tianocore

UEFI & EDK II TRAINING

EDK II Porting to a New Open Board Platform

tianocore.org

LESSON OBJECTIVE

- Define the porting task list for porting existing open platforms in EDK II
- Determine the necessary porting for each stage and boot phase of a new EDK II open platform project

Reference: Minimum Platform Architecture Specification

APPROACH - PORTING EDK II

Find Similar Open Board Projects

Staged Approach by Features

tianocore

Porting Task List

1. Get the EDK II packages to a local workspace
2. Select the Ref and correct Intel® FSP Package
3. Copy a reference `OpenBoardPkg/BoardXxx`
4. Use feature stages to port all required project modules
5. Validate each stage test point results defined with each stage

Porting Task List

1. **Get the EDK II packages to a local workspace**
2. **Select the Ref correct Intel® FSP Package**
3. **Copy a reference `OpenBoardPkg/BoardXxx`**
4. Use feature stages to port all required project modules
5. Validate each stage test point results defined with each stage

Analysis OpenBoard Reference Platforms

Steps 1 – 3

- Find a similar OpenBoard EDK II Platform in Github `edk2_platforms`
- Get the reference OpenBoard Pkg EDK II Platform from Github
- Build the chosen reference OpenBoard Pkg EDK II Platform
- Study the reference Build directory, OpenBoard .FDF and .DSC files
- Copy a reference board directory to a new name (i.e `<Generation>OpenBoardPkg/NewBoardXxx` where string "`New`" is meaningful to the project.)
- if it is a NEW Open Platform, Copy a reference `<Generation>OpenBoardPkg/BoardXxx` to a new name (e.g. `NewOpenBoardPkg/NewBoardX`)

Get the Reference OpenBoard Source

Follow the Build and Download instructions for building the reference open board
Download below repository to this WORKSPACE:

edk2 repository and Openssl

```
git clone --recursive https://github.com/tianocore/edk2.git
```

edk2-platforms repository

```
git clone https://github.com/tianocore/edk2-platforms.git
```

edk2-non-osi repository

```
git clone https://github.com/tianocore/edk2-non-osi.git
```

FSP repository

```
git clone https://github.com/IntelFsp/FSP.git
```

Build the Reference OpenBoard

Open a Command Window and CD to the workspace directory
For Linux - CD to the edk2 to run the “edksetup.sh” script

```
bash$ cd edk2
bash$ source edksetup.sh
bash$ cd ../
```

CD to the Intel directory in edk2-platforms

```
$> cd edk2-platforms/Platform/Intel
```

Run the python build script with the OpenBoard Name (Kabylake is used)

```
$> python build_bios.py –p KabyLakeRvp3
```

Use the “Build” Directory as a Reference

After the Build is completed the Build directory will be a mirror of all the sources used to build the reference OpenBoard.

This can serve as a cross reference to determine what sources are used in the chosen reference OpenBoard.

- Library references for the library class implementation instance
- Report file `BuildReport.log` for PCDs and LIBRARY instances

```
<workspace>
  Build /KabylakeOpenBoardPkg /KabylakeRvp3 /DEBUG_<BuildTag>/
     FV /
     IA32 /
        <Dirs built for SEC and PEI> /
     X64 /
         <Dirs built for DXE – BDS – Boot> /
    BuildReport.log < report file for PCDs, Libraries, Build flags, …>
```

Study the OpenBoard .DSC and .FDF

Porting requires becoming familiar with the chosen reference platforms DSC and FDF files.

DSC

DSC will point to the correct Libraries used in the reference platform

FDF

FDF will describe the Flash layout and FVs used for the different stages of the boot flow

DSC Files for KabyLakeRvp3

```
/edk2-platforms/Platform/
  Intel/KabylakeOpenBoardPkg/
   KabylakeRvp3
     OpenBoardPkg.dsc
     OpenBoardPkgConfig.dsc
     OpenBoardPkgPcd.dsc
     OpenBoardPkgBuildOption.dsc
/edk2-platforms/Platform/
  Intel/MinPlatformPkg/
    Include/Dsc/
      CoreCommonLib.dsc
      CorePeiLib.dsc
      CoreDxeLib.dsc
      CorePeiInclude.dsc
      CoreDxeInclude.dsc
```

```
/edk2-platforms/Silicon/
  Intel/KabylakeSiliconPkg/
    SiPkgCommonLib.dsc
    SiPkgPeiLib.dsc
    SiPkgDxeLib.dsc
    SiPkgPei.dsc
    SiPkgDxe.dsc
    SiPkgBuildOption.dsc
```

`OpenBoardPkg.dsc` – Includes all of the Platform, common core and silicon related .dsc files

Example: DSC File

```
[Defines]
  #
  # Set platform specific package/folder name, same as passed from PREBUILD script.
  # PLATFORM_PACKAGE would be the same as PLATFORM_NAME as well as package build folder
  # DEFINE only takes effect at R9 DSC and FDF.
  #
  DEFINE      PLATFORM_PACKAGE                = MinPlatformPkg
  DEFINE      PLATFORM_SI_PACKAGE             = KabylakeSiliconPkg
  DEFINE      PLATFORM_SI_BIN_PACKAGE         = KabylakeSiliconBinPkg
  DEFINE      PLATFORM_FSP_BIN_PACKAGE        = AmberLakeFspBinPkg
  DEFINE      PLATFORM_BOARD_PACKAGE          = KabylakeOpenBoardPkg
  DEFINE      BOARD                           = KabylakeRvp3
  DEFINE      PROJECT                         = $(PLATFORM_BOARD_PACKAGE)/$(BOARD)

  #
  # Platform On/Off features are defined here
  #
  !include OpenBoardPkgConfig.dsc
  !include OpenBoardPkgPcd.dsc

[Defines]
!if gIntelFsp2WrapperTokenSpaceGuid.PcdFspModeSelection == 1
  #
  # For backward compatibility API mode will use KabylakeFspBinPkg.
  # KabylakeFspBinPkg only supports API mode.
```

Link to DSC File

Link to EDK II DSC Spec.

FDF Files for KabyLakeRvp3

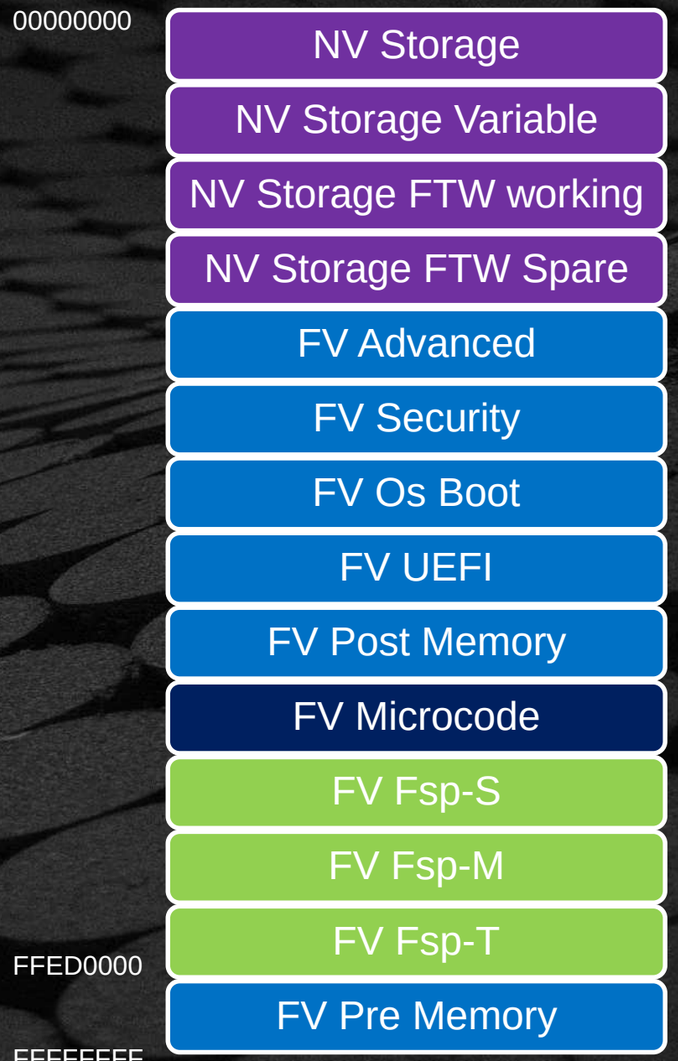
```
/edk2-platforms/Platform/
  Intel/KabylakeOpenBoardPkg/
   Include/Fdf/
       FlashMapInclude.fdf
   KabylakeRvp3
    OpenBoardPkg.fdf
```

OpenBoardPkg.fdf – Includes all of the Platform and common core related .fdf files

```
/edk2-platforms/Platform/
  Intel/MinPlatformPkg/
  Include/Fdf/
     CorePreMemoryInclude.fdf
     CorePostMemoryInclude.fdf
     CoreUefiBootInclude.fdf
     CoreOsBootInclude.fdf
     CoreSecurityPreMemoryInclude.fdf
     CoreSecurityPostMemoryInclude.fdf
     CoreSecurityLateInclude.fdf
     RuleInclude.fdf
```

Flash Layout for Kabylake

```
edk2-platforms/Platform/
  Intel/
   KabylakeOpenBoardPkg/
    Include/
     Fdf/
      FlashMapInclude.fdf
```



Example: FDF File

```
[FD.KabylakeRvp3]
#
# FD Tokens, BaseAddress, Size, ErasePolarity, BlockSize, and NumBlocks, cannot be
# assigned with PCD values. Instead, it uses the definitions for its variety, which
# are FLASH_BASE, FLASH_SIZE, FLASH_BLOCK_SIZE and FLASH_NUM_BLOCKS.
#
BaseAddress   = $(FLASH_BASE) | gSiPkgTokenSpaceGuid.PcdFlashAreaBaseAddress      #The base address of the FLASH
Device.
Size          = $(FLASH_SIZE) | gSiPkgTokenSpaceGuid.PcdFlashAreaSize            #The size in bytes of the FLASH
Device
ErasePolarity = 1
BlockSize     = $(FLASH_BLOCK_SIZE)
NumBlocks     = $(FLASH_NUM_BLOCKS)

DEFINE SIPKG_DXE_SMM_BIN  = INF
DEFINE SIPKG_PEI_BIN      = INF

# Set FLASH_REGION_FV_RECOVERY_OFFSET to PcdNemCodeCacheBase, because macro expression is not supported.
# So, PlatformSecLib uses PcdFlashAreaBaseAddress + PcdNemCodeCacheBase to get the real CodeCache base address.
SET gSiPkgTokenSpaceGuid.PcdNemCodeCacheBase = $(gMinPlatformPkgTokenSpaceGuid.PcdFlashFvPreMemoryOffset)
SET gSiPkgTokenSpaceGuid.PcdFlashMicrocodeFvBase = $(gSiPkgTokenSpaceGuid.PcdFlashAreaBaseAddress) +
$(gSiPkgTokenSpaceGuid.PcdFlashMicrocodeFvOffset)
SET gSiPkgTokenSpaceGuid.PcdFlashMicrocodeFvSize = $(gSiPkgTokenSpaceGuid.PcdFlashMicrocodeFvSize)
SET gUefiCpuPkgTokenSpaceGuid.PcdCpuMicrocodePatchAddress = $(gSiPkgTokenSpaceGuid.PcdFlashMicrocodeFvBase) + 0x60
SET gUefiCpuPkgTokenSpaceGuid.PcdCpuMicrocodePatchRegionSize = $(gSiPkgTokenSpaceGuid.PcdFlashMicrocodeFvSize) - 0x60
```

Link to FDF File

Link to EDK II FDF Spec

Copy a Similar Reference OpenBoard

```
MyWorkSpace/
      edk2/
        - "edk2 Common"
      edk2-platforms/
        Platform/ "Platform"
           Intel/
             MinPlatformPkg/ "Platform common"
             KabylakeOpenBoardPkg
             NewOpenBoardPkg
                  BoardXxx/ "Board"
        Silicon/
           Intel/
             KabyLakeSilconPkg/"Silicon"
      edk2-non-osi/
        Silicon/
           Intel/
      FSP/"Silicon"
```

1. Get the EDK II packages locally to the workspace
2. Select the Ref OpenBoard and correct Intel® FSP silicon initialization solution
3. Copy a reference open platform board `<Generation>OpenBoardPkg/BoardXxx` to a new directory –

   Copy `<Generation>OpenBoardPkg/BoardXxx`

   to `NewOpenBoardPkg/BoardXxx`

Only make changes in the "`NewOpenBoardPkg`"

Copy a New Board Only

```
MyWorkSpace/
      edk2/
        - "edk2 Common"
      edk2-platforms/
        Platform/ "Platform"
           Intel/
             MinPlatformPkg/
             KabylakeOpenBoardPkg
             <Generation>OpenBoardPkg
                  BoardXxx/
                  NewBoardXxx/ "Board"
        Silicon/
           Intel/
             KabyLakeSilconPkg/"Silicon"
      edk2-non-osi/
        Silicon/
           Intel/
```

1. Get the EDK II packages locally to the workspace
2. Select the Ref OpenBoard and correct Intel® FSP silicon initialization solution
3. Copy a reference board <Generation>OpenBoardPkg/BoardXxx to a new directory –

   Copy `BoardXxx` to `NewBoardXxx`

Only make changes in the "`NewBoardXxx`"

PI Modules – One board, one dir

```
KabylakeOpenBoardPkg
  Acpi
  FspWrapper
    Library
      PeiFspPolicyUpdateLib
  Include
  Library
  KabylakeRvp3   <-
   . . .
  NewBoardXxx    <-
    Include
    Library
       . . .
    OpenBoardPkg.dsc
    OpenBoardPkg.fdf
```

```
NewBoardXxx
  Library
    BoardInitLib
      PeiBoardInitPreMemLib
      PeiBoardInitPostMemLib


    BasePlatformHookLib

    BoardAcpiLib
```

If a New OpenBoard Package

```
NewOpenBoardPkg
  Acpi
  FspWrapper
    Library
      PeiFspPolicyUpdateLib
  Include
  Library
  BoardXxx
    Include
    Library
      . . .
    OpenBoardPkg.dsc
    OpenBoardPkg.fdf
```

```
BoardXxx
  Library
    BoardInitLib
      PeiBoardInitPreMemLib
      PeiBoardInitPostMemLib


    BasePlatformHookLib

    BoardAcpiLib
```

Porting Task List

1. Get the EDK II packages to a local workspace
2. Select the Ref and correct Intel® FSP Package
3. Copy a reference `OpenBoardPkg/BoardXxx`
4. Use feature stages to port all required project modules
5. Validate each stage test point results defined with each stage

PORT BY STAGES

Use the staged architecture as a porting guide

Staged Approach by Features

- Platform Firmware Boot Stage PCD

PCD Variable:

`gPlatformModuleTokenSpaceGuid.PcdBootStage`

Stage 1	enable debug
Stage 2	memory initialization
Stage 3	boot to UEFI shell only
Stage 4	boot to OS
Stage 5	boot to OS w/ security enabled
Stage 6	Advanced Feature Selection
Stage 7	Performance Optimizations

PCD Is tested within .FDF to see which modules to include

Staged Approach by Features

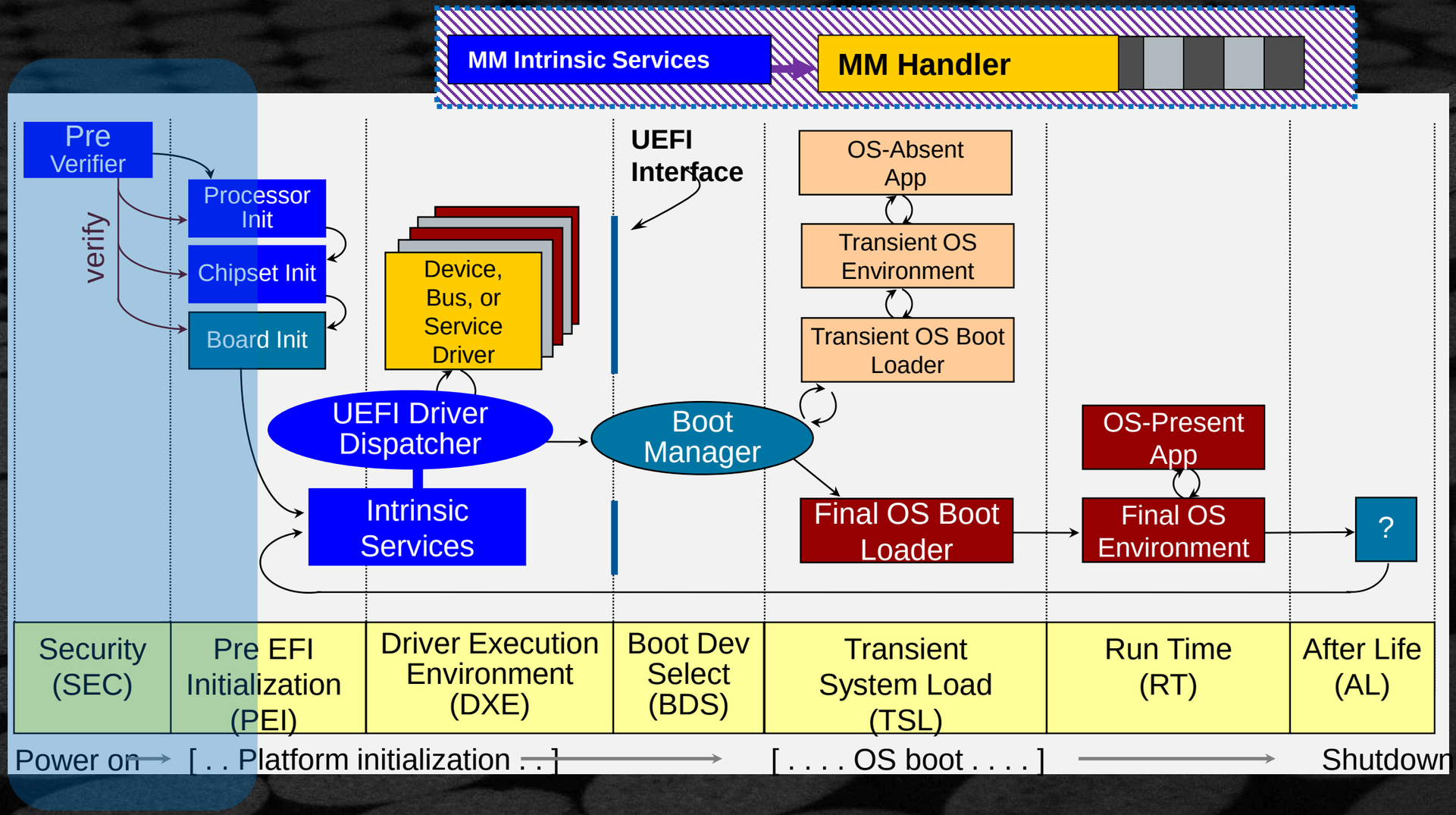
- Platform Firmware Boot Stage PCD

Stage 1	enable debug
Stage 2	memory initialization
Stage 3	boot to UEFI shell only
Stage 4	boot to OS
Stage 5	boot to OS w/ security enabled
Stage 6	Advanced Feature Selection
Stage 7	Performance Optimizations

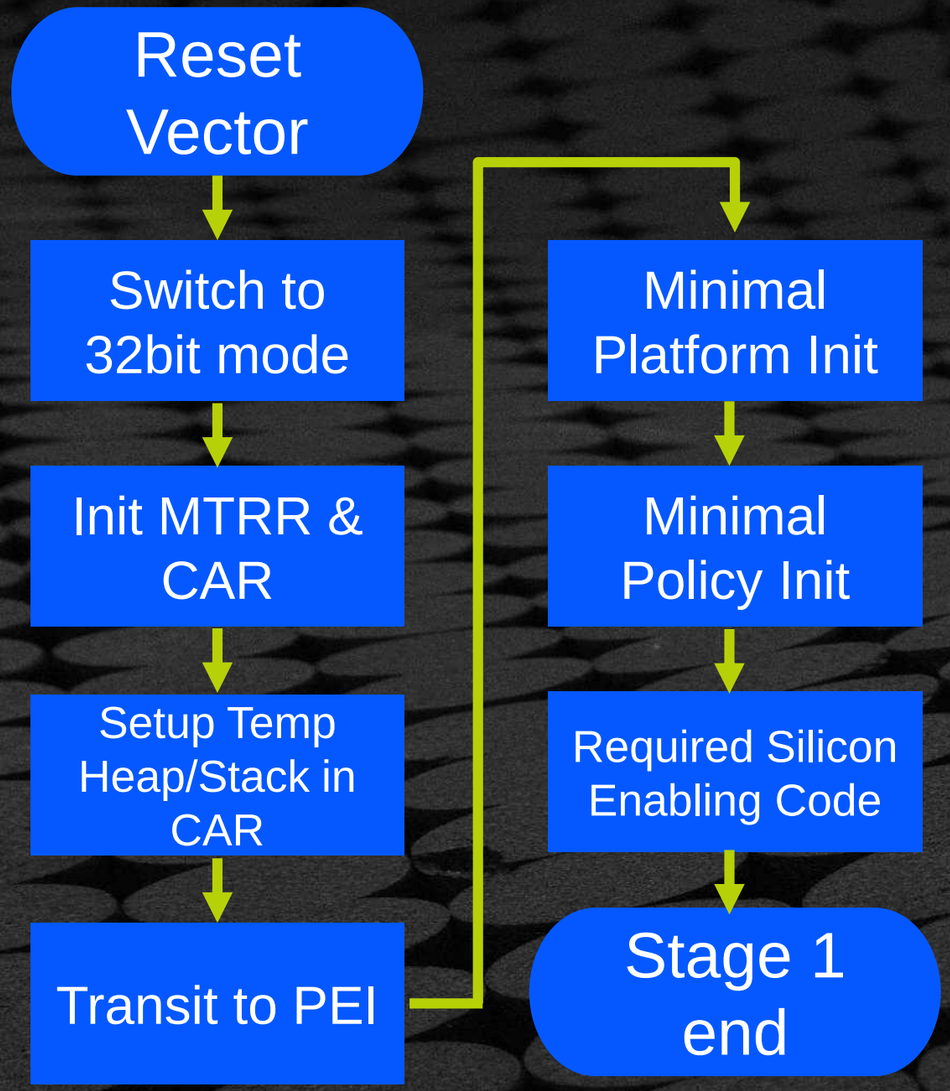
PCD Is tested within .FDF to see which modules to include

Boot Flow – Stage 1

Stage 1



High Level Control Flow – Stage 1



Major Execution Activities

- Establish temporary memory (initialize MTRR)
- Perform pre-memory board-specific initialization
- Board detection
- GPIOs
- Serial Port initialization (Example: SIO, HSUART)

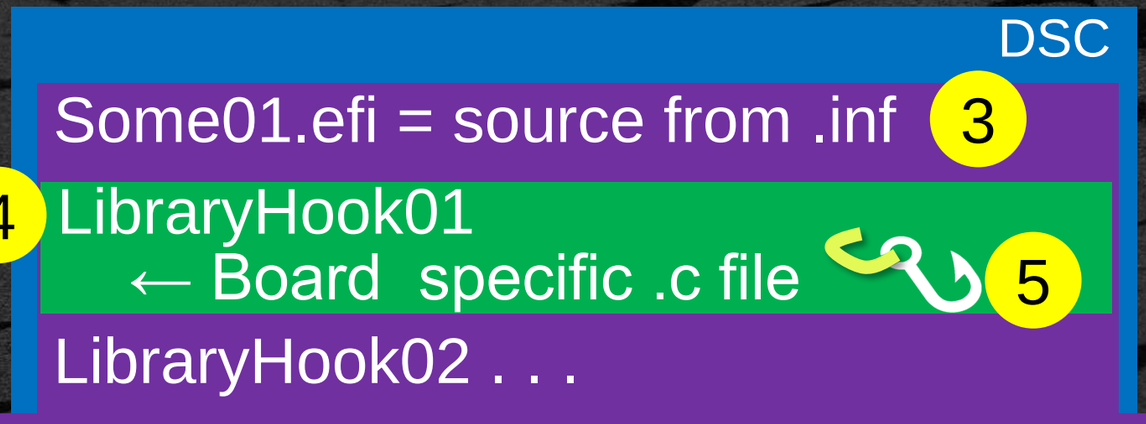
Process of Porting

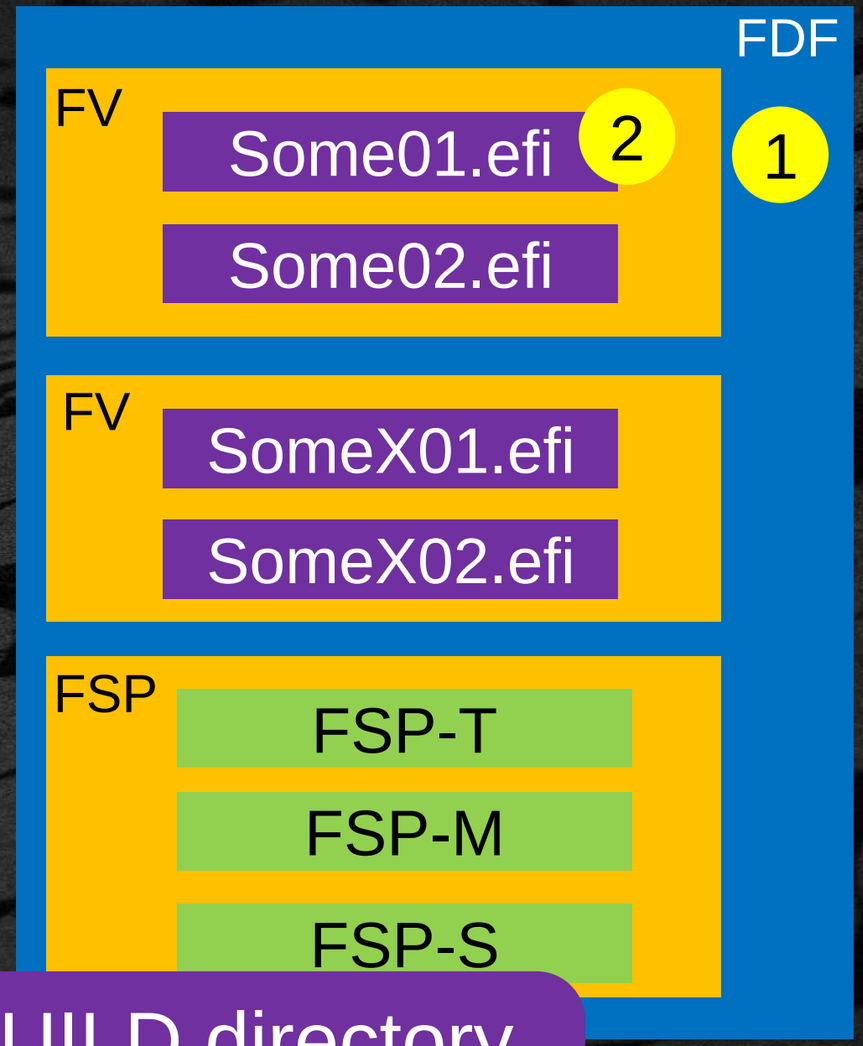
Check the Board/Platform .FDF file layout

Investigate the FDF then DSC files

Porting process per stage find and update platform hooks

1. Locate FVs for each stage
2. Modules for each FV contents
3. Module Locations
4. Platform Porting Libraries per Module
5. Update the Hook Function for Board





Also check the reference platform BUILD directory

tianocore

Stage 1 Firmware Volumes

1

Name	Content
FvPreMemory	SEC + StatusCode
FvBspPreMemory	Pre-memory board initialization
FvFspT	SEC silicon initialization –T-RAM
FvFspM	Memory initialization
-> FvPreMemorySilicon	Pre-memory silicon initialization
FvFspS	Silicon initialization
-> FvPostMemorySilicon	Post-memory silicon initialization

tianocore

Stage 1 FVs Contents

FV	Components	Purpose
FvPreMemory	SecCore.efi	• Reset Vector • Passes PEI core the address of FvFspM • Passes PEI core the debug configuration
	ReportFvPei.efi	Installs firmware volumes
	PlatformInitPreMemory.efi	Performs pre memory initialization
	SiliconPolicyPeiPreMemory.efi	Publishes silicon initialization configuration
FvFspT	TempRamInit API	Set up Temp Memory (CAR)
FvFspM	FspmWrapperPeim.efi	call FspMemoryInit API

2

Mapped according to .FDF file layout

tianocore

Find Stage 1 Modules
- Example SecCore.efi

3

Producing Package – UefiCpuPkg

Libraries Consumed – PlatformSecLib, SerialPortLib

Library	API definition	Producing Pkg	Description
PlatformSecLib	UefiCpuPkg	MinPlatformPkg	Reset vector and SEC initialization code.
SerialPortLib	MdeModulePkg	Board<X>Pkg	SIO vendor specific initialization to enable serial port.

Stage 1 Platform Libraries

Item	API Definition Package	Producing Package	Description
`BoardInitLib`	MinPlatformPkg	BoardPkg	Board initialization library.
`ReportFvLib`	MinPlatformPkg	MinPlatformPkg	Installs platform firmware volumes.
`SerialPortLib`	MdeModulePkg	BoardPkg	SIO vendor specific initialization to enable serial port.
`SiliconPolicyInitLib`	IntelSiliconPkg	SiliconPkg	Provides default silicon configuration policy data.
`SiliconPolicyUpdateLib`	IntelSiliconPkg	BoardPkg	Provides board updates to silicon configuration policy data.
`PlatformSecLib`	UefiCpuPkg	MinPlatformPkg	Reset vector and SEC initialization code.

tianocore

How to search for Libraries in the Workspace

4

1. Search the workspace .DSC files for the string of the library
2. Open the .DSC files associated with the open board platform project
3. Determine which Library is used and that should have the build path in the workspace
4. DSC file will have similar to:

```
SomeLib|Path_to_the_Library_used.inf
```

5. Verify the instance used from the `Build` directory

Platform SEC Lib - Kabylake

Search the board platform .DSC file for where the `PlatformSecLib` is in the Platform-Board Tree

DSC FILE:

```
PlatformSecLib|MinPlatformPkg/FspWrapper/Library/\
  SecFspWrapperPlatformSecLib/SecFspWrapperPlatformSecLib.inf
```

Directory:

```
Library/SecFspWrapperPlatformSecLib/
    FsptCoreUpd.c
    FspWrapperPlatformSecLib.c
    Platforminint.c
    SecFspWrapperPlatformSecLib.inf
    SecGetPerformance.c
    SecPlatforminformation.c
    SecRamInitData.c
    SecTemRamDone.c
```

```
IA32/
  fsp.h
  PeiCoreEntry.nasm
  SecEntry.nasm
  Stack.nasm
```

_ModuleEntryPoint

Establish Temporary Memory

Call to FSP API

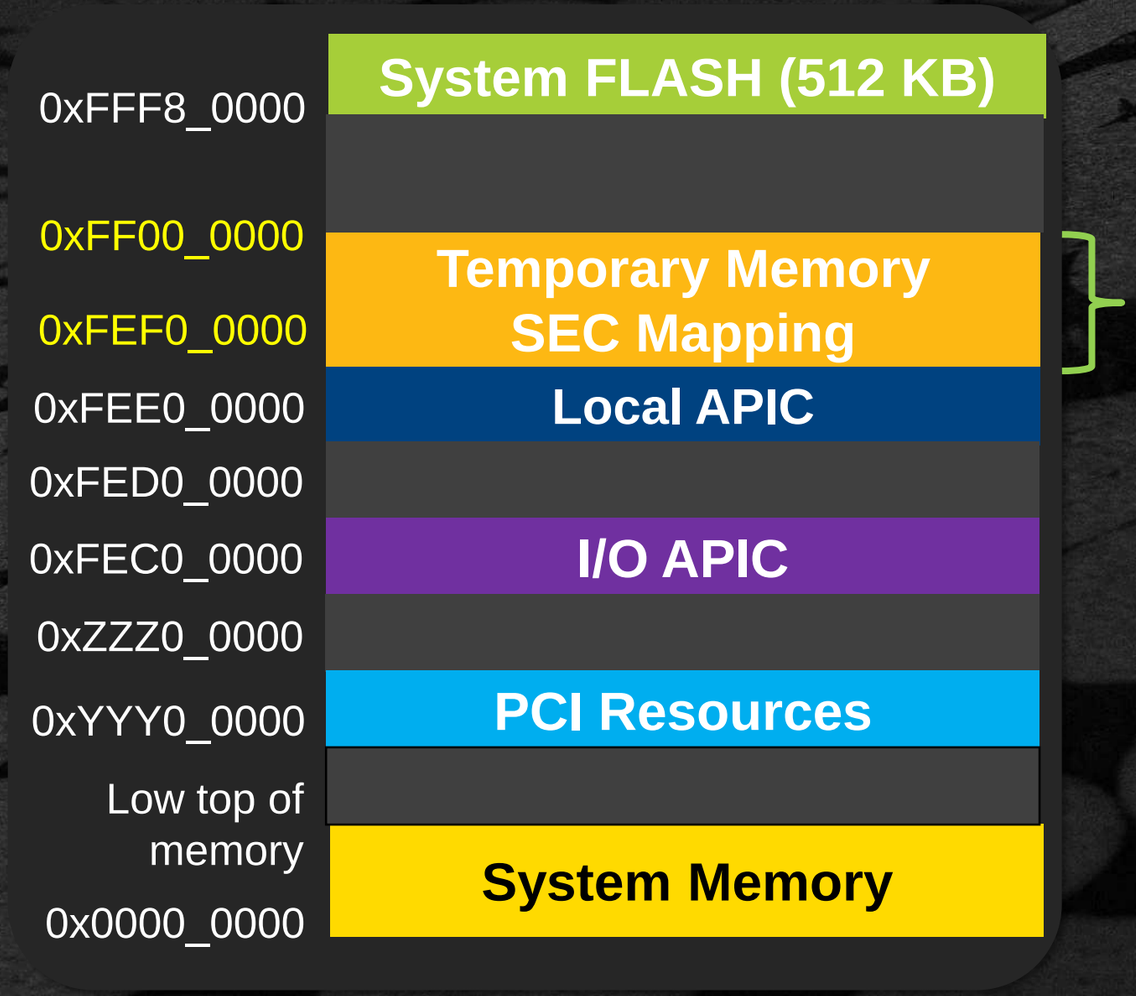
Call to `TempRamInit` API located in the FSP Binary module

FSP-T

`TempRamInit API`

`TempRamInit Api`

- Initializes T-RAM silicon functionality
  - No Evection mode / Cache As Ram
- Tests T-RAM functionality



Transition to PEI Entry Point

SecMain calls the entry point into PEI Core

```
Secmain(
{ . . .
  // Transfer the control to the PEI core
  ASSERT (PeiCoreEntryPoint != NULL);
  (*PeiCoreEntryPoint) (SecCoreData, PpiList);
  UNREACHABLE ();
```

`PeiCore` is the `PeiCoreEntryPoint` in the FvPreMemory Firmware Volume

`edk2/MdeModulePkg/Core/Pei/PeiMain PeiMain.c`

```
EFIAPI
PeiCore (
  IN CONST EFI_SEC_PEI_HAND_OFF        *SecCoreDataPtr,
  IN CONST EFI_PEI_PPI_DESCRIPTOR      *PpiList,
  IN VOID                              *Data
  )
{
```

Platform Initialization Board Hook Modules - Stage 1

```
MinPlatformPkg/
  Include/
    Library/
      BoardInitLib.h ←— // hooks
  Library/
  . . .
  PlatformInit/
    PlatformInitPei/
      PlatformInitPreMem/
```

```
BoardDetect()
BoardDebugInit()
BoardBootModeDetect()
BoardInitBeforeMemoryInit()
```

5

Platform folder `PlatformInit` controls the platform initialization flow

Link: BoardInitLib.h

tianocore

Hook - Board Detection

-Kabylake example

```
MinPlatformPkg/
  . . .
  PlatformInit/
    PlatformInitPei ->
      PlatformInitPreMem.c
        BoardDetect()
KabylakeOpenBoardPkg/
  . . .
  KabylakeRvp3/
    Library/
      BoardInitLib ->
        PeiBoardInitPreMemLib.c
          BoardDetect()
        PeiKabylakeRvp3Detect.c
          KabylakeRvp3BoardDetect()
```

Uses PCD Library calls to set / get Board SKU for Storing Board ID

`LibPcdGetSku()` & `LibPcdSetSku()`

`KabylakeRvp3BoardDetect()` function reads Board ID from embedded controller (EC) using the LPC bus

`LibPcdSetSku()` stores Board ID

`LibPcdGetSku()` used from that point on

Multi-SKU PCD – Board ID

DSC File – SKU Set at BUILD time

```
. . .
SKUID_IDENTIFIER = ?

[SkuIds]
0|DEFAULT
4|BoardX
0x42|BoardY

[PcdsDynamicDefault.common.BoardX]
gBoardModuleTokenSpaceGuid.PcdGpioPin|0x8
gBoardModuleTokenSpaceGuid.PcdGpioInitValue|\
        {0x00, 0x04, 0x02, 0x04, ...}

[PcdsDynamicDefault.common.BoardY]
gBoardModuleTokenSpaceGuid.PcdGpioPin|0x4
gBoardModuleTokenSpaceGuid.PcdGpioInitValue|\
        {0x00, 0x02, 0x01, 0x02, ...}
```

SKU PCD Set Dynamically

```
BoardXBoardDetect( VOID)
{
. . .
  if (LibPcdGetSku () != 0) {
    return EFI_SUCCESS;
  }
  if (IsBoardX ()) {
     LibPcdSetSku (BoardIdIsBoardX);
     ASSERT (LibPcdGetSku() ==
                BoardIdIsBoardX);
  }
  return EFI_SUCCESS;
}
```

PCD Determines Board ID at build time

Board Debug Initialization

```
Platform/Intel/KabylakeOpenBoardPkg/
 . . .
 KabylakeRvp3/
  Library/
    BoardInitLib/
     PeiKabylakeRvp3InitPreMemLib.c
       KabylakeRvp3DebugInit()
           EarlySiliconInit()
         . . .
       KabylakeRvp3BoardBootModeDetect()
          return BOOT_WITH_FULL_CONFIGURATION

Silicon/Intel/KabylakeSiliconPkg/
  Library/
    SiliconInitLib/
      SiliconInitPreMem.c
        EarlySiliconInit()
```

Once Board detection is successful, the next step is Board initialization.

Kabylake example calls `EarlySiliconInit`

- Early Platform PCH initialization

- Boot Mode Detect – example returns Boot with full configuration

Board Init Before Memory Init

```
Platform/Intel/KabylakeOpenBoardPkg/
 . . .
 KabylakeRvp3/
  Library/
   BoardInitLib /
     PeiKabylakeRvp3InitPreMemLib.c
       KabylakeRvp3BoardInitBeforeMemoryInit()
         KabylakeRvp3InitPreMem ()
         I2CGpioExpanderInitPreMem()
         GpioInitPreMem ()
         SioInit ()
         SiliconInit ()
 Silicon/Intel/KabylakeSiliconPkg/
   Library/
     SiliconInitLib/
       SiliconInitPreMem.c
         SiliconInit()
```

Board Initialization before Memory Initialization

The board specific initialization may include:

1. Invoke a set of PCD for policy initialization later.
2. Configure the hardware devices (such as GPIO, SIO)
3. Silicon Initialization – i.e. PCH

GPIOs – Stage 1

```
Platform/Intel/KabylakeOpenBoardPkg/
 . . .
 KabylakeRvp3/
  Library/
   BoardInitLib /
     PeiKabylakeRvp3InitPreMemLib.c
       KabylakeRvp3BoardInitBeforeMemoryInit()
         KabylakeRvp3InitPreMem ()
         I2CGpioExpanderInitPreMem()
         GpioInitPreMem ()
         SioInit ()
         SiliconInit ()
 Silicon/Intel/KabylakeSiliconPkg/
   Library/
     PeiDxeSmmGpioLib
       GpioInit.c
         GpioConfigureSklPch()
```

If GPIOs need to be Initialized pre-memory then the hook `GpioInitPreMem()` calls the silicon library for GPIOs

Example for Kabylake PCH GPIO pins

Table `KabylakeRvp3GpioTable.c`

Silicon Policy – Stage 1

```
Platform/Intel/MinPlatformPkg
  . . .
   PlatformInit/
     SiliconPolicyPei/
      SiliconPolicyPeiPreMem.c
        SiliconPolicyInitPreMem()

Silicon/Intel/KabylakeSiliconPkg/
     Library
       PeiSiliconPolicyInitLibFsp
          PeiFspPolicyInitLib.c
           SiliconPolicyInitPreMem()
```

FSP-M

Fsp Silicon Policy Update Pre Mem

Performs silicon pre-mem policy initialization.

The meaning of Policy is defined by silicon code.

Input Policy should be FspmUpd.

Value of FspmUpd has been initialized by FSP binary default value

Debug Configuration - Serial Port

Serial port parameters come from the board and are used for debug features, serial input/output devices supporting local or remote consoles, and OS level debuggers

Library – `SerialPortLib` find in workspace used by board .dsc

Serial port configuration options consumed by `SerialPortLib`

`SerialPortLib` is used by the `StatusCodeHandlerPei.inf` component to initialize and display messages to a serial port

Serial port configuration options are published via PCH_SERIAL_IO_CONFIG used by `PeiPchPolicyLib`

```
Silicon/Intel/KabylakeSiliconPkg/Pch/Library/PeiPchPolicyLib
```

Debug Configuration PCDs - Serial Port

gEfiMdeModulePkgTokenSpaceGuid.PcdSerialBaudRate - Baud rate for the 16550 serial port
gEfiMdeModulePkgTokenSpaceGuid.PcdSerialUseMmio - Enable serial port MMIO addressing
gEfiMdeModulePkgTokenSpaceGuid.PcdSerialUseHardwareFlowControl - Enable serial port HW flow control
gEfiMdeModulePkgTokenSpaceGuid.PcdSerialDetectCable - Enable blocking Tx if no cable
gEfiMdeModulePkgTokenSpaceGuid.PcdSerialRegisterBase - Register the serial port base address
gEfiMdeModulePkgTokenSpaceGuid.PcdSerialLineControl - Serial port line control configuration
gEfiMdeModulePkgTokenSpaceGuid.PcdSerialFifoControl - Serial port FIFO control
gMinPlatformPkgTokenSpaceGuid.PcdSecSerialPortDebugEnable - Enable serial port debug in SEC phase

Debug configuration PCDs

gEfiMdePkgTokenSpaceGuid.PcdFixedDebugPrintErrorLevel - Control optimization based on debug print level – message type
gEfiMdePkgTokenSpaceGuid.PcdDebugPropertyMask - Control DebugLib behavior
gEfiMdePkgTokenSpaceGuid.PcdDebugPrintErrorLevel - Control run time debug print level
gEfiMdePkgTokenSpaceGuid.PcdReportStatusCodePropertyMask - Control display of status codes

tianocore

Stage 1 Checklist

Steps to enable a board for Stage 1.

1. Copy the EDK II packages to a local workspace
2. Select the correct Intel® FSP & review requirements
3. Get silicon initialization policy for the given board.
4. Customize the silicon initialization policy to the board-specific requirements.
5. Determine other firmware and software components
6. Determine board-specific information required to fetch code and show debug output.
7. Copy a reference `GenerationOpenBoardPkg/BoardXxx` and update the board interfaces in Required Functions.
   - serial port initialization code in `PlatformHookSerialPortInitialize ()` at `BoardPkg/Library/BasePlatformHookLib`.
   - Add Board detection code in `BoardDetect ()`

EDK II Open Platform Spec Test points Stage 1

tianocore

Stage 1 Checklist - Board Pre-mem Lib

Example: Kabylake –

`KabylakeOpenBoardPkg/KabylakeRvp3/Library/BoardInitLib/PeiBoardInitPreMemLib`

`KabyLakeOpenBoardPkg/NewBoardXxx/Library/BoardInitLib/PeiBoardInitPreMemLib`

1. Add Board pre-memory debug initialization code in `BoardDebugInit ()`
2. Ensure all PCDs in the Configuration section are correct (i.e. Serial debug)
3. Verify values of other PCDs are correct
4. Verify the flash layout is compliant
5. Verify the required binaries included in the build
6. Verify debug output during the code execution path for Stage 1
7. Verify the test point results are correct

EDK II Open Platform Spec Test points Stage 1

Staged Approach by Features

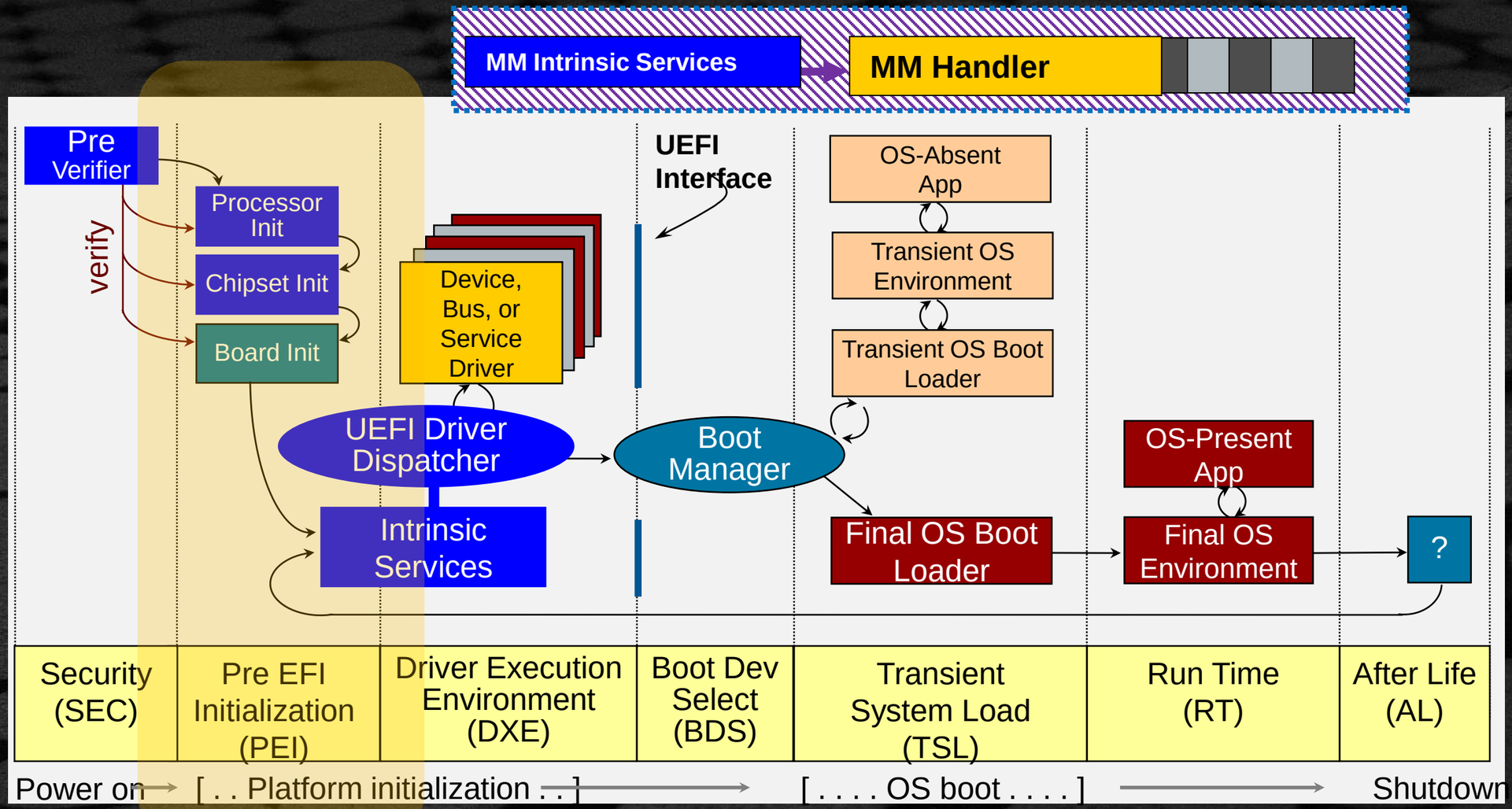
- Platform Firmware Boot Stage PCD

Stage 1	enable debug
Stage 2	memory initialization
Stage 3	boot to UEFI shell only
Stage 4	boot to OS
Stage 5	boot to OS w/ security enabled
Stage 6	Advanced Feature Selection
Stage 7	Performance Optimizations

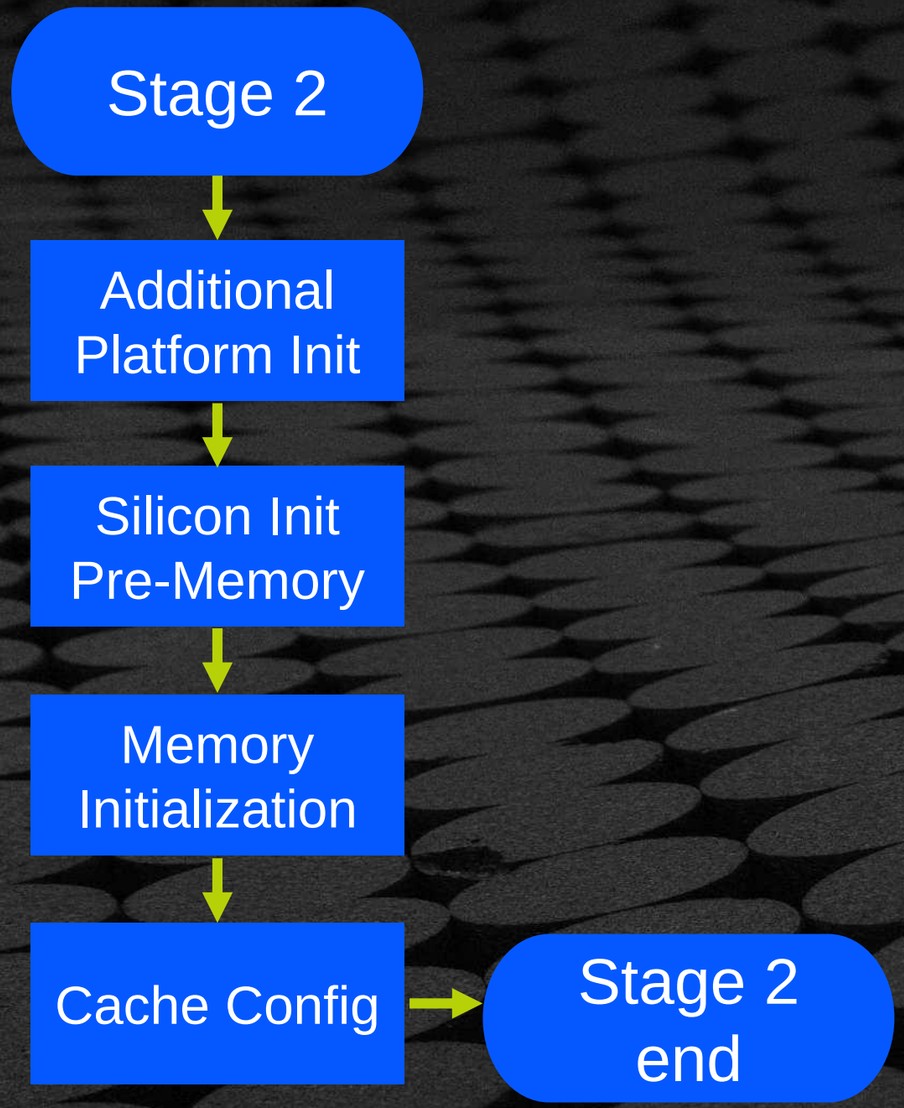
PCD Is tested within .FDF to see which modules to include

Boot Flow – Stage 2

Stage 2



High Level Control Flow – Stage 2



Major Execution Activities

- Complete execution of the memory initialization module
- Discover, train and install permanent memory
- Migrate the temporary memory/stack to permanent memory.
- Migrate any code modules from temporary RAM to permanent memory.
- Perform cache configuration for a post-memory environment.
- Execute memory installed notification actions.

tianocore

Stage 2 Firmware Volumes

Name	Content
FvPostMemory	Post-memory modules
FvFspM	Memory initialization
-> FvPreMemorySilicon	Pre-memory silicon initialization
FvFspS	Silicon initialization
-> FvPostMemorySilicon	Post-memory silicon initialization

tianocore

Stage 2 FVs Contents

FV	Components	Purpose
FvPostMemory	`PlatformInitPostMem.efi`	Performs post memory initialization
	`FspsWrapperPeim.efi`	Calls FSP-M & FSP-S FV
	`SiliconPolicyPeiPostMem.efi`	Publishes silicon initialization configuration
	`DxeIpl.efi`	Load and invoke DXE
	`...`	

Mapped according to .FDF file layout

Stage 2 Platform Libraries - PEI

Item	API Definition Package	Producing Package	Description
`BoardInitLib`	MinPlatformPkg	BoardPkg	Board initialization library.
`SiliconPolicyInitLib`	IntelSiliconPkg	SiliconPkg	Provides default silicon configuration policy data.
`SiliconPolicy UpdateLib`	IntelSiliconPkg	BoardPkg	Provides board updates to silicon configuration policy data.
`TestPointCheckLib`	MinPlatformPkg	MinPlatformPkg	Test point check library.
`TestPointLib`	MinPlatformPkg	MinPlatformPkg	Test point library

Stage 2 Platform Libraries - DXE

Item	API Definition Package	Producing Package	Description
`ResetSystemLib`	MdeModulePkg	SiliconPkg	For DXE reset architecture protocol
`PciHostBridgeLib`	MdeModulePkg	SiliconPkg	For DXE PCI host bridge driver

Platform Initialization Board Hook Modules - Stage 2

```
MinPlatformPkg/
 Include/
  Library/
   BoardinitLib.h
   SiliconPolicyInitLib.h
   SiliconPolicyUpdateLib.h
 Library/
  . . .
  PlatformInit/
    PlatformInitPei/
      PlatformInitPreMem/
      PlatformInitPostMem/
```

```
BoardBootModeDetect()
BoardInitBeforeMemoryInit()

MemoryInit()

SiliconPolicyInitPreMemory()
SiliconPolicyUpdatePreMemory()
SiliconPolicyDonePreMemory()
```

Platform Initialization Memory Init

```
edk2-platforms../
 MinPlatformPkg/
   . . .
   PlatformInit/
    PlatformInitPei/
     PlatformInitPreMem/
     PlatformInitPostMem/

edk2/
   FspIntelFsp2WrapperPkg/
     FspmWrapperPeim
       FspMemoryInitApi()
```

Notify call back

`BoardInitAfterMemoryInit()`

FSP-M

`FspMemoryInitApi()`

Platform Initialization Board Hook Modules

```
MinPlatformPkg/
  . . .
 PlatformInit/
  PlatformInitPei/
   PlatformInitPreMem/
   PlatformInitPostMem/

    BoardInitBeforeSiliconInit()

    PlatformInitEndOfPei()
     BoardInitAfterSiliconInit()
```

```
KabylakeOpenBoardPkg/
 KabylakeRvp3/
  Library/
   BoardInitLib/
      PeiKabylakeRvp3InitPostMemLib.c
        KabylakeRvp3/
        BoardInitBeforeSiliconInit()
          ConfigureGpio()
            . . .
          LateSiliconInit()
```

Hook for Before Silicon Initialization

Platform Initialization Board Hook Modules

```
MinPlatformPkg/
  . . .
 PlatformInit/
  PlatformInitPei/
   PlatformInitPreMem/
   PlatformInitPostMem/

    BoardInitBeforeSiliconInit()

    PlatformInitEndOfPei()
     BoardInitAfterSiliconInit()
```

```
KabylakeOpenBoardPkg/
 KabylakeRvp3/
  Library/
   BoardInitLib/
      PeiBoardInitPostMemLib.c
         BoardInitAfterSiliconInit()
```

Hook for After Silicon Initialization

FSP Wrapper for Silicon Initialization

```
Edk2/
 FspIntelFsp2WrapperPkg/
   FspsWrapperPeim/
     PeiMemoryDiscoveredNotify()
       Finish Memory Migration
       --- > FSP Silicon Init API
       PostFspHobProcess ()
```

```
MinPlatformPkg/FspWrapper/
  Library/
    PeiFspWrapperHobProcessLib/
      FspWrapperHobProcessLib.c
        PostFspsHobProcess()
```

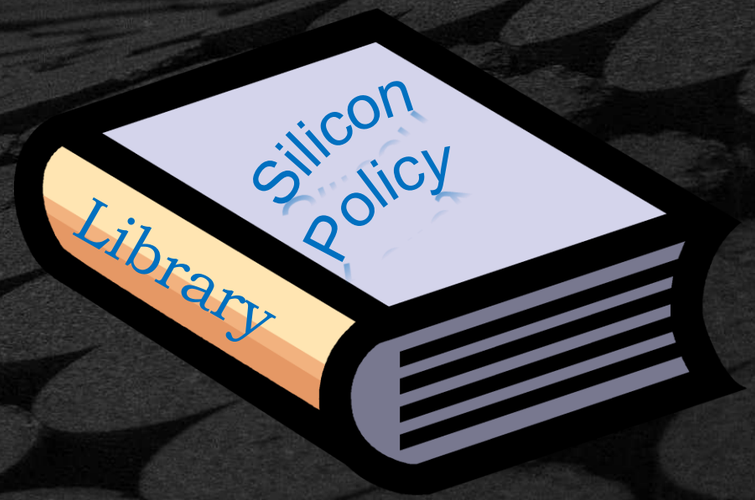
FSP-S

FspSiliconInit API

Silicon Policy Update Lib

Using the `SiliconPolicyUpdateLib`, the board package may reference a variety of sources to obtain the board-specific policy values

1. PCD database
2. UEFI Variable
3. Binary Blob
4. Built-in C structure
5. Hardware information

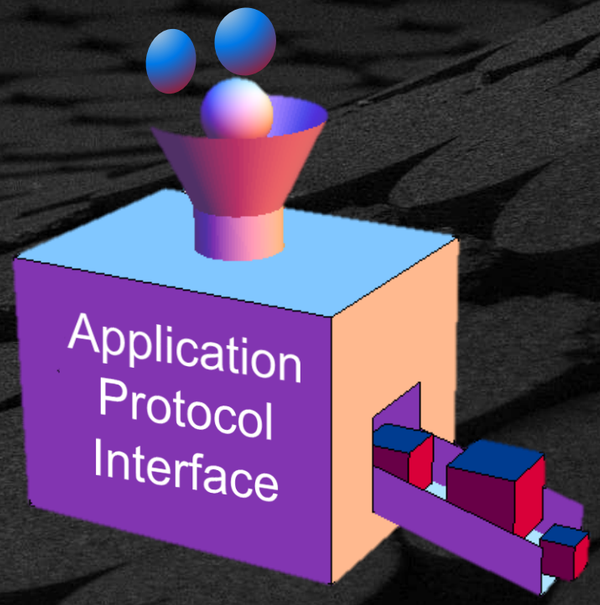


One silicon policy data structure created per silicon module

Silicon Policy APIs

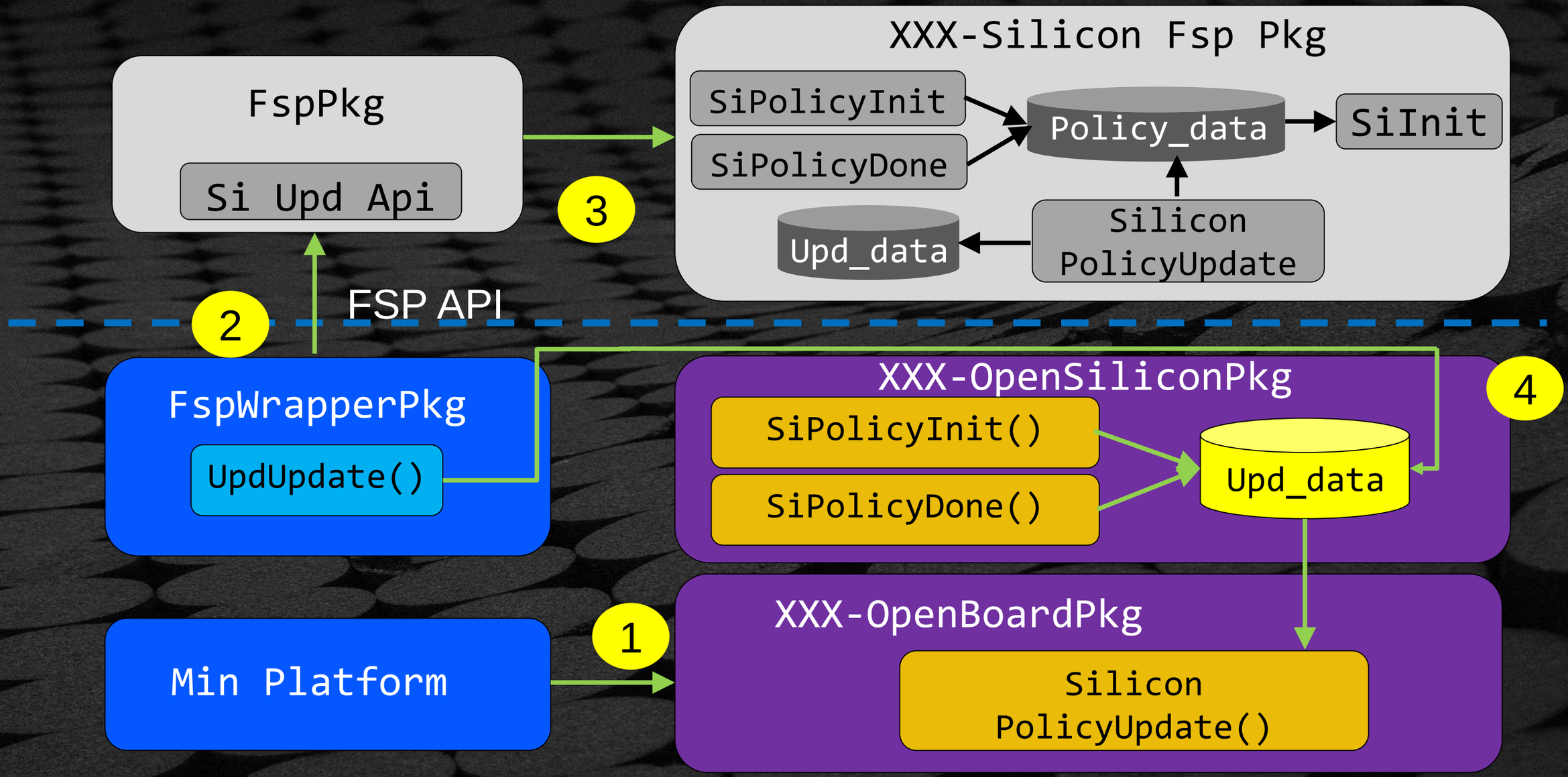
Silicon code expose APIs to:

- Initialize all policy data to the default value, based upon the current silicon.
- Inform silicon code that all policy data have been updated, and they are ready to consume.



Board module only updates non-default values

FSP Silicon Policy Data Flow



Platform Initialization Board Hook Silicon

```
Platform/Intel/
MinPlatformPkg/
   . . .
   PlatformInit/
     PlatformInitPei/

     SiliconPolicyPei/
       SiliconPolicyPeiPostMem.c
        Silicon . . . Entrypoint()

        SiliconPolicyInitPostMem()
        SiliconPolicyUpdatePostMem()
        SiliconPolicyDonePostMem()
```

```
Silicon/Intel/
KabylakeSiliconPkg/
   Library/
     PeiSiliconPolicyInitLibFsp/
       PeiFspPolicyInitLib.c
        SiliconPolicyInitPostMem()

      // Using the FSP Update policy
          PeiFspPchPolicyInit ()
          PeiFspMePolicyInit ()
          PeiFspSaPolicyInit ()
          PeiFspCpuPolicyInit ()
```

Example: Kabylake – post-mem FSP silicon init.

Platform Initialization Board Hook Silicon

```
Platform/Intel/
MinPlatformPkg/
   . . .
   PlatformInit/
     PlatformInitPei/

     SiliconPolicyPei/
       SiliconPolicyPeiPostMem.c
        Silicon . . . Entrypoint()

        SiliconPolicyInitPostMem()
        SiliconPolicyUpdatePostMem()
        SiliconPolicyDonePostMem()
```

```
KabylakeOpenBoardPkg
   FspWrapper
    Library
      PeiSiliconPolicyUpdateLibFsp
        PeiFspPolicyUpdateLib.c
         SiliconPolicyUpdatePostMem()

           PeiFspSaPolicyUpdate ()
           PeiFspPchPolicyUpdate ()
```

Notice the Update is from board FSP Wrapper Library

Platform Initialization Board Hook Silicon

```
Platform/Intel/
MinPlatformPkg/
   . . .
   PlatformInit/
     PlatformInitPei/

     SiliconPolicyPei/
       SiliconPolicyPeiPostMem.c
        Silicon . . . Entrypoint()

        SiliconPolicyInitPostMem()
        SiliconPolicyUpdatePostMem()
        SiliconPolicyDonePostMem()
```

```
Silicon/Intel/
 KabylakeSiliconPkg/
  Library/
   PeiSiliconPolicyInitLibFsp/
      PeiFspPolicyInitLib.c
        SiliconPolicyDonePostMem()
```

Silicon post-mem policy is finalized

Prepare for Hand-off to DXE

Hob Output	• 2 Hob lists - FSP & FSP Wrapper
MTRR Configuration	• 2 locations – after permanent Memory & Prior to DXE IPL
DXE IPL	• Load and invoke DXE

Stage 2 Checklist

Steps to enable a board for Stage 2.

1. Update *NewOpenBoardPkg/BoardXxx*

- Add Board boot mode detection code in `BoardBootModeDetect ()` , *BoardXxx*`/BoardInitLib/Pei`*NewBoardXxx*`InitPreMemoryLib.c`.
  - The boot mode can be hardcoded. It should reflect actual functionality based upon the feature, such as S3 (silicon register), Capsule (variable), Recovery(GPIO).
- Add Board pre-memory initialization code in `BoardInitBeforeMemoryInit ()` and `BoardInitAfterMemoryInit ()` , *BoardXxx*`/BoardInitLib/Pei`*BoardXxx*`InitPreMemLib.c.`
  - It initializes board specific hardware devices, such as GPIO.
  - It also updates pre-memory policy configuration by using PCD
- Add Board policy update code in `SiliconPolicyUpdatePreMemory ()` , *BoardXxx*`/PeiSiliconPolicyUpdateLib/Pei`*BoardXxx*`InitPreMemoryLib.c`.
  - The PCD updated in `BoardInitBeforeMemoryInit ()` might be used here.

2. Ensure all PCDs in the configuration section (DSC files) are correct for your board.
3. Ensure all required binaries in the flash file (FDF files) are correct for your board.
   - Boot, collect log, verify test point results are correct.

EDK II Open Platform Spec Test points Stage 2

Staged Approach by Features

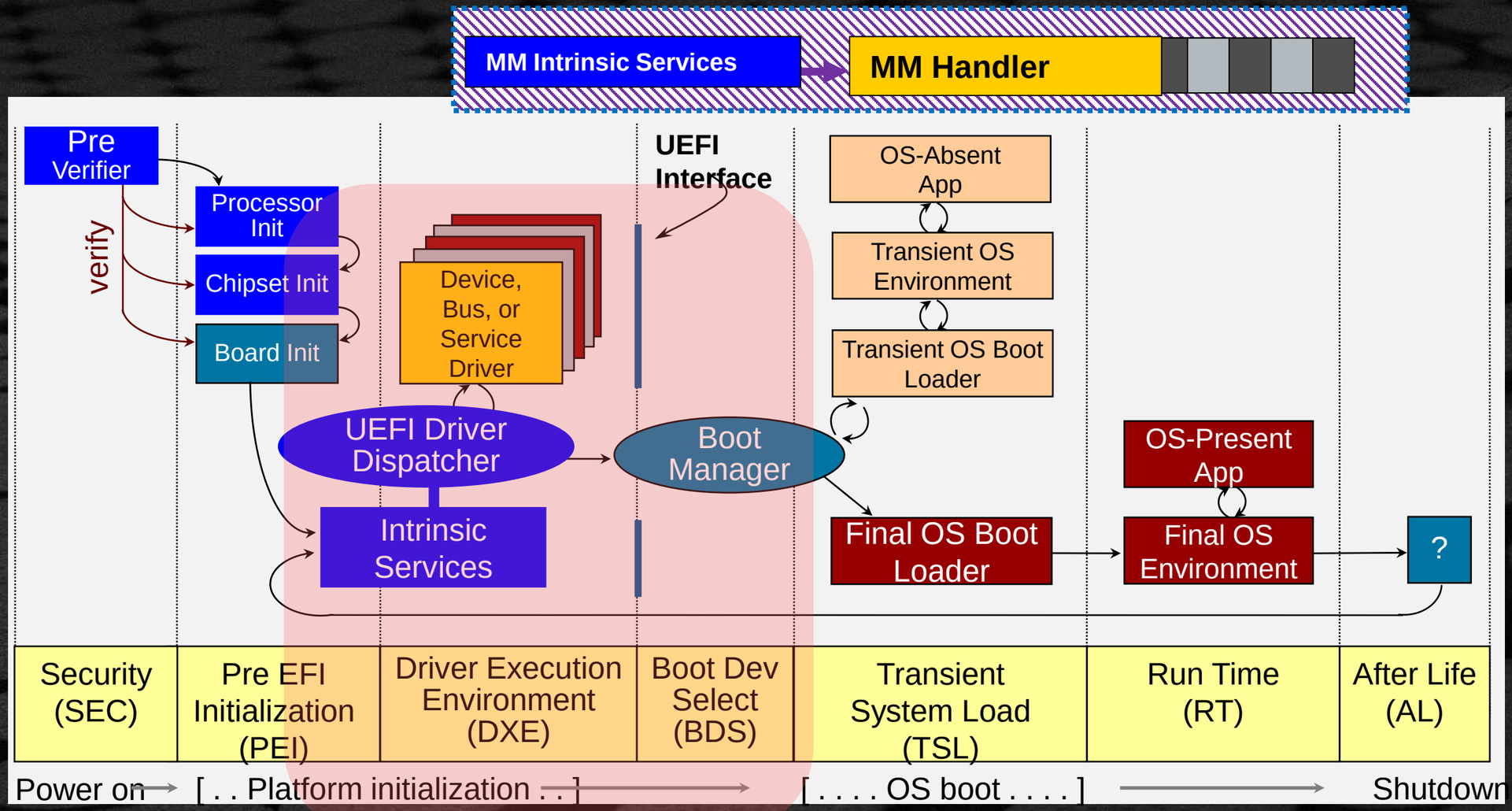
- Platform Firmware Boot Stage PCD

Stage 1	enable debug
Stage 2	memory initialization
Stage 3	boot to UEFI shell only
Stage 4	boot to OS
Stage 5	boot to OS w/ security enabled
Stage 6	Advanced Feature Selection
Stage 7	Performance Optimizations

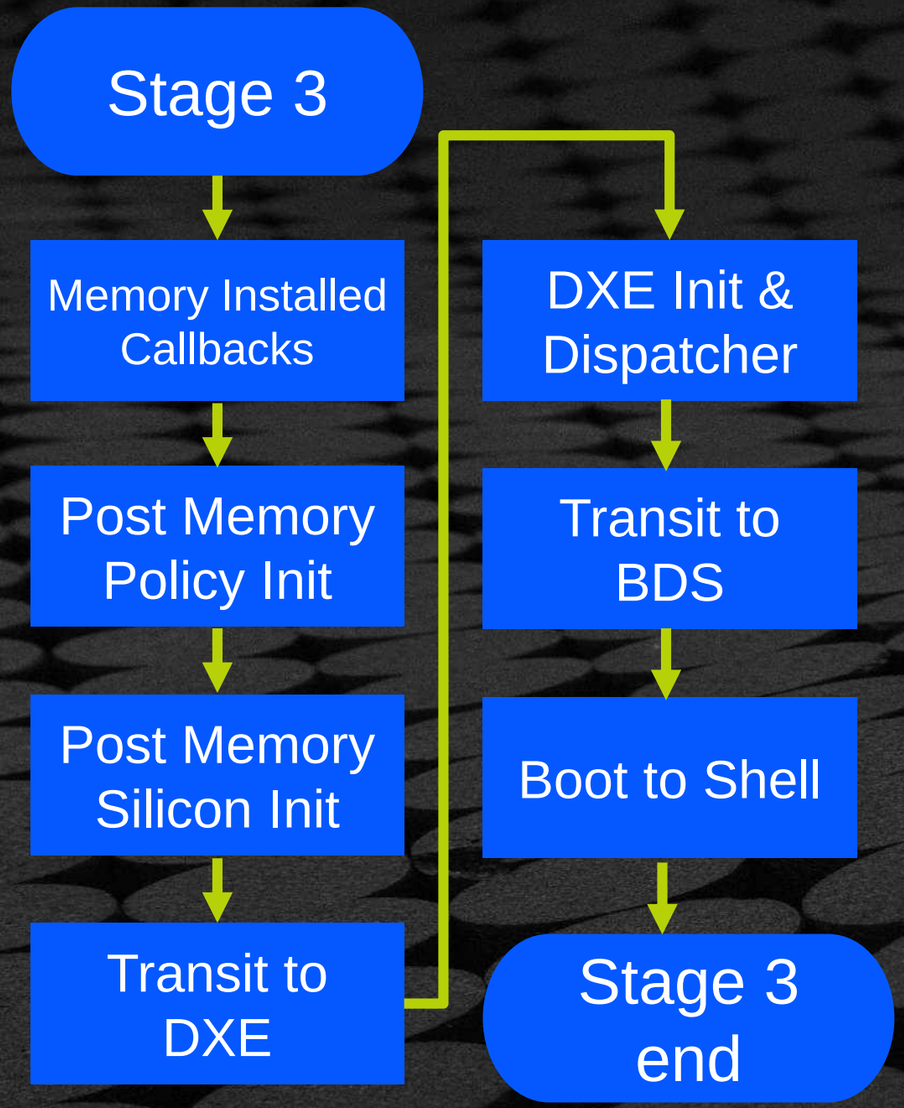
PCD Is tested within .FDF to see which modules to include

Boot Flow – Stage 3

Stage 3



High Level Control Flow – Stage 3



Major Execution Activities

- DXE Initial Program Load (IPL)
- DXE Core initialization and dispatcher execution
- Initialize the generic infrastructure required for the DXE environment
  - DXE architectural protocols - Initialization of architecturally required hardware such as timers
- Post-memory silicon policy initialization
- Serial console input and output capabilities

tianocore

Stage 3 Firmware Volumes

Name	Content
FvUefiBoot	Common UEFI, DXE & BDS Services

Stage 3 FVs Contents

FV	Components	Purpose
FvUefiBoot	`DxeCore.efi`	UEFI System Table, DXE services, Dispatcher and APs
	. . .	All other DXE Modules See .FDF file

Mapped according to .FDF file layout

Platform Architecture Libraries – Stage 3

Stage 3 Modules - List of UEFI and DXE Components: EDK II Open Platform Spec.

Stage 3 Platform Architecture Libraries

Item	API Def Package	Producing Package	Description
`SerialPortLib`	`MdeModulePkg`	`MinPlatformPkg`	Serial port leveraging PEI and HOB initialization.
`PlatformBootManagerLib`	`MdeModulePkg`	`MinPlatformPkg`	Basic platform boot manager port.

Only modules in the board package should be modified

Platform Initialization Board hook Modules

Silicon Initialization done in Stage 2 for Stage 3

```
Platform/Intel/
MinPlatformPkg/
 Include/
  Library/
   BoardinitLib.h
   SiliconPolicyInitLib.h
   SiliconPolicyUpdateLib.h

 Library/
  . . .
  PlatformInit/
    PlatformInitPei/
     PlatformInitPostMem/
```

```
BoardInitBeforeSiliconInit()

SiliconPolicyInitPostemory()
SiliconPolicyUpdatePostMemory()
SiliconPolicyDonePostMemory()

BoardInitAfterSiliconInit()

DxeLoadCore()
```

- No silicon-specific initialization in DXE phase
- HOBs should transfer Init settings to Stage 3

Platform Initialization Board hook Modules - Stage 3 DXE

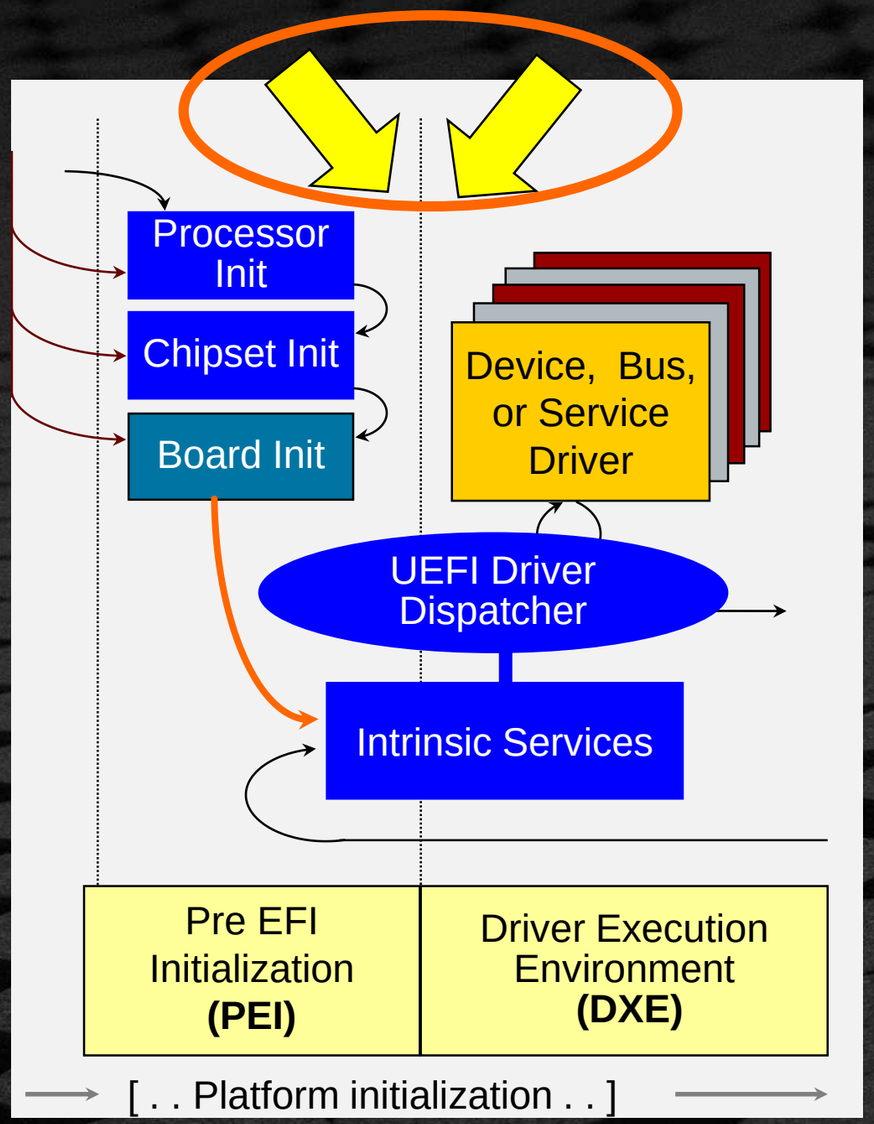
```
Platform/Intel/
MinPlatformPkg/
 Include/
  Library/
   BoardinitLib.h

 Library/
  . . .
  PlatformInit/
    PlatformInitDxe/
      BoardNotificationInit()
```

```
BoardNotificationInit()
```

Create events

- PCI Enumeration complete
- DXE SMM ready to Lock

Initial Program Load (IPL) PEIM for DXE



`DxeLoadCore()`

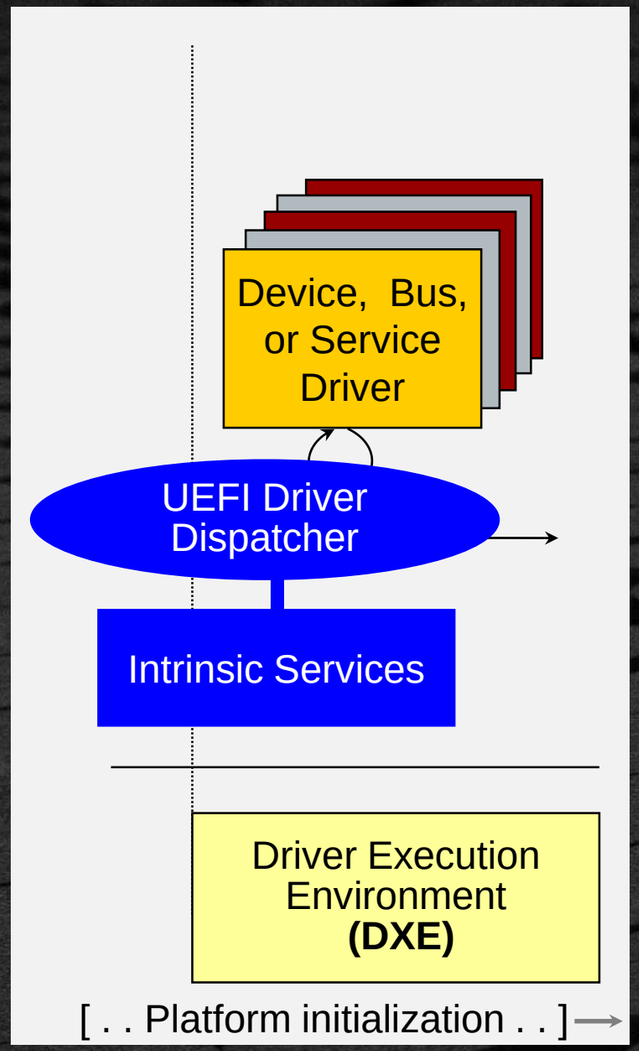
Creates HOBs

Locates DXE main

Switch Stacks

→ `DxeMain()`

DXE Phase Stage 3



DXE Core initialization and dispatcher execution

Establish Architectural Protocols

Install any required DXE / UEFI Drivers for Stage 3

Architectural Protocols for KabyLake

Architectural Protocol	EDK II Location
gEfiBdsArchProtocolGuid	MdeModulePkg/Universal/BdsDxe/
gEfiCapsuleArchProtocolGuid	MdeModulePkg/Universal/CapsuleRuntimeDxe
gEfiCpuArchProtocolGuid	UefiCpuPkg/CpuDxe/
gEfiMetronomeArchProtocolGuid	MdeModulePkg/Universal/Metronome
gEfiMonotonicCounterArchProtocolGuid	MdeModulePkg/Universal/MonotonicCounterRuntimeDxe
gEfiRealTimeClockArchProtocolGuid	PcAtChipsetPkg/PcatRealTimeClockRuntimeDxe
gEfiResetArchProtocolGuid	MdeModulePkg/Universal/ResetSystemRuntimeDxe
gEfiRuntimeArchProtocolGuid	MdeModulePkg/Core/RuntimeDxe
gEfiSecurity2ArchProtocolGuid	MdeModulePkg/Universal/SecurityStubDxe
gEfiTimerArchProtocolGuid	PcAtChipsetPkg/HpetTimerDxe
gEfiVariableArchProtocolGuid	MdeModulePkg/Universal/Variable/RuntimeDxe/ VariableRuntimeDxe or VariableSmmRuntimeDxe (depends on PcdBootToShellOnly)
gEfiVariableWriteArchProtocolGuid	MdeModulePkg/Universal/Variable/RuntimeDxe/ VariableRuntimeDxe or VariableSmmRuntimeDxe (depends on PcdBootToShellOnly)
gEfiWatchdogTimerArchProtocolGuid	MdeModulePkg/Universal/WatchdogTimerDxe

Platform Boot Manager before BDS

- Stage 3 DXE – BDS

```
Platform/Intel/
MinPlatformPkg/
   Bds/
    Library/
      DxePlatformBootManagerLib/
        BdsPlatform.c
           PlatformBootManagerBeforeConsole()
```

`PlatformBootManagerLib`

- Call before BDS to connect all devices
- Creates event, `OnReadyToBootCallBack`
- Updates consule etc.

Typically, no board porting is required

Stage 3 Checklist

Steps to enable a board for Stage 3.

1. Add board post-memory initialization code in BoardInitBeforeSiliconInit () and BoardInitAfterSiliconInit () , *Board*Pkg/BoardInitLib/Pei*NewBoardXxx*InitPostMemoryLib.c.
   - Initialize board-specific hardware device, such as GPIO.
   - Update post-memory policy configuration by using PCD.
2. Add board policy update code in SiliconPolicyUpdatePostMemory () , *Board*Pkg/PeiSiliconPolicyUpdateLib /Pei*NewBoardXxx*InitLib.c.
   - The PCD updated in BoardInitBeforeSiliconInit () might be used here.
3. Add board initialization DXE code in BoardInitAfterPciEnumeration () , BoardInitReadyToBoot(), BoardInitEndOfFirmware () .
   – Note: *The functions may be empty if no updating is required.*
4. Ensure all PCDs in the configuration section (DSC files) are correct for your board. - Set gMinPlatformPkgTokenSpaceGuid.PcdBootStage = 3
5. Ensure all required binaries in the flash file (FDF files) are correct for your board.
6. Boot, collect debug log, and verify the test point results are correct.

EDK II Open Platform Spec Test points Stage 3

Staged Approach by Features

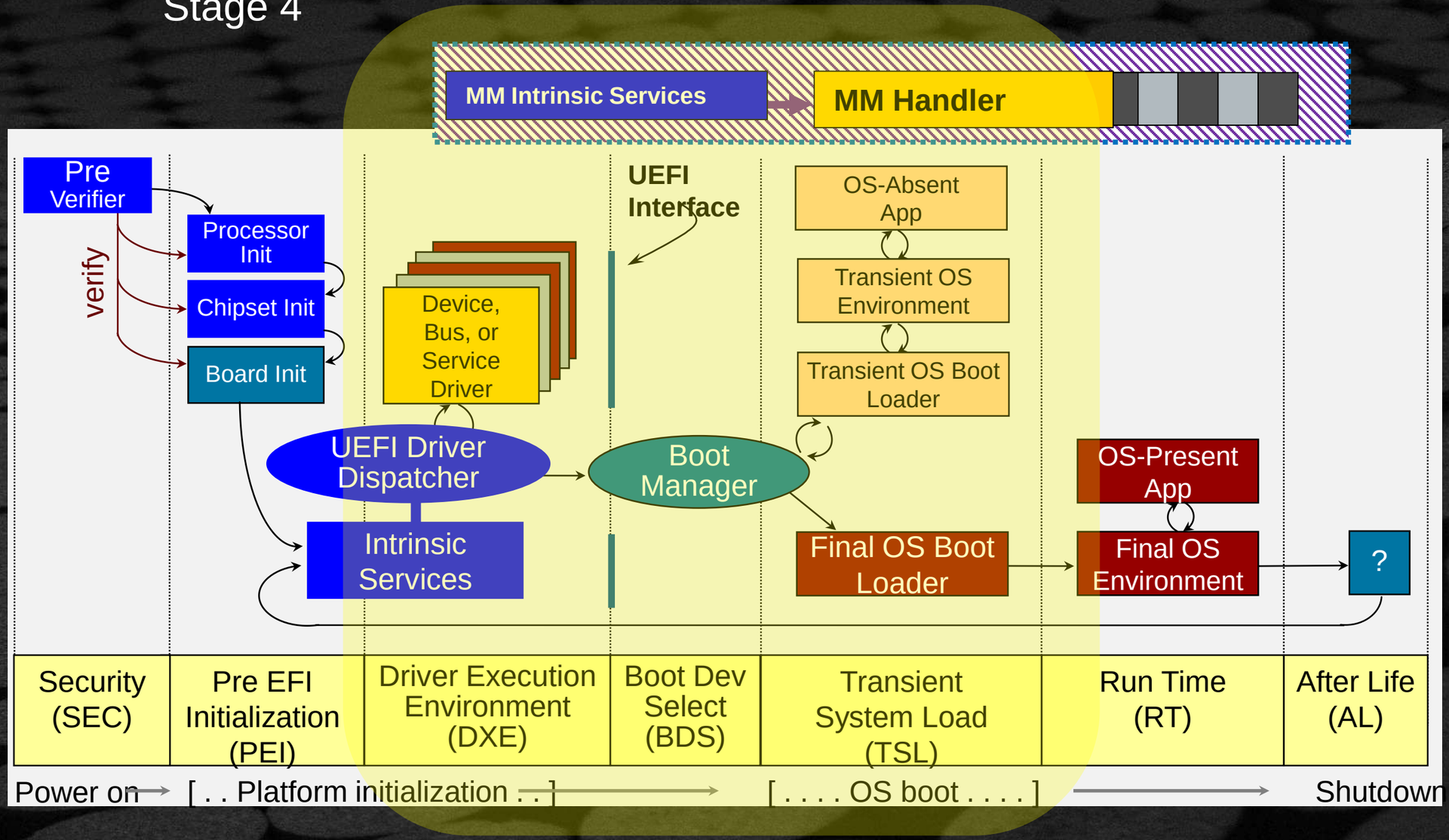
- Platform Firmware Boot Stage PCD

Stage 1	enable debug
Stage 2	memory initialization
Stage 3	boot to UEFI shell only
Stage 4	boot to OS
Stage 5	boot to OS w/ security enabled
Stage 6	Advanced Feature Selection
Stage 7	Performance Optimizations

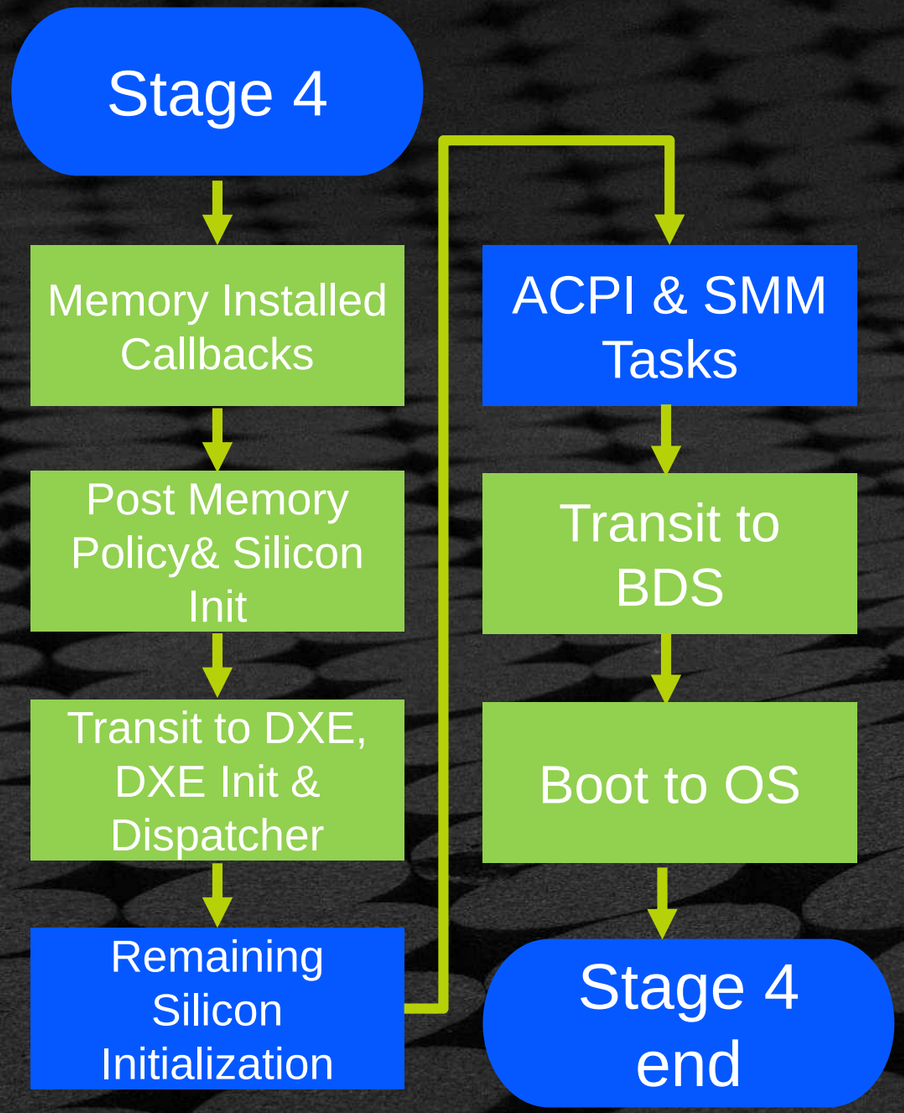
PCD Is tested within .FDF to see which modules to include

Boot Flow – Stage 4

Stage 4



High Level Control Flow – Stage 4



Major Execution Activities

- Minimum ACPI Table Initialization
- Additional input, output, and storage support based on platform and operating system requirements
- SMM
- Perform ACPI enable/disable
- Kernel debug support
- UEFI variable support

Re-use from Stage 3

Stage 4 additional tasks

tianocore

Stage 4 Firmware Volumes

Name	Content
`FvOsBoot`	Continued DXE/BDS Services
`FvLateSilicon`	ACPI and SMM silicon support

- Finalize silicon initialization
- Add basic operating system required interfaces
- Add support for minimal featured operating system boot.

tianocore

Stage 4 FVs Contents

FV	Components	Purpose
FvOsBoot	`FvLateSilicon.fv`	Additional silicon initialization support that is performed late in boot
	List of ACPI modules	ACPI Table, Platform & Board DXE
	List of SMM modules	SmmIpl, SMM Core,SMM CPU etc, (DXE)
	List Storage media modules	Sata, USB, . . . , (DXE)

Mapped according to .FDF file layout

Platform Architecture Libraries – Stage 4

Item	API Def Package	Producing Package	Description
BoardAcpiLib	MinPlatformPkg	BoardPkg	Services for ACPI table creation and SMM enable/dis
BoardInitLib	MinPlatformPkg	BoardPkg	Board specific initialization hook at DXE phase for `BoardNotificationInit`

Board porting requires creation of libraries produced by the `BoardPkg`

Tips for Board Specific ACPI

Board Library	• Use board specific library for ACPI global NVS area assignments instead defining in ASL Code
NVS Space	• Do Not define huge amount of board specific configuration in the global NVS area
Special Features	• Board specific devices or advanced features should be moved to the board specific directory
ASL & C In Same Dir	• Use the same directory for the GlobalNvs.asl and GobalNvsAreaDef.h files

tianocore

Example: Board Specific ACPI

- Tip: Board specific device selection - define a board-neutral name
- Data structure is board neutral …/Include/Acpi/GlobalNvsAreaDef.h
- KabylakeOpenBoardPkg/KabylakeRvp3/Library/BoardAcpiLib/ \
  DxeKabylakeRvp3AcpiTableLib.c - Link

```
VOID
KabylakeRvp3UpdateGlobalNvs (
VOID
)
{  // Update global NVS area for ASL and SMM init code to use
mGlobalNvsArea.Area = (VOID *)(UINTN)PcdGet64 (PcdAcpiGnvsAddress);
mGlobalNvsArea.Area->PowerState = 1;
mGlobalNvsArea.Area->NativePCIESupport = PcdGet8 (PcdPciExpNative);
mGlobalNvsArea.Area->ApicEnable = GLOBAL_NVS_DEVICE_ENABLE;
mGlobalNvsArea.Area->LowPowerS0Idle = PcdGet8 (PcdLowPowerS0Idle);
mGlobalNvsArea.Area->Ps2MouseEnable = FALSE;
mGlobalNvsArea.Area->Ps2KbMsEnable = PcdGet8 (PcdPs2KbMsEnable);
}
```

Tip: use board specific library for ACPI NVS Area

Required Function Interfaces DXE

```
Platform/Intel/
MinPlatformPkg/
  PlatformInit/
    Library/
      MultiBoardInitSupportLib/
        DxeMultiBoardInitSupportLib.c
    PlatformInitDxe/
      PlatformInitDxe.c
        BoardNotificationInitEntryPoint()
```

`BoardInitLib`

Board specific initialization hook at DXE phase.

Notify:
`OnPciEnumerationComplete`
`SmmReadyToLockRegistration`

Registers two callbacks to call FSP notifies

Required SMM Function Interfaces DXE

```
Platform/Intel/
KabylakeOpenBoardPkg/
  KabylakeRvp3/
    Library/
      BoardAcpiLib/
        SmmBoardAcpiEnableLib.c

      or

        SmmMultiBoardAcpiSupportLib.c
```

BoardAcpiEnableLib

Board specific initialization hook at DXE phase.

```
SiliconEnableAcpi()
SiliconDisableAcpi ()
```

Required ACPI Function Interfaces DXE

```
Platform/Intel/
KabylakeOpenBoardPkg/
  KabylakeRvp3/
    Library/
      BoardAcpiLib/
        DxeBoardAcpiTableLib.c

      or

        DxeMultiBoardAcpiSupportLib.c
```

`BoardAcpiTableLib`

Board specific initialization hook at DXE phase.

```
BoardUpdateAcpiTable ()
```

Stage 4 Checklist

Steps to enable a board for Stage 4.

1. Install the minimal DSDT - In rare cases: Install board-specific SSDT
2. Ensure all PCDs in the configuration section (DSC files) are correct for your board. - Set `gMinPlatformPkgTokenSpaceGuid.PcdBootStage = 4`
3. Ensure all required binaries in the flash file (FDF files) are correct for your board.
4. Boot, collect log, verify test point results defined are correct

EDK II Open Platform Spec Test points stage 4

Staged Approach by Features

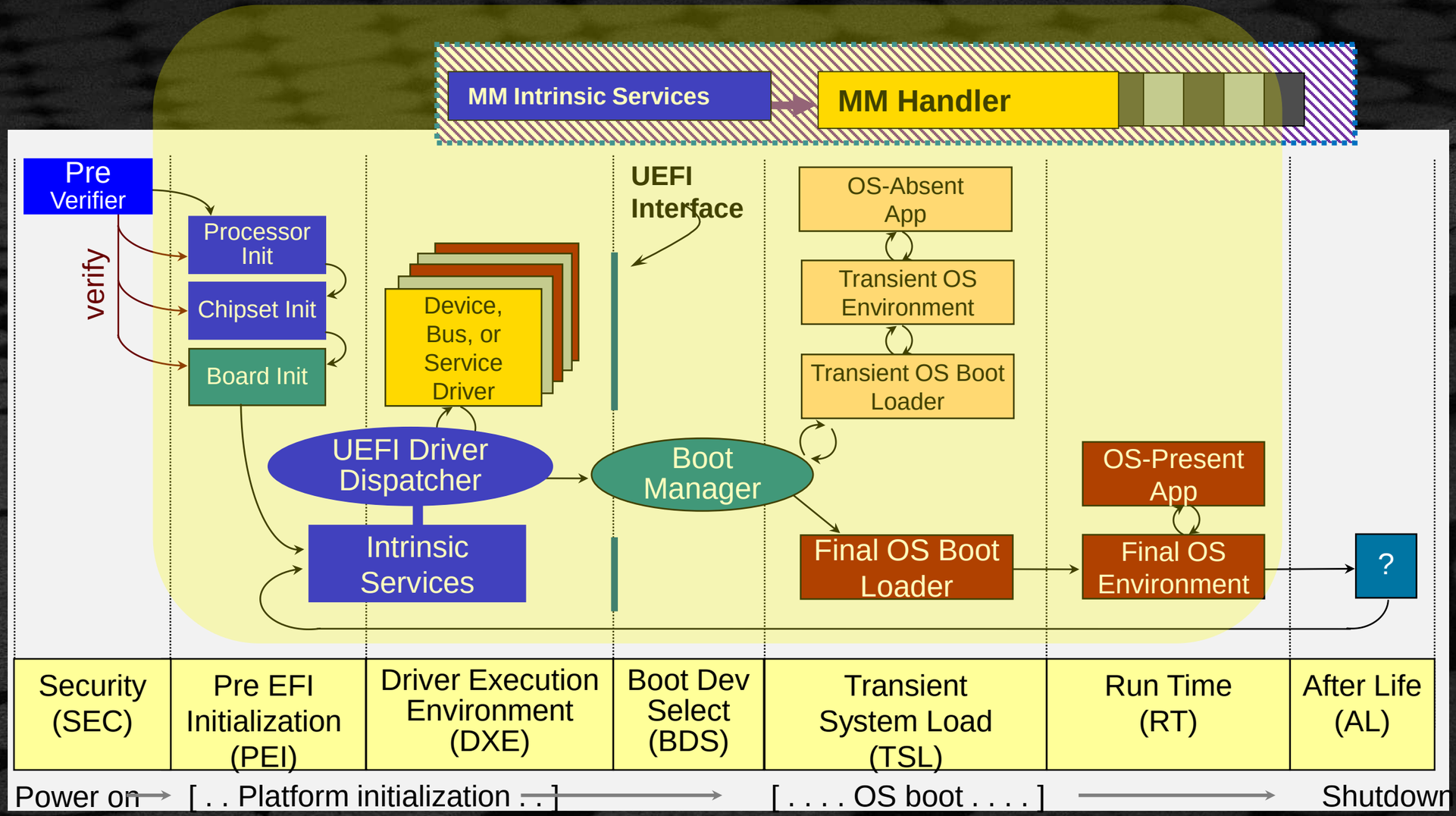
- Platform Firmware Boot Stage PCD

Stage 1	enable debug
Stage 2	memory initialization
Stage 3	boot to UEFI shell only
Stage 4	boot to OS
Stage 5	boot to OS w/ security enabled
Stage 6	Advanced Feature Selection
Stage 7	Performance Optimizations

PCD Is tested within .FDF to see which modules to include

Boot Flow – Stage 5

Stage 5

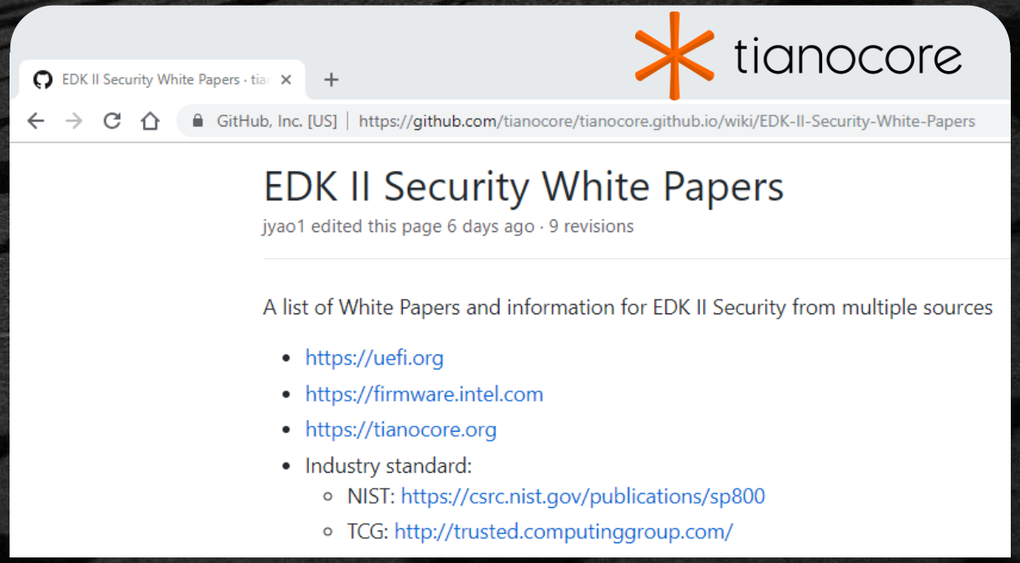


Stage 5 is Focused on Security

List of Documents on Security with EDK II - EDK II tianocore Wiki

Minimal security checks should be done

- Chipsec
- Microsoft Hardware Security Test Interface (HSTI)
- Windows Security Mitigations Table (WSMT).
- Memory Attribute Table
- SMM Memory Attribute Table
- 3rd party option ROM
- Trusted Console - Trusted Storage
- DMA protection for Intel® VT-d/IOMMU
- SMI Handler
- TPM
- Firmware Update



tianocore

Stage 5 Porting - Goals

Satisfy industry standard security specifications

- Establish a Static Root of Trust for Verification (S-RTV)
- Static Root of Trust for Measurement (S-RTM)
- Direct Memory Access (DMA) Protection

Stage 5 – Security- Features

TCG trusted boot

- TCG measured boot chain of trust should be enabled in this stage.

UEFI Secure Boot

- Authenticated UEFI Variable support must be completely functional. This is a basic requirement for secure authentication and UEFI Secure Boot key management

Stage 5 Firmware Volumes

Name	Content
FvSecurity	Security related modules
FvSecurityPreMemory	
FvSecurityPostMemory	
FvSecurityLate	
NvStorage	Real NV storage on flash

Supports key security features.

tianocore

Stage 5 FVs Contents

FV	Components	Purpose
FvSecurity		
	List of TCG modules	TPM Services, Config, Platform, etc
	List of Intel VT-d modules	IOMMU PEI & DXE Services
	List Security modules	Provide security architecture protocol Secure boot UI config …
	List of variable SMM modules	Fault-tolerant, Variable, SMM Services

Mapped according to .FDF file layout

Modules – Stage 5

PEI Components

- Tcg2 Pei – SecurityPkg
- Tcg2 platform Pei – PlatformPkg
- Intel Vt-d PMR - SiliconPkg

DXE Components

- Tcg2 DXE – SecurityPkg
- Tcg2 platform Dxe – PlatformPkg
- SecureBootConfigDxe -SecurityPkg
- Intel Vt-d Dxe- SiliconPkg

SMM Components

- Tcg2 SMM – SecurityPkg
- FaultTolerantWriteSMM
- VariableSMM

tianocore

Stage 5 Required Functions

Phase	Name	Purpose
PEI	`PeimEntryMA ()`	Entry point for the TPM2 PEIM
	`IntelVTdPmrInitialize ()`	Entry point for the VT-d PEIM
DXE	`DriverEntry ()`	Entry point for the TPM2 DXE module
	`IntelVTdInitialize()`	Entry point for the VT-d DXE module
	`UserPhysicalPresent()`	Indicates whether a physical user is present for UEFI secure boot
	`ProcessTcgPp & ProcessTcgMor`	ProcessTcgPp Process the TPM physical presence (PP) request & memory overwrite request (MOR)
SMM	`InitializeTcgSmm ()`	Entry point for the TPM2 SMM module
	`MemoryClearCallback ()`	Callback function for setting the MOR variable

Security Related PCDs and Variables

- Authenticated Variables -UEFI Secure Boot
  - `PK, KEK, db and dbx`
- TPM Policy: TCG trusted boot
  - `PcdTpmInstanceGuid, PcdTpm2InitializationPolicy, PcdTpm2SelfTestPolicy`
- Memory Only Reset (MOR) - Policy: TCG MOR
  - `PRE_MEM_SILICON_POLICY, L“MemoryOverwriteRequestControl”`
- Intel® VT-d – DMA protection
  - `PcdVTdPolicyPropertyMask`

Definitions may be both source and binary

Security Feature Related PCDs

- Enable SMI handler profile
  - gMinPlatformModuleTokenSpaceGuid.PcdSmiHandlerProfileEnable
- Enable TPM
  - gMinPlatformModuleTokenSpaceGuid.PcdTpm2Enable
- Enable UEFI Secure boot
  - gMinPlatformModuleTokenSpaceGuid.PcdUefiSecureBootEnable

Enable/Disable in `OpenBoardPkgConfig.dsc`

Stage 5 Checklist

Steps to enable a board for Stage 5:

1. Update `BoardPkg/Board`.
   - Deploy the UEFI secure boot variables (PK/KEK/db/dbx)
   - Configure `PcdTpmInstanceGuid` to select TPM hardware. Default value of `gEfiTpmDeviceInstanceTpm20DtpmGuid` is usually correct.
2. UEFI secure boot: Update PlatformSecureLib : `UserPhysicalPresent()` to check if a user is physically present to authorize change of authenticated variables.
3. For TCG Trusted Boot:
   - May select TPM2 instance `PcdTpmInstanceGuid`.
   - May set `PcdFirmwareDebuggerInitialized` based on whether or not a Firmware Debugger is attached to platform.
4. For DMA Protection: include IOMMU driver for DMA protection, if silicon supports IOMMU.
5. Ensure all PCDs in the configuration section (DSC files) are correct for your board.
   Set `MinPlatformPkgTokenSpaceGuid.PcdBootStage = 5`
6. Ensure all required binaries in the flash file (FDF files) are correct for your board.
7. Boot, collect log, verify test point results are correct.

EDK II Open Platform Spec Test points stage 5

Staged Approach by Features

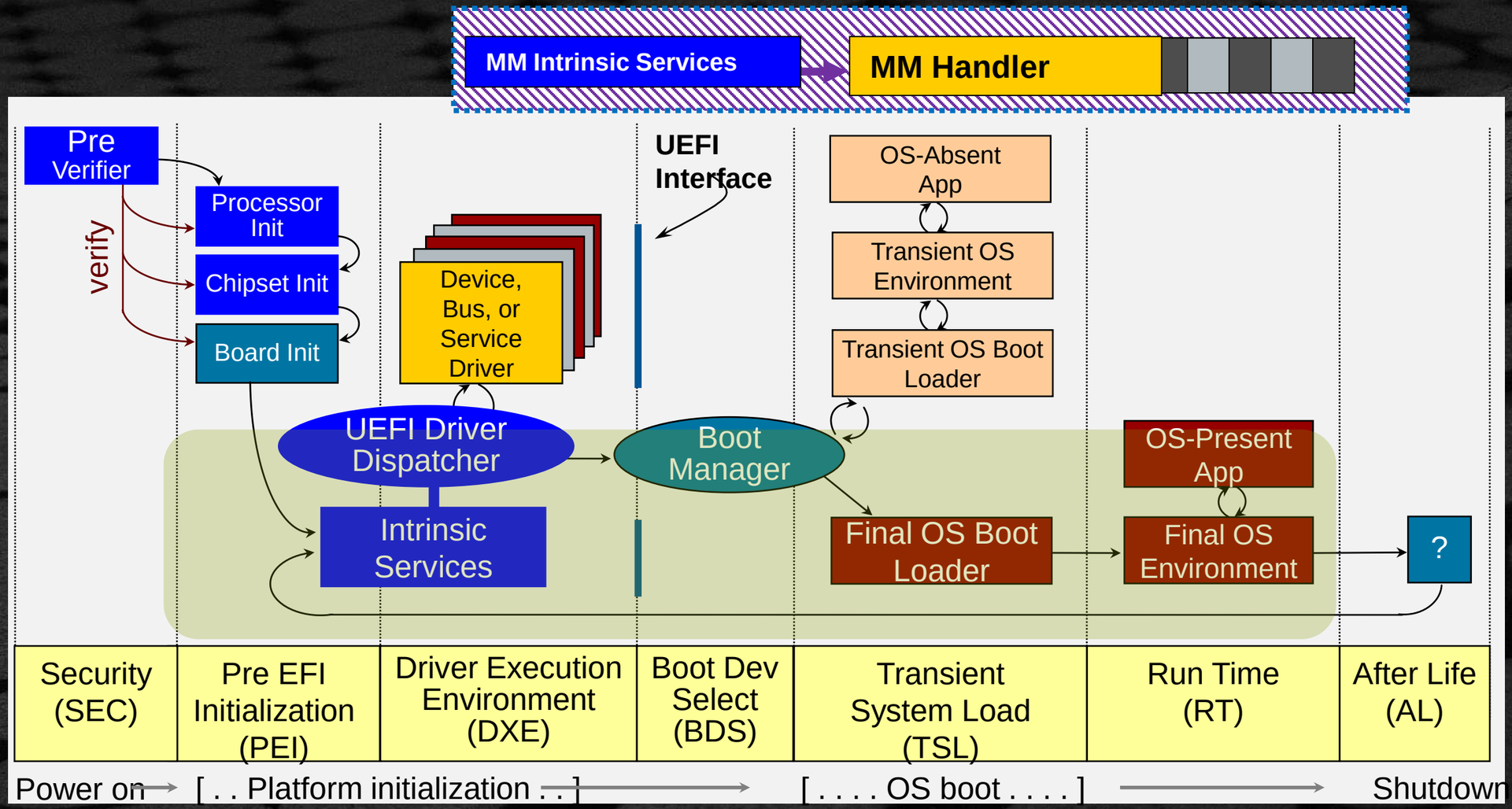
- Platform Firmware Boot Stage PCD

Stage 1	enable debug
Stage 2	memory initialization
Stage 3	boot to UEFI shell only
Stage 4	boot to OS
Stage 5	boot to OS w/ security enabled
Stage 6	Advanced Feature Selection
Stage 7	Performance Optimizations

PCD Is tested within .FDF to see which modules to include

Boot Flow – Stage 6

Stage 6



Add Non-essential Features - Stage 6

Examples:

Core Features

- Signed Capsule update
- Signed Recovery
- Source Debug Enable
- NVMe (secondary storage)
- eMMC (secondary storage)
- S3 resume
- Network

Platform Features

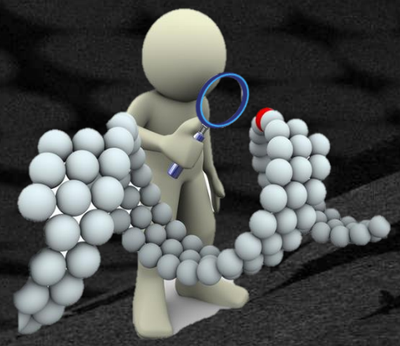
- SMBIOS
- Intel® Active Management Technology (AMT)
- Thunderbolt™

Board Features

- Embedded Controller
- Setup (policy)

Limit required features to reduce complexity

Advanced Feature Requirements



Each advanced feature:

- Has low coupled component interactions
- Complete, mutually independent and only depend on packages in edk2 rep.
- Organized as cohesive packages related to the same advanced feature
- Support enabling individually as part of porting within stage 6
- Possible to distribute independently in binary form – add/remove PCD Switch
- Self-documenting – contains a ReadMe.md template

tianocore

Advanced Feature Design

- Feature Template Readme.md

- Firmware Volumes
- Modules
- Required Functions
- Configuration
- Data Flows
- Control Flows
- Build Files
- Test Point Results
- Functional Exit Criteria
- Feature Enabling Checklist
- Common Optimizations

tianocore

Advanced Feature Design

- Feature Template Readme.md

This template should be included in feature review and placed in the feature root directory as README.md

Firmware Volumes - The binary containers needed for the feature.

Modules - The EDK II component binaries and static libraries required.

Required Functions - Functions that are useful for understanding, porting, or debugging the feature and how these key functions are integrated into the Stage I-V required functions.

Configuration - The configurable parameters for a given feature.

Data Flows - The architecturally defined data structures and flows for a given feature.

Control Flows - Key control flows for the feature.

Build Files - The DSC/FDF for integrating the feature.

Test Point Results - The test that can verify porting is complete for the feature.

Functional Exit Criteria - The testable functionality for the feature.

Feature Enabling Checklist - The required activities to achieve desired functionality for the feature.

Common Optimizations - Common size or performance tuning options for this feature.

tianocore

Stage 6 Firmware Volumes

Name	Content
FvAdvanced	Advanced feature software stacks
FvAdvancedPreMem	Stage 1 SEC PEI
FvAdvancedPostMem	Stage 2 PEI DXE
FvAdvancedLate	Stage 3-4 DXE UEFI / OS

Advanced Features should be modular to these stages of the boot flow

Example – Adding Network

The Network stack modules are board and silicon independent thus no porting is necessary. Feature is capable with Stage 3-4.

1. Set PCDs:

   `gAdvancedFeaturePkgTokenSpaceGuid.PcdNetworkEnable|TRUE`

   `gEfiMdeModulePkgTokenSpaceGuid.PcdEfiNetworkSupport|TRUE`

2. Add the network modules to the board `.dsc` and `.fdf` within the Stage 6 FV
3. Common `.dsc` and `.fdf` include files found in `edk2/NetworkPkg`
   - `NetworkDefines.dsc.inc`
   - `NetworkPcds.dsc.inc`
   - `NetworkPkg/NetworkLibs.dsc.inc`
   - `NetworkPkg/NetworkComponents.dsc.inc`
   - `Network.fdf.inc`
4. Add appropriate `!include` statements to `OpenBoardPkg.dsc &.fdf`

Example Include Network in Board .DSC

OpenBoardPkg.dsc

```
. . .
[Defines]
!include NetworkPkg/NetworkDefines.dsc.inc
. . .

[LibraryClasses.common]
!include NetworkPkg/NetworkLibs.dsc.inc


. . .

[Components.X64]
!include NetworkPkg/NetworkComponents.dsc.inc
```

Add the NetworkPkg “`!include`“ file statements in the sections shown

Example Include Network in Board .FDF

OpenBoardPkg.fdf

```
. . .
[FV.FvAdvancedLate]
. . .
FvNameGuid         = 05411CAD-6C35-4675-B6CA-8748032144B4
!include NetworkPkg/Network.fdf.inc

. . .

[FV.FvAdvanced]
. . .
FvNameGuid         = B23E7388-9953-45C7-9201-0473DDE5487A
. . .
FILE FV_IMAGE = 07FC4960-5322-4DDC-A6A4-A17DE492DFE3 {
       SECTION GUIDED EE4E5898-3914-4259-9D6E-DC7BD79403CF
           PROCESSING_REQUIRED = TRUE {
         SECTION FV_IMAGE = FvAdvancedLate
       }
     }
```

Add the NetworkPkg “!include” statements in the sections shown

tianocore

Example – Disable Some Network Features

Remove IPv4 in OpenBoardPkg.dsc

```
. . .
[Defines]
!include NetworkPkg/NetworkDefines.dsc.inc
  DEFINE NETWORK_IP4_ENABLE = FALSE
```

Remove ISCSI in OpenBoardPkg.dsc

```
[Defines]
!include NetworkPkg/NetworkDefines.dsc.inc
 DEFINE NETWORK_ISCSI_ENABLE = FALSE
```

Example – Adding Thunderbolt™

Thunderbolt is a specific feature for the Kabylake

Set `gBoardModuleTokenSpaceGuid.PcdTbtEnable|TRUE` in `OpenBoardPkgConfig.dsc`

`OpenBoardPkg.dsc`

```
!if gBoardModuleTokenSpaceGuid.PcdTbtEnable == TRUE
# Enable For Thunderbolt(TM)
[LibraryClasses.common]
  TbtCommonLib|KabylakeOpenBoardPkg/Features/Tbt/Library/PeiDxeSmmTbtCommonLib/TbtCommonLib.inf
  DxeTbtPolicyLib|KabylakeOpenBoardPkg/Features/Tbt/Library/DxeTbtPolicyLib/DxeTbtPolicyLib.inf

[LibraryClasses.IA32]
  PeiTbtPolicyLib|KabylakeOpenBoardPkg/Features/Tbt/Library/PeiTbtPolicyLib/PeiTbtPolicyLib.inf
  PeiDTbtInitLib|KabylakeOpenBoardPkg/Features/Tbt/Library/Private/PeiDTbtInitLib/PeiDTbtInitLib.inf

[Components.IA32]
  KabylakeOpenBoardPkg/Features/Tbt/TbtInit/Pei/PeiTbtInit.inf

[Components.X64]
  KabylakeOpenBoardPkg/Features/Tbt/TbtInit/Smm/TbtSmm.inf
  KabylakeOpenBoardPkg/Features/Tbt/TbtInit/Dxe/TbtDxe.inf
  KabylakeOpenBoardPkg/Features/PciHotPlug/PciHotPlug.inf
!endif
```

Example Include Tbt in Board .FDF

OpenBoardPkg.fdf

```
. . .
[FV.FvAdvancedPreMem]
. . .
FvNameGuid = 6053D78A-457E-4490-A237-31D0FBE2F305

!if gBoardModuleTokenSpaceGuid.PcdTbtEnable == TRUE
INF. . ./Features/Tbt/TbtInit/Pei/PeiTbtInit.inf
!endif

. . .

[FV.FvAdvancedLate]
. . .
FvNameGuid = 6053D78A-457E-4490-A237-31D0FBE2F305

!if gBoardModuleTokenSpaceGuid.PcdTbtEnable == TRUE
INF. . ./Features/Tbt/TbtInit/Dxe/TbtDxeinf
INF. . ./Features/PciHotPlug/PciHotPlug.inf
INF. . ./Features/Tbt/TbtInit/Smm/TbtSmm.inf
!endif
```

```
[FV.FvAdvanced]
. . .
FvNameGuid          = B23E7388-9953-45C7- . . .
FILE FV_IMAGE = 35E7406A-5842-4F2B . . . {
       SECTION FV_IMAGE = FvAdvancedPreMem
     }

FILE FV_IMAGE = F5DCB34F-27EA-48AC . . . {
       SECTION FV_IMAGE = FvAdvancedPostMem
     }

FILE FV_IMAGE = 07FC4960-5322-4DDC- . . . {
       SECTION GUIDED EE4E5898-3914-425 . . .
           PROCESSING_REQUIRED = TRUE {
         SECTION FV_IMAGE = FvAdvancedLate
       }
     }
```

Stage 6 Checklist

Steps to enable a board for Stage 6:

1. Set gMinPlatformPkgTokenSpaceGuid.PcdBootStage = 6
2. If required, create board specific library class instance for feature.
3. Add appropriate updates to PCDs and library class implementations to `NewBoardXxx .dsc` files
4. Add required modules to appropriate FV sections in `OpenBoardPkg.fdf`
5. Verify Feature enabled is added to build (check Build Directory)
6. Verify the feature added functionally works. This may require several tests depending on the added feature.

Updates depend on the features being added

Staged Approach by Features

- Platform Firmware Boot Stage PCD

Stage 1	enable debug
Stage 2	memory initialization
Stage 3	boot to UEFI shell only
Stage 4	boot to OS
Stage 5	boot to OS w/ security enabled
Stage 6	Advanced Feature Selection
Stage 7	Performance Optimizations

PCD Is tested within .FDF to see which modules to include

Tuning for Performance

Check which features are needed and disable unsupported features

Removed unused components from the defined FV

Use available tools to perform detailed analysis of performance and size

See EDK II Open Platform Spec Stage 7: Tuning

tianocore

SUMMARY

- Define the porting task list for porting existing open platforms in EDK II
- Determine the necessary porting for each stage and boot phase of a new EDK II open platform project


Questions?

Return to Main Training Page

Return to Training Table of contents for next presentation link


tianocore