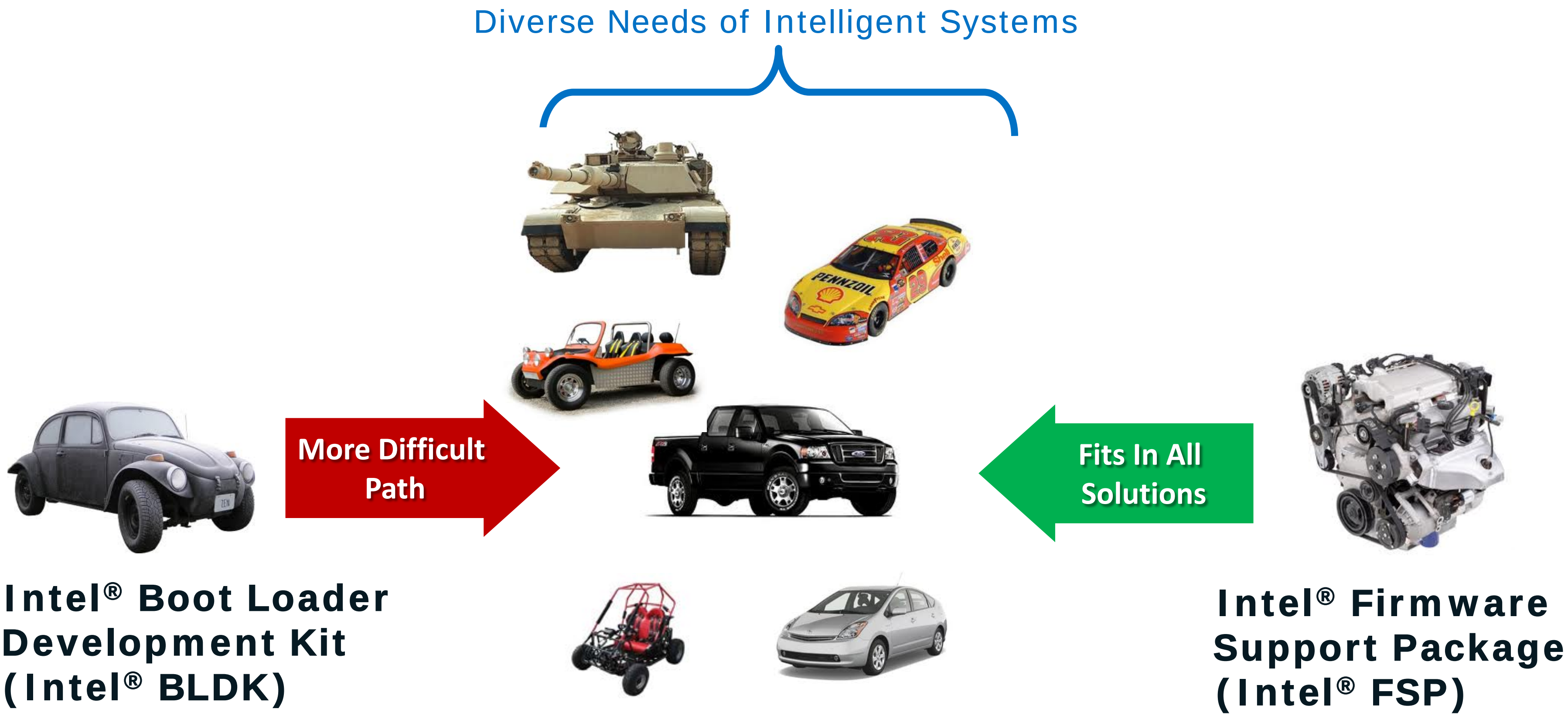
New!

Intel® Firmware Support Package (Intel® FSP)

Just the engine – can be put into any vehicle, but customer provides the vehicle

Intelligent Systems Group is transitioning enablement of BIOS alternatives from Intel BLDK to Intel FSP

Meeting Diverse Needs of ISG Customers



Diverse Needs of Intelligent Systems

More Difficult Path

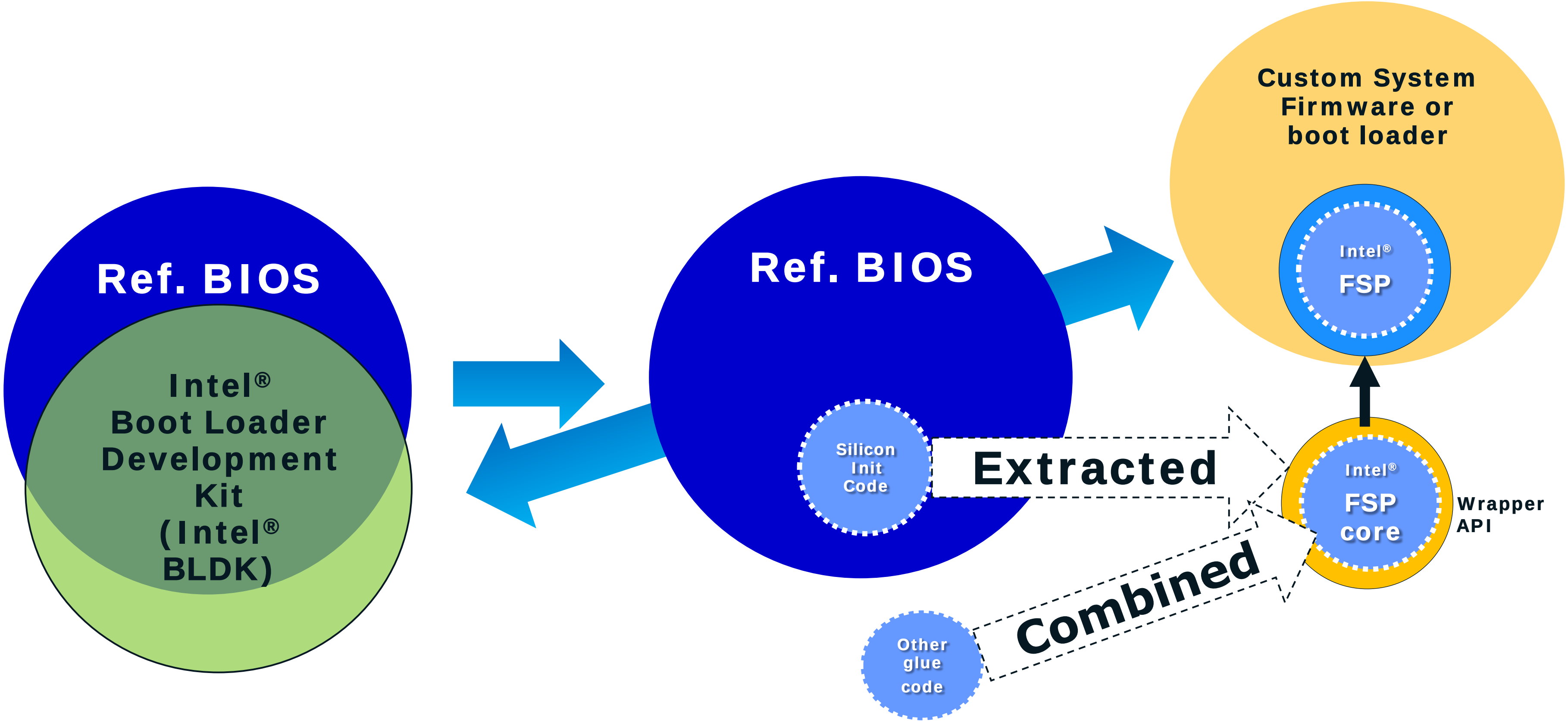
Fits In All Solutions

Intel® Boot Loader Development Kit (Intel® BLDK)

Intel® Firmware Support Package (Intel® FSP)

Intel FSP provides greater flexibility to meet diverse requirements of Intelligent Systems

Intel® Firmware Support Package (Intel® FSP)



ISG Boot Loader Implementations Transitioning from Intel BLDK to Intel FSP

What is Intel® Firmware Support Package?

Intel® Firmware Support Package (Intel® FSP) includes:

- A binary file
- An integration guide
- A rebasing tool
- An IDE configuration tool / Boot Setting File (BSF)

Provide silicon initialization code:

- Initializes processor core, chipset as explained in BIOS Writers' Guide
- Is relocatable in ROM
- Can be configured for platform customization

Boot loader agnostic and can be easily integrated with many options:

- Open source boot loaders: Coreboot, U-boot, etc.
- RTOS
- Others

What is NOT

Intel® Firmware Support Package is **NOT** a stand-alone boot loader

Intel® Firmware Support Package needs to be integrated into a boot loader of your choice:

- BIOS
- Open Source options (Coreboot, U-Boot, etc.)
- RTOS
- Proprietary, etc.

Agenda

- Introduction
- **Philosophy and Principles**
- Programming Interface and Data Structure
- Development Options
- Summary

Philosophy

There are …

- plenty of smart firmware engineers
- comprehensive specifications and standards
- successful implementation examples using various boot loaders.

There isn't …

- enough open technical information to program a new silicon

Therefore …

- Intel provides what Intel knows the best, and let the ecosystem do what they are the best at

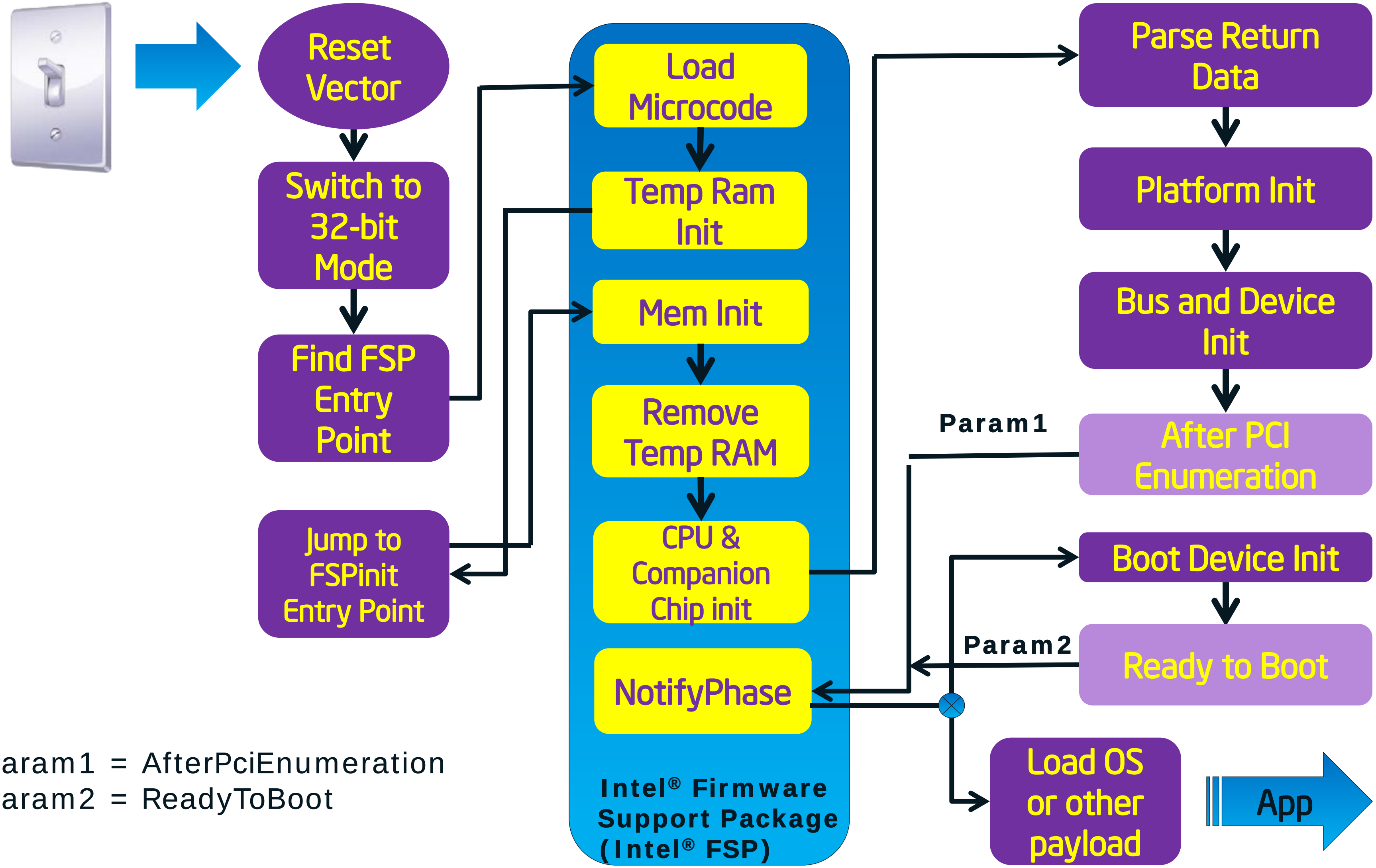
Objectives

- **Lower the threshold in Intel® architecture adoption**
  - We recognize that the ecosystem deems firmware as one of the hurdles to adopt Intel architecture in the embedded space.
- **Offer scalable and easy-to-adopt FW solutions**
  - We recognize that there are many types of customers, vendors and service providers in the ecosystem.
- **Enable and grow the ecosystem**
  - Intel will work with the ecosystem to serve everyone’s needs.

Agenda

- Introduction
- Philosophy and Principles
- **Programming Interface and Data Structure**
- Summary

A Sample Boot Flow Involving Intel® Firmware Support Package (Intel® FSP)



Reset Vector

Switch to 32-bit Mode

Find FSP Entry Point

Jump to FSPinit Entry Point

Load Microcode

Temp Ram Init

Mem Init

Remove Temp RAM

CPU & Companion Chip init

NotifyPhase

Intel® Firmware Support Package (Intel® FSP)

Parse Return Data

Platform Init

Bus and Device Init

Param1

After PCI Enumeration

Boot Device Init

Param2

Ready to Boot

Load OS or other payload

App

Param1 = AfterPciEnumeration
Param2 = ReadyToBoot

Intel® Firmware Support Package (Intel® FSP) Integration

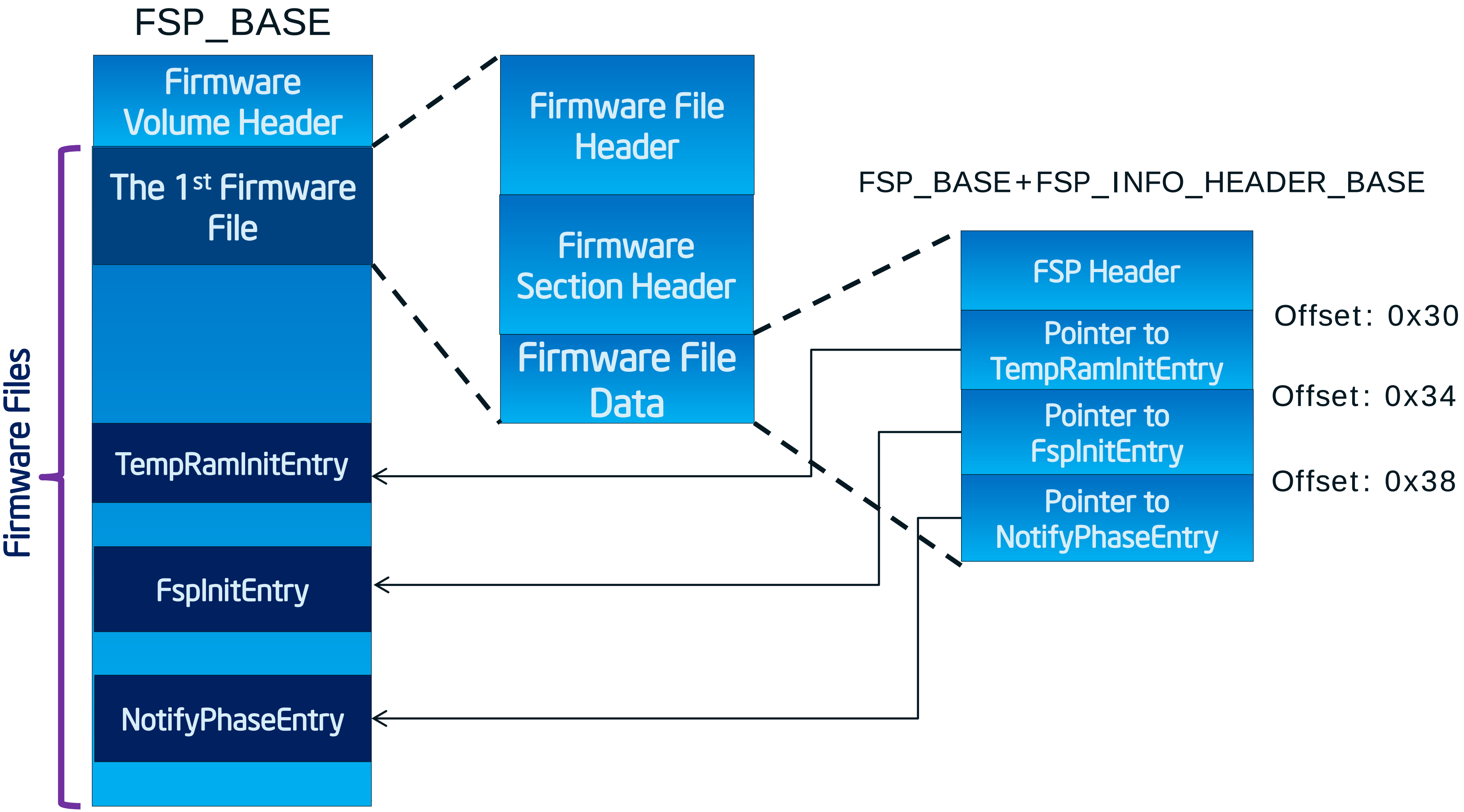
- Find Intel® Firmware Support Package (Intel® FSP) Entry Points
- Call Intel FSP Functions
- Handle Return Data
- Customize Intel FSP Internal Configurations
- Change Position of Intel FSP in your ROM

Intel® Firmware Support Package (Intel® FSP) Organization & Format

- The Intel® Firmware Support Package (Intel® FSP) binary follows the UEFI Platform Initialization Firmware Volume Specification format.
- The Firmware Volume (FV) is described in the Volume 3: Shared Architecture Elements specification.
  - ***FV is a way to organize/structure binary components and enables a standardized way to parse the binary and handle the individual binary components that make up the FV.***

UEFI specifications can be downloaded from:
http://www.uefi.org/specs/

Intel® Firmware Support Package (Intel® FSP) Binary Structure



Discover Intel® Firmware Support Package (Intel® FSP) Entry Points

The Hard Way:

Find the Intel® Firmware Support Package (Intel® FSP) = Find the Intel FSP Header within the FV

- Use EFI_FIRMWARE_VOLUME_HEADER to parse the FSP FV header, and skip the standard and extended FV header
- Locate the EFI_FFS_FILE_HEADER
- Locate the EFI_RAW_SECTION header
- Locate the data in the Raw section, and FSP_INFORMATION_HEADER is there.

Discover Intel® Firmware Support Package (Intel® FSP) Entry Points

The “Easy” way:

- FSP header (FSP_INFO_HEADER_BASE) is an offset from FSP_BASE, current value is 0x94
- Pointer to FSP “TempRamInitEntry” can be found at FSP_BASE + FSP_INFO_HEADER_BASE + 0x30
- Pointer to FSP “FspInitEntry” can be found at FSP_BASE + FSP_INFO_HEADER_BASE + 0x34
- Pointer to FSP “NotifyPhaseEntry” can be found at FSP_BASE + FSP_INFO_HEADER_BASE + 0x38

Pre and Post Intel® Firmware Support Package (Intel® FSP) Function Invocation

- Host Boot Loader must be in 32-bit flat mode.
- Both the code and data selectors should have 4GB access range (by default)
- Interrupts should be off (by default)
- All Intel FSP API return an unsigned 32-bit integer as status.
- Intel FSP will reserve a region of memory (1MB for current implementation) for its own use throughout the operation.

Intel® Firmware Support Package (Intel® FSP) Programming Interfaces

3 APIs

- **TempRamInit**:

```
typedef FSP_STATUS
(FSPAPI *FSP_TEMP_RAM_INIT) (
  IN  TEMP_RAM_INIT_PARAMS     *TempRamInitParam
);
```

  Enables cache for being used as temporary memory and code caching

- **FspInit**

```
typedef FSP_STATUS
(FSPAPI *FSP_FSP_INIT) (
  IN  FSP_INIT_PARAMS         *FspInitParam
);
```

  Performs the processor and chipset initialization

- **NotifyPhase**

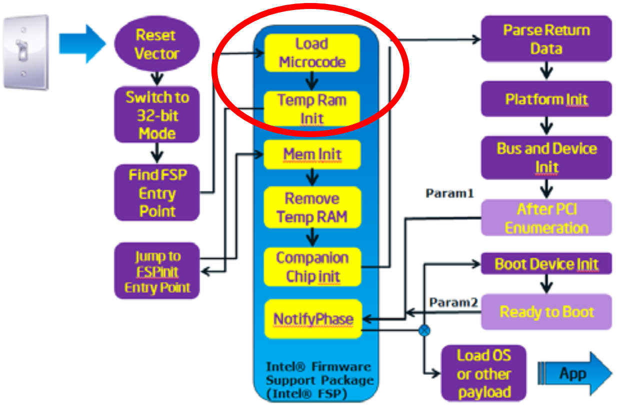
```
typedef  FSP_STATUS
(FSPAPI *FSP_NOTIFY_PHASE) (
  IN  NOTIFY_PHASE_PARAMS *NotifyPhaseParams
);
```

All APIs return FSP_STATUS

0:	Success
Others:	Error code

Intel® Firmware Support Package (Intel® FSP) Function: TempRamInitEntry

- This function can be entered soon after coming out from reset.
- This is the place where microcode patches are loaded and temporary memory is set up prior to initialize the memory.
  - Jump to TempRamInitEntry (instead of call).



  - After this function is executed, stack is available for "C" style calling convention.
- Prior to "jump":
  - Host boot loader code loads ESP as a pointer to a ROM-based stack, which contains the input parameters and the return address.
  - If available, pass a pointer to MTRR (optional)
- The return value indicates success or "invalid input parameter".

Sample Code

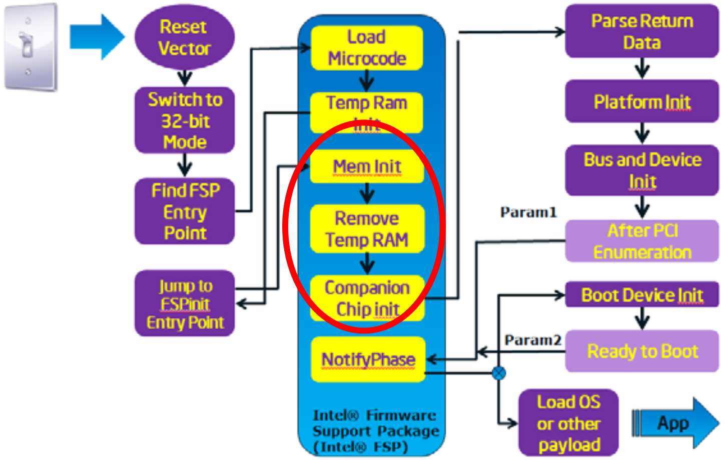
```
.equ  FSP_BIN_BASE,         0xFFF80000

.global basic_init
basic_init:
  < your code here >
  #
  # Parse the FV to find the FSP INFO Header
  #
  lea   findFspHeaderStack, %esp
  jmp   find_fsp_entry

findFspHeaderDone:
  mov   %eax,   %ebp  # put fsp header add in ebp
  mov   0x30(%ebp), %eax
  add   0x1c(%ebp), %eax

  lea   tempRamInitStack, %esp     # rom stack

  #
  # call FSP PEI to setup temporary Stack
  #
  jmp   *%eax

temp_RamInit_done:
  addl  $4, %esp

  <Your code here: call FspInitEntry>
```

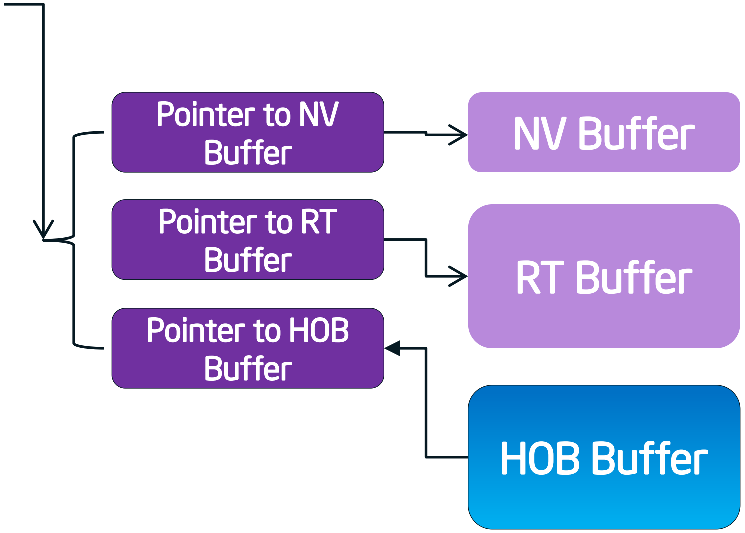
```
    < your code here >


    .align 4
  findFspHeaderStack:
    .long  findFspHeaderDone

  tempRamInitStack:
    .long  temp_RamInit_done
                         # ret addr
    .long  0x00000000
                # pointer to parameters
```

Hard-coded Entry-Point Address

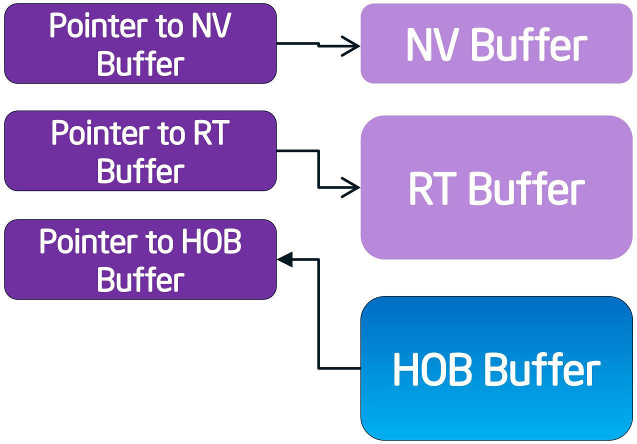
Intel® Firmware Support Package (Intel® FSP) Function: FspInitEntry

- This function is called after TempRamInitEntry.
- This is the place where Intel® Firmware Support Package (Intel® FSP) initializes memory, CPU, and companion chips.
- This function expects a pointer from the host boot loader so that Intel FSP can populate the Intel FSP Init data structures.
- These data structures are platform dependent and will be documented with each Intel FSP release.



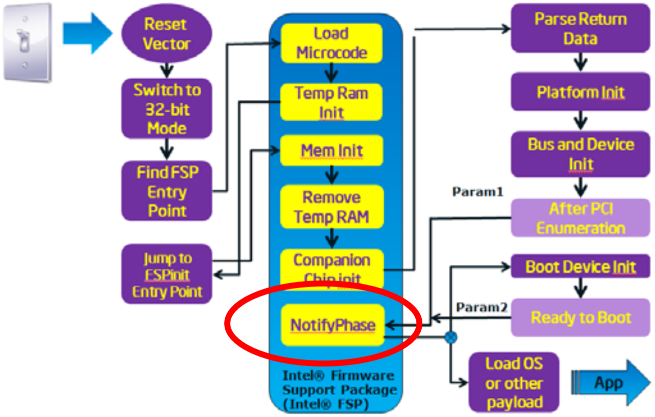


Intel® Firmware Support Package (Intel® FSP) Function: FspInitEntry Data Structures



- NVBuffer pointer:
  - The Intel® Firmware Support Package (Intel® FSP) returns a set of parameters that are needed to initialize the memory during S3 resume through the HOB in the boot phase.
  - The Boot Loader stores this data in an NVRAM
  - During S3, if the Boot Loader detects an S3 resume, it initializes the NV Buffer pointer in the Intel FSP Init data structure to point to this data in NVRAM.
  - The data can also be used to bypass some init code in order to save boot time.
- RTBuffer (in memory): Platform specific input data needed to initialize the silicon during each boot.
- HOB (Hand-Off data Block) pointer, filled in by Intel FSP: data consumed by the host boot loader, but the region is reserved by FSP.

Intel® Firmware Support Package (Intel® FSP) Function: NotifyPhaseEntry



- This Intel® Firmware Support Package (Intel® FSP) function is used by the host boot loader to notify the Intel FSP about the different phases in the boot process.
- This allows the Intel FSP to take appropriate actions as needed during different initialization phases.
- The phases will be platform dependent and will be documented with the Intel FSP release.
- There are currently two phases defined for the host boot loader to call back to Intel FSP: “post PCI enumeration” and “ready to boot”.

Callback Notifications

EnumInitPhaseAfterPciEnumeration

This callback is used when the host boot loader completes the PCI enumeration and the resource allocation for the PCI devices. Intel® Firmware Support Package (Intel® FSP) will use it to do some specific initialization for processor and chipset that requires PCI resource assignment.

EnumInitPhaseReadyToBoot

This callback is used just before the boot loader hands off to the OS loader. Intel FSP will use it to do some specific initialization for processor and chipset that is required before control is transferred to the OS.

Sample Code

```
#define FSP_BIN_BASE         0xFFF80000
  #define FSP_HEADER_OFFSET    0x94
  #define FSP_NOTIFY_FSPI_ENTRY 0x38

  #define FSP_NOTIFY_FSPI_PTR  (FSP_BIN_BASE + FSP_HEADER_OFFSET + FSP_NOTIFY_FSPI_ENTRY)

  typedef enum {
    EnumInitPhaseAfterPciEnumeration,
    EnumInitPhaseReadyToBoot
  } FSP_INIT_PHASE;

  typedef struct {
    FSP_INIT_PHASE    Phase;
  } NOTIFY_PHASE_PARAMS;
  typedef FSP_STATUS (FSPAPI *FSP_NOTFY_PHASE) (NOTIFY_PHASE_PARAMS
*NotifyPhaseParamPtr);

  void boot_loader_post_pci_notify_fn ()
  {
    FSP_NOTFY_PHASE         FspNotifyPhase;
    NOTIFY_PHASE_PARAMS     NotifyPhaseParams;

    NotifyPhaseParams.Phase = EnumInitPhaseAfterPciEnumeration;

    /* call FSP to Notify PostPciEnumeration */
    FspNotifyPhase = (FSP_NOTFY_PHASE)(*(uint32_t *)(FSP_NOTIFY_FSPI_PTR));
    FspNotifyPhase (&NotifyPhaseParams);

  }
```

Handle Return Data

- Intel® Firmware Support Package (Intel® FSP) builds a set of data structures before passing them back to the host boot loader.
- The data structures comply with UEFI Hand-Off-Block (HOB) format described in Volume 3: Shared Architectural Elements specification
- It is up to the host boot loader developers to decide how to consume the data in HOB because some data may be irrelevant to some boot loaders.
- As an example: GetMemorySize has the following code:

```
*LowMemoryLength += (UINT32) (Hob.ResourceDescriptor
-> ResourceLength);
```

Sample code in parsing HOB data

```
void
GetMemorySize (
  uint32_t         *LowMemoryLength,
  void             *HobBufferPtr
  )
{
  EFI_PEI_HOB_POINTERS    Hob;
  *LowMemoryLength = 0x100000;
  //
  // Get the HOB list for processing
  //
  Hob.Raw = HobBufferPtr;
  //
  // Collect memory ranges
  //
  while (!END_OF_HOB_LIST (Hob)) {
    if (Hob.Header->HobType == EFI_HOB_TYPE_RESOURCE_DESCRIPTOR) {
      if (Hob.ResourceDescriptor->ResourceType == EFI_RESOURCE_SYSTEM_MEMORY) {
        //
        // Need memory above 1MB to be collected here
        //
        if (Hob.ResourceDescriptor->PhysicalStart >= 0x100000 &&
            Hob.ResourceDescriptor->PhysicalStart < (EFI_PHYSICAL_ADDRESS) 0x100000000) {
          *LowMemoryLength += (uint32_t) (Hob.ResourceDescriptor->ResourceLength);
        }
      }
    }
    Hob.Raw = GET_NEXT_HOB (Hob);
  }
  return;
}
```

Customize Intel® Firmware Support Package (Intel® FSP) Internal Configurations

- Intel® Firmware Support Package (Intel® FSP) contains a configurable data region
  - Used by Intel FSP during initialization
  - Can be statically customized using a separate tool
- Binary Configuration Tool (BCT) will be provided with the Intel FSP binary to customize Intel FSP configuration region.
- More information will be provided when the first release of Intel FSP is available.

Position Intel® Firmware Support Package (Intel® FSP) in Your ROM

- Intel® Firmware Support Package (Intel® FSP) is not Position Independent Code, and the whole binary has to be rebased when it is relocated from the default address.

- Currently, Intel FSP default location is at the physical 0xFFF80000, which may change, and will be documented in every Intel FSP release.

- The location can also be changed with a rebase tool, which will also be released with the 1st official release of Intel FSP.

Other Considerations

- SMM:
  - SMM code will be provided only if there is a silicon workaround that necessitates it.
  - It’s the responsibility of the boot loader to provide the necessary infrastructure in terms of rendezvousing all the processors and calling the SMM code provided by Intel® Firmware Support Package (Intel® FSP) in SMM mode.
  - Details about the SMM handler may be platform/silicon dependent

- S3 Resume:
  - The host boot loader should detect the boot mode and if it is in S3 Resume, initialize the NVS Buffer pointer to enable Intel FSP to bring the memory out of self refresh.

- Fast boot option:
  - Intel FSP will check a flag in the RT_Buffer structure of the FspInit API to decide if the memory training can be skipped.

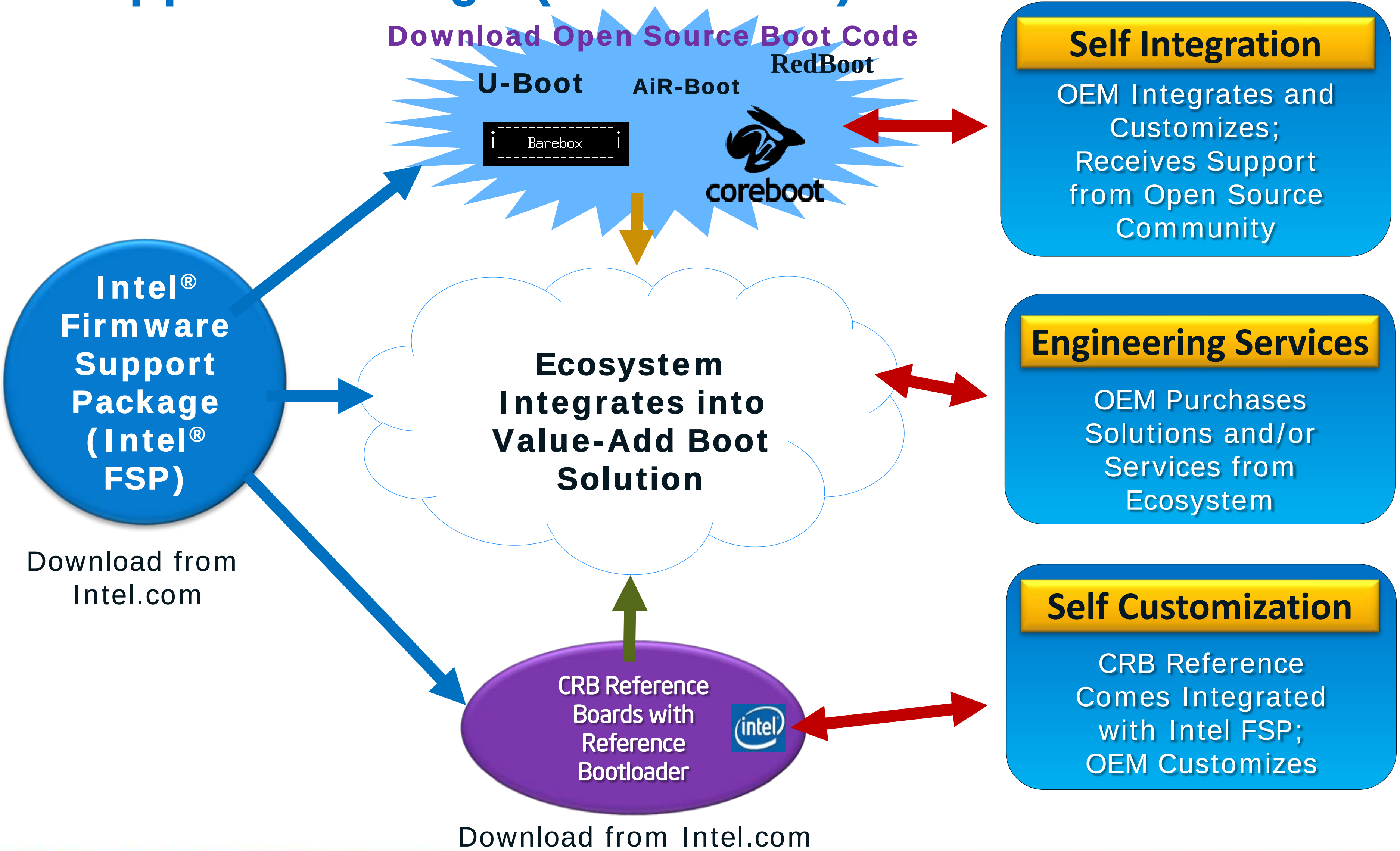
Other Considerations – Continued…

- Power Management:
  - Intel® Firmware Support Package (Intel® FSP) does not provide power management functions besides making power management features available to the host boot loader.
  - ACPI is an independent component of boot loader.
- Bus enumeration:
  - Intel FSP will initialize the CPU and the companion chips to a stage that all bus topology can be discovered by the host boot loader
- Security:
  - Intel FSP locks registers that Intel documents specify. Other system-level security features need to be done by the host boot loader.
- 64-bit Long Mode:
  - Intel FSP operates in 32-bit mode; it is the responsibility of the host boot loader to transition to 64-bit Long Mode if it is desired.

Agenda

- Introduction
- Philosophy and Principles
- Programming Interface and Data Structure
- **Development Options**
- Summary

Development Options with Intel® Firmware Support Package (Intel® FSP)



Intel® Firmware Support Package (Intel® FSP)

Download from Intel.com

Download Open Source Boot Code

U-Boot

AiR-Boot

RedBoot

Ecosystem Integrates into Value-Add Boot Solution

CRB Reference Boards with Reference Bootloader

Download from Intel.com

Self Integration

OEM Integrates and Customizes; Receives Support from Open Source Community

Engineering Services

OEM Purchases Solutions and/or Services from Ecosystem

Self Customization

CRB Reference Comes Integrated with Intel FSP; OEM Customizes

Agenda

- Introduction
- Philosophy and Principles
- Programming Interface and Data Structure
- Development Options
- **Summary**

Summary and Call For Actions

- Intel ISG's customers are versatile in product design and fast in execution. Many of them divert from a traditional PC-architecture.

- Intel® Firmware Support Package (Intel® FSP) is a flexible and scalable firmware solution offered to you to be integrated into a boot loader of choice.

- Intel FSP fellow-travelers are aligned and ready to serve you. We welcome more fellow-travelers to join us.

- Visit Intel FSP web site to download the documents for more details (http://www.intel.com/fsp)

- Check with your sales representative to get the NDA presentation and roadmap.

Useful Links

- UEFI specifications be downloaded from http://www.uefi.org/specs/
- Intel® Firmware Support Package web site:

  http://www.intel.com/fsp

Intelligent Systems

Contact Info of Presenter:

Jiming.sun@intel.com

1-408-765-0927

Intel® Intelligent Systems Summit

Intel® Intelligent Systems: A new era in embedded computing intel Intelligent Systems

THANK YOU

Intel® Intelligent Systems Summit

Intel® Intelligent Systems: A new era in embedded computing