InsydeH2O® 5.4

Alder Lake OEM Customization Guide

Revision 0.7
Dec. 08, 2021

Disclaimer

Insyde Software provides this document and the programs "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose.

This document could contain technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in future revisions of this document. Insyde Software is under no obligation to notify any person of the changes.

The following trademarks are used in this document:

Insyde and InsydeH2O are registered trademarks or trademarks of Insyde Software. All other trademarks or trade names are property of their respective holders.

Contents

Revision Number	Description	Author	Release Date
0.1	Initial release.	Evonne Li	October 23, 2020
0.2	Update chapter "Program SsidSvid"	Jason CY Hsu	November 17, 2020
0.3	1. Update Bios Guard chapter 2. Add chapter 27 Chasm Falls	Esther Lee	June 2, 2021
0.4	1. Update 27 Chasm Falls 2. Add 28 Thunderbolt Retimer 3. Add 29 Hardware Security Test Interface	Ricky Chen	Aug. 5, 2021
0.5	Enhance BootGuard Section	Patrick Kuo	September 24, 2021
0.6	Extended BIOS Customization	Rex Wu	Oct. 13, 2021
0.7	Rename Section 27.4 "How to adjust SBB size" to "How to adjust SBB", and enhance the content.	Esther Lee	Dec.08, 2021

1 Introduction

We have created this document to assist engineers in developing InsydeH2O 5.4 BIOS for their Alder Lake project. Please follow the Intel reference code for Alder Lake chipset, CPU, and features porting. Alder Lake InsydeH2O 5.4 BIOS has used it and we have tested it on the Alder Lake CRB.

When using Alder Lake releases from Insyde Software, we strongly suggest following these recommendations for OEM projects:

1. Update Intel reference code and bug fixes.

2. Reference Alder Lake CRB $(PROJECT_REL_PATH)/$(PROJECT_PKG) code and then customize it for the OEM project.

The following terms are used throughout this document to describe varying aspects of input localization:

$(PROJECT_REL_PATH)	For CRB, it's Board/Intel For Project, it may be Board/Xyz
$(PROJECT_PKG)	For CRB, it's AlderLakePMultiBoardPkg or AlderLakeSMultiBoardPkg For Project, it may be BoardNameBoardPkg
$(CHIPSET_REL_PATH)	Intel/AlderLake
$(CHIPSET_PKG)	AlderLakeChipsetPkg
$(PLATFORM_SI_PACKAGE)	ClientOneSiliconPkg
$(PLATFORMSAMPLE_PACKAGE)	AlderLakePlatSamplePkg
$(PLATFORM_BOARD_PACKAGE)	AlderLakeBoardPkg

AlderLakeBoardPkg is part of Intel platform sample code, and it's under $(CHIPSET_REL_PATH)

AlderLakeS**Multi**BoardPkg is Insyde CRB implementation sample, and it's under $(PROJECT_REL_PATH)

2 Project Porting Rule

This section describes how to use OEM services and Override folders.

2.1How to modify Kernel/Chipset code

If your project modifies Kernel/Chipset code, you need to feedback information regarding these modifications to the Kernel/Chipset team. Additionally, the modified code must be placed in the $(PROJECT_REL_PATH)/$(PROJECT_PKG)/Override related location. If a component is purely OEM/ODM dependent, it is not necessary to put it into the Override folder.

InsydeH2O 5.4 treats Override folder as temp solution and we use OEM services as customized method in InsydeH2O 5.4. Please feedback this override information to Kernel/Chipset team whether the code is for Kernel or Chipset. After reviewing, Kernel/Chipset will offer one new OEM service for your requirement, then you can put your modified code to this OEM service. For more information about InsydeH2O 5.4 OEM services, please refer to "InsydeH2O 5.4 Alder Lake OEM Chipset Services Technical Reference".

Clarification: OEM project engineers may face time constraints to produce code that functions well. In order to save time, engineers can simply override Kernel/Chipset code to indicate that there is no way for that code to serve their needs

2.2Modify Intel reference code and Feedback

Example: If you need to modify **CpuInitDxe.c** of CPU reference code.

CpuInitDxe driver is defined in

`$(CHIPSET_REL_PATH)/$(PLATFORM_SI_PACKAGE)/Product/AlderLake /SiPkgDxe.dsc` as below:

`$(PLATFORM_SI_PACKAGE)/Cpu/CpuInit/Dxe/CpuInitDxe.inf`

Please follow these steps to override:

1. Copy **CpuInitDxe.c** from $(CHIPSET_REL_PATH)/$(PLATFORM_SI_PACKAGE)/Cpu/CpuInit/Dxe to $(PROJECT_REL_PATH)/$(PROJECT_PKG) /Override/$(PLATFORM_SI_PACKAGE)/Cpu/CpuInit/Dxe

2. Add the following code into $(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc

```
!disable $(PLATFORM_SI_PACKAGE)/Cpu/CpuInit/Dxe/CpuInitDxe.inf

$(PLATFORM_SI_PACKAGE)/Cpu/CpuInit/Dxe/CpuInitDxe.inf {

  <SOURCE_OVERRIDE_PATH>

    $(PROJECT_PKG)/Override/$(PLATFORM_SI_PACKAGE)/Cpu/CpuInit/Dxe

}
```

Note: Please feedback this override to Kernel/Chipset, then they will review this code and decide whether to offer new OEM service or not.

3 Project Build

The CRB binary is supposed to build from $(PROJECT_REL_PATH)/$(PROJECT_PKG).

AlderLake CRB BoardNameBoardPkg is AlderLakeSMultiBoardPkg.

For Alder Lake:

$(CHIPSET_REL_PATH)/$(CHIPSET_PKG) means Intel/AlderLake/AlderLakeChipsetPkg.

$(PROJECT_REL_PATH) means "Board/Intel" on CRB, and "Board/Xyz" on project.

$(PROJECT_PKG) means AlderLakeSMultiBoardPkg on CRB, and BoardNameBoardPkg on project.

3.1 Separate schematics specific design into BoardNameBoardPkg

GPIO settings

IRQ routings

Flashmap, Ihisi protected area, etc.

3.2 OEM should copy the CRB BoardNameBoardPkg as a reference

1. After creating your project package, modify the `ProjectSetup.bat` to add the $(PROJECT_REL_PATH) for your project. Ex:

   ```
   set PACKAGES_PATH=^

   %WORKSPACE%\Board\Xyz;^

   %WORKSPACE%\Intel\AlderLake;^
   ```

2. Modify `ProjectSetup.bat` to set as project build (CrbBuild=NO).

   ```
   set CrbBuild=NO
   ```

3. Chipset code placed in Intel/AlderLake/AlderLakeChipsetPkg and Intel/AlderLake/ClientOneSiliconPkg is maintained by Chipset team. OEM can use PCDs or OEM services to do customization.

   Insyde provides two kinds of OEM services including OemSvcKernelLib and OemSvcChipsetLib.

   Chipset Lib can be found in:

   ```
   $(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Include/Library/BaseOemSvcChipsetLib.h

   $(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Include/Library/PeiOemSvcChipsetLib.h

   $(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Include/Library/DxeOemSvcChipsetLib.h

   $(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Include/Library/SecOemSvcChipsetLib.h
   ```

```
$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Include/Library/SmmOemSvcChipsetLib.h
```

Kernel Lib can be found in:

```
Insyde/InsydeOemServicesPkg/Include/Library/BaseOemSvcKernelLib.h

Insyde/InsydeOemServicesPkg/Include/Library/PeiOemSvcKernelLib.h

Insyde/InsydeOemServicesPkg/Include/Library/DxeOemSvcKernelLib.h

Insyde/InsydeOemServicesPkg/Include/Library/SmmOemSvcKernelLib.h
```

4. Build your project in $(PROJECT_REL_PATH)/$(PROJECT_PKG).
5. Alder Lake features a full clock that is used as H/W default. If it needs to change clock gen, you may use the following method: Use ME kit porting clock value.

4 Program GPIO Pins

This section describes how to modify the GPIO settings according to the platform control hub and OEM specifications.

4.1Introduction

InsydeH2O 5.4 offers three OEM services "OemSvcEarlyGpioSettingTable", "OemSvcModifyGpioSettingTablePreMem" and "OemSvcModifyGpioSettingTable" for customers to change their GPIO setting before programming GPIO.

Customers can modify or replace the GPIO table by OEM service hooks, "OemSvcEarlyGpioSettingTable", "OemSvcModifyGpioSettingTablePreMem" and "OemSvcModifyGpioSettingTable", and control the procedure functions in Chipset layer by return status.

4.2Related files

$(CHIPSET_REL_PATH)/AlderLakeBoardPkg/AlderLakeMBoards/Library/BoardInitLib/Pei/PeiInitPostMemLib.c

$(CHIPSET_REL_PATH)/AlderLakeBoardPkg/AlderLakeMBoards/Library/BoardInitLib/Pei/PeiInitPreMemLib.c

$(CHIPSET_REL_PATH)/AlderLakeBoardPkg/AlderLakePBoards/Library/BoardInitLib/Pei/PeiInitPostMemLib.c

$(CHIPSET_REL_PATH)/AlderLakeBoardPkg/AlderLakePBoards/Library/BoardInitLib/Pei/PeiInitPreMemLib.c

$(CHIPSET_REL_PATH)/AlderLakeBoardPkg/AlderLakeSBoards/Library/BoardInitLib/Pei/PeiInitPostMemLib.c

$(CHIPSET_REL_PATH)/AlderLakeBoardPkg/AlderLakeSBoards/Library/BoardInitLib/Pei/PeiInitPreMemLib.c

$(CHIPSET_REL_PATH)/AlderLakeBoardPkg/AlderLakeUBoards/Library/BoardInitLib/Pei/PeiInitPostMemLib.c

$(CHIPSET_REL_PATH)/AlderLakeBoardPkg/AlderLakeUBoards/Library/BoardInitLib/Pei/PeiInitPreMemLib.c

$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Library/PeiOemSvcChipsetLib/OemSvcEarlyGpioSettingTable.c

$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Library/PeiOemSvcChipsetLib/OemSvcModifyGpioSettingTable.c

$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Library/PeiOemSvcChipsetLib/OemSvcModifyGpioSettingTablePreMem.c

4.3Setting

InsydeH2O 5.4 offers three OEM services “OemSvcEarlyGpioSettingTable”, "OemSvcModifyGpioSettingTablePreMem" and “OemSvcModifyGpioSettingTable” for customers to change their GPIO setting before programming GPIO. For more information, please refer "InsydeH2O 5.4 Alder Lake OEM Chipset Services Technical Reference".

5 Set PlatformHardwareSwitch Value

5.1Related files

Insyde/InsydeModulePkg/InsydeModulePkg.dec

$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/ChipsetSvcPei/InitPortConfig.c

$(CHIPSET_REL_PATH)/$(PLATFORMSAMPLE_PACKAGE/Library/PeiPolicyUpdateLib/PeiPchPolicyUpdate.c

5.2Setting

InsydeH2O 5.4 uses the following PCDs instead of the OEM service "OemSvcChipsetSetPlatformHardwareSwitch" to provides the interface for customer to enable/disable USB2/USB3/SATA/PCIe port.

gInsydeTokenSpaceGuid.PcdH2OChipsetUsbPortEnable
gInsydeTokenSpaceGuid.PcdH2OChipsetUsb3PortEnable
gInsydeTokenSpaceGuid.PcdH2OChipsetSataPortEnable
gInsydeTokenSpaceGuid.PcdH2OChipsetPciePortEnable

Customer could modify the value of the PCD based project configuration via the Check Point: **gH2OPeiCpInitChipsetPolicyGuid**.

In the **InitPortConfig.c**, we can use the InitPortConfigPcds function to define the SATA, USB and PCIE device default values. InitPortConfigPcds function will get SCU settings, then use these settings to "logical AND" with device default values.

Note: If the device is set to enable in InitPortConfigPcds and SCU, then the device will be activated. If the device is set to disable in InitPortConfigPcds or SCU, then the device will be inactivated

The old OemSvcSetPlatformHardwareSwitch() is removed from InsydeH2O 5.4.

6 Add CPU Microcode

Microcode update consist of a processor-supplied binary that contains a descriptive header and data. The microcode update is composed of a multi-byte header, followed by the encoded data, and then followed by an optional extended signature table.

6.1Introduction

OEM teams can follow below steps to complete microcode integration:

1. Place the new Microcode.txt or Microcode.inc or Microcode.mcb file in $(PROJECT_REL_PATH)/$(PROJECT_PKG)/Binary/MicrocodeUpdates
2. Modify $(PROJECT_REL_PATH)/$(PROJECT_PKG)/Binary/MicrocodeUpdates/MicrocodeUpdates.inf

```
[Sources]

  M_xx_xxxxx_xxxxxxxx.inc  (add new one on here)
```

6.2Related files

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Binary/MicrocodeUpdates/MicrocodeUpdates.inf
```

6.3Setting

1. MicrocodeUpdates.inf

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Binary/MicrocodeUpdates/MicrocodeUpdates.inf

```
…
[Sources]
    M_xx_xxxxx_xxxxxxxx.inc  (add new one on here)
```

7 Programming HDAudio with Verb Table

7.1 Introduction

This section describes how to program HDA with Verb Table.

InsydeH2O 5.4 has an OEM service "OemSvcGetVerbTable" to provides the interface for customer to customize the Verb Table.

7.2 Related files

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/PeiOemSvcKernelLib/OemSvcGetVerbTable.c
```

7.3 Setting

1. OemSvcGetVerbTable.c

OemSvcGetVerbTable/Library/PeiOemSvcKernelLib/OemSvcGetVerbTable.c

```
…
//
// To define Verb Table ID.
//
#if 0
  // Sample code for VerbTable
  // HdaVerbTableAlc700
#define OEM_VERB_TABLE_ID_1         1
  //
  //  VerbTable: (Realtek ALC700) CNL RVP
  //  Revision ID = 0xff
  //  Codec Verb Table for CNL PCH boards
  //  Codec Address: CAd value (0/1/2)
  //  Codec Vendor: 0x10EC0700
#define OEM_VERB_TABLE_1_HEADER1     0x10EC0700, \
                                     0x00000000, \
                                     0xFF,       \
                                     0x01,       \
                                     161,        \
                                     0x0

#define OEM_VERB_TABLE_1_DATA1  0x001720F2, \
                                0x00172110, \
                                0x001722EC, \
                                0x00172310, \
…
```

Warning: The header includes many characteristics of an audio codec. Currently, only 2 fields of the

header are examined to identify an audio codec:

1. Vendor ID / Device ID
2. Revision

If Revision is set to 0xFF, it means the revision will not be examined to identify an audio codec, but the Vendor ID / Device ID will still be examined.

8 Modify PCI IRQ Routing

This section describes how to program PCI IRQ routing.

8.1 Related files

`$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Package.dsc`

`$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc`

`$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Acpi/AcpiTables/Dsdt/PciTree.asl`

`$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Acpi/AcpiTables/Dsdt/BusPRT/BxxPRT.asl`

8.2 Setting

1. Modify PCDs.

   The following PCDs defined in `$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Package.dsc` are used to control the PIRQ settings.

   PcdControllerDeviceIrqRoutingEntry:

   It is used to define controller device IRQ routing information.

   PcdVirtualBusTable:

   The first value is the Bridge Bus number, the second value is the Bridge Device number, the third value is the Bridge Function number, and the fourth value is the virtual bus number. The data array defines the subordinate PCI bus number virtually in order to define the routing table before PCI bus driver assigns subordinate bus numbers.

   PcdIrqPoolTable:

   It is used to set the IRQ resource, where you can give free ISA IRQs to be used by PCI devices.

   PcdPirqPriorityTable:

   It is used to rise up the priority of particular pins.

   If OEMs want to modify the settings, copy these PCDs to `$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc` and modify the value. Chipset CRB setting is located at `$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Package.dsc`, it could be a sample for reference.

`$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Package.dsc`

`[PcdsDynamicEx]`
`…`
`gChipsetPkgTokenSpaceGuid.PcdControllerDeviceIrqRoutingEntry|{ \`
`0x00, 0xF8, 0x60, UINT16(0xDEB8), 0x61, UINT16(0xDEB8), 0x62, UINT16(0xDEB8), 0x63,`

```
  UINT16(0xDEB8), 0x00, 0x00, UINT32(0x00000000), 0x00,  \  #D31
  0x00, 0xF0, 0x68, UINT16(0xDEB8), 0x69, UINT16(0xDEB8), 0x6A, UINT16(0xDEB8), 0x6B,
  UINT16(0xDEB8), 0x00, 0x00, UINT32(0x00000000), 0x00,  \  #D30
  …
  }


  gChipsetPkgTokenSpaceGuid.PcdVirtualBusTable|{ \
  …
  0x00, 0x1c, 0x00, 0x04, \  # PCIE Root Port 1
  0x00, 0x1c, 0x01, 0x05, \  # PCIE Root Port 2
  …
  }

  gChipsetPkgTokenSpaceGuid.PcdIrqPoolTable|{…, 11, 0x00, 10, 0x00, …}
  gChipsetPkgTokenSpaceGuid.PcdPirqPriorityTable|{…, 0, 0, …}
  …
```

2. Modify related ACPI table.

   `$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc`

   A. Before modifying BxxPRT.asl, PciTree.asl, please copy all the files in AcpiTables.inf module from $(CHIPSET_REL_PATH)/AlderLakeBoardPkg/Acpi/AcpiTables/ to $(PROJECT_REL_PATH)/$(PROJECT_PKG) /Acpi/AcpiTables/

   B. Use `!disable $(PLATFORM_BOARD_PACKAGE)/Acpi/AcpiTables/AcpiTables.inf` and re-define your ACPI table module in `$(PROJECT_PKG)/Project.dsc`

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc
[Components.X64]
…
  !disable $(PLATFORM_BOARD_PACKAGE)/Acpi/AcpiTables/AcpiTables.inf
  $(PROJECT_PKG)/Acpi/AcpiTables/AcpiTables.inf
…
```

   C. Modify BxxPRT.asl and PciTree.asl

      H2OIDE can get the figure of all buses residing in the system and then generate corresponding BxxPRT.asl in order to meet ACPI specification. These files define the _PRT objects under their corresponding bus. Include the ASL files generated by H2OIDE under $(PROJECT_REL_PATH)/$(PROJECT_PKG)/Acpi/AcpiTables/Dsdt/BusPRT/ into the PciTree.asl.

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Acpi/AcpiTables/Dsdt/PciTree.asl
…
Scope(\_SB) {
// _PRT - BUS 0x00 - BEGIN - TOOL GENERATED. DO NOT MODIFY.
  include ("Dsdt/BusPRT\\B00PRT.asl")
// _PRT - BUS 0x00 - END - TOOL GENERATED. DO NOT MODIFY.
// _PRT - BUS 0x04 - BEGIN - TOOL GENERATED. DO NOT MODIFY.
  include ("Dsdt/BusPRT\\B04PRT.asl")
// _PRT - BUS 0x04 - END - TOOL GENERATED. DO NOT MODIFY.
// _PRT - BUS 0x05 - BEGIN - TOOL GENERATED. DO NOT MODIFY.
  include ("Dsdt/BusPRT\\B05PRT.asl")
```

```
// _PRT - BUS 0x05 - END - TOOL GENERATED. DO NOT MODIFY.
// _PRT - BUS 0x06 - BEGIN - TOOL GENERATED. DO NOT MODIFY.
  include ("Dsdt/BusPRT\\B06PRT.asl")
// _PRT - BUS 0x06 - END - TOOL GENERATED. DO NOT MODIFY.
…
} // end _SB scope
```

```
$(PROJECT_REL_PATH)/$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Acpi/AcpiTables/Dsdt/BusPRT/BxxPRT.asl
```

```
  Name(PR02, Package(){
    Package(){0x0000FFFF, 0, LNKA, 0 },
    Package(){0x0000FFFF, 1, LNKB, 0 },
    Package(){0x0000FFFF, 2, LNKC, 0 },
    Package(){0x0000FFFF, 3, LNKD, 0 },
  })
  Name(AR02, Package(){
// P.E.G. Port Slot x16
    Package(){0x0000FFFF, 0, 0, 16 },
    Package(){0x0000FFFF, 1, 0, 17 },
    Package(){0x0000FFFF, 2, 0, 18 },
    Package(){0x0000FFFF, 3, 0, 19 },
  })
```

3. GetRoutingTable.c

The `mVirtualBusTablePtr[]` is used in "ModifyIrqTable" function which will modify the IrqRoutingPtr to reflect the correct bus. The "CheckIRQ" function is used to check the IRQ setting of PCI slot in "Setup" value and record it. The first value is the Bridge Bus number, the second value is the Bridge Device number, the third value is the Bridge Function number, and the fourth value is the virtual bus number. The `mVirtualBusTablePtr[]` data array defines the subordinate PCI bus number virtually in order to define the routing table before PCI bus driver assigns subordinate bus numbers.

`Insyde/InsydeModulePkg/Include/Protocol/LegacyBiosPlatform.h`

```
EFI_LEGACY_IRQ_ROUTING_ENTRY
```

```
typedef struct {
  ///
  /// PCI bus of the entry.
  ///
  UINT8                     Bus;
  ///
  /// PCI device of this entry.
  ///
  UINT8                     Device;
  ///
  /// An IBV value and IRQ mask for PIRQ pins A through D.
  ///
  EFI_LEGACY_PIRQ_ENTRY PirqEntry[4];
  ///
  /// If nonzero, the slot number assigned by the board manufacturer.
  ///
  UINT8                     Slot;
  ///
  /// Reserved for future use.
  ///
  UINT8                     Reserved;
} EFI_LEGACY_IRQ_ROUTING_ENTRY;
```

```
EFI_LEGACY_PIRQ_ENTRY
typedef struct {
  ///
  /// If nonzero, a value assigned by the IBV.
  ///
  UINT8   Pirq;
  ///
  /// If nonzero, the IRQs that can be assigned to this device.
  ///
  UINT16  IrqMask;
} EFI_LEGACY_PIRQ_ENTRY;
```

```
EFI_LEGACY_IRQ_PRIORITY_TABLE_ENTRY
typedef struct {
  ///
  /// IRQ for this entry.
  ///
  UINT8 Irq;
  ///
  /// Status of this IRQ.
  ///
  /// PCI_UNUSED 0x00. This IRQ has not been assigned to PCI.
  ///
  /// PCI_USED 0xFF. This IRQ has been assigned to PCI.
  ///
  /// LEGACY_USED 0xFE. This IRQ has been used by an SIO legacy
  /// device and cannot be used by PCI.
  ///
  UINT8 Used;
} EFI_LEGACY_IRQ_PRIORITY_TABLE_ENTRY;
```

4. Searching "$PIR" in 0xF0000 ~ 0xFFFFF to find $PIR Table.

```
-s f000:0 ffff '$PIR'
F000:E890
-d f000:e890
F000:E890  24 50 49 52 00 01 A0 00-00 F8 00 00 86 80 10 24   $PIR...........$
F000:E8A0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 83   ................
F000:E8B0  00 E8 60 20 0C 63 20 0C-62 20 0C 6B 20 0C 00 00   ..` .c .b .k ...
F000:E8C0  00 F8 62 20 0C 61 20 0C-00 00 00 00 00 00 00 00   ..b .a .........
F000:E8D0  00 10 60 20 0C 61 20 0C-00 00 00 00 00 00 00 00   ..` .a .........
F000:E8E0  00 00 60 20 0C 61 20 0C-00 00 00 00 00 00 00 00   ..` .a .........
F000:E8F0  01 00 68 20 0C 00 00 00-00 00 00 00 00 00 00 00   ..h ............
F000:E900  01 08 69 20 0C 00 00 00-00 00 00 00 00 00 00 00   ..i ............
```

'$PIR'

8.3How to use SetPIrqPriority

1. Check the hardware design to find out whether the device is connected to a specific bridge. Take the following case as an example:

   A PCIE card is connected to the first PCIE bridge Bus:0, Device:28, Function:1, and it has 2 functions: Function 0(Use INTA), Function 1(Use INTB).

2. In this case, the virtual bus number is 5.

   Check PcdVirtualBusTable in `$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc`

```
gChipsetPkgTokenSpaceGuid.PcdVirtualBusTable|{  0x00, 0x01, 0x01, 0x0A, \
                                                0x00, 0x01, 0x02, 0x0B, \
                                                0x00, 0x1c, 0x00, 0x04, \
                                                0x00, 0x1c, 0x01, 0x05, \
                                                0x00, 0x1c, 0x02, 0x06, \
                                                0x00, 0x1c, 0x03, 0x07, \
                                                0x00, 0x01, 0x00, 0x02} #EndEntry
```

3. You can find the PIRQ link value {PIRQA, PIRQB, PIRQC, PIRQD, PIRQE, PIRQF, PIRQG, PIRQH} in `gChipsetPkgTokenSpaceGuid.PcdPirqLinkValueArray|{0x60, 0x61, 0x62, 0x63, 0x68, 0x69, 0x6A, 0x6B} #EndEntry`

4. Check `$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc`, and make sure the routing entry for the device has been added in PcdControllerDeviceIrqRoutingEntry:

```
0x05, 0x00, 0x69, UINT16(0xDEB8), 0x6A, UINT16(0xDEB8), 0x63, UINT16(0xDEB8),
0x60, UINT16(0xDEB8), 0x00, 0xFF, UINT32(0x00000000), 0x00,  \  #B05
```

   The above setting means: Function 0 use PIRQF, Function 1 use PIRQG. The 11th member of PcdControllerDeviceIrqRoutingEntry is designed for PCI slots from 1~N. If the 12th member of PcdControllerDeviceIrqRoutingEntry is 0xFF means the virtual bus number is used, otherwise left it as 0.

5. If you want to set function 0 to use IRQ9 and function 1 to use shared IRQ3, you can modify PIRQF & PIRQG settings by PcdPirqPriorityTable in `$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc`.

```
gChipsetPkgTokenSpaceGuid.PcdPirqPriorityTable|{ 0,  \# PIRQ A
                                                 0,  \# PIRQ B
                                                 0,  \# PIRQ C
                                                 0,  \# PIRQ D
                                                 0,  \# PIRQ E
                                                 9,  \# PIRQ F
                                                 3,  \# PIRQ G
```

```
0}  #EndEntry
```

9 Adding Super IO

9.1 Introduction

Assume that the SIO name is XXXX in the following description.

1. Create SIO package (SioXXXXPkg)

   Copy a required SIO package and rename as (SioXXXXPkg) into the root folder.

2. Import SIO package

   Modify XXXX according to your SIO chip.

   `!import SioXXXXPkg/Package.dsc` in Project.dsc.

   `!import SioXXXXPkg/Package.fdf` in Project.fdf.

3. Setting the SIO related PCDs for project own requirements

   Add PCDs in Project.dsc and then modify them if required.

   A. Set PcdSioXXXXSetup to create a page of SIO in SCU, its default value is FALSE.

   ```
   gSioGuid.PcdSioXXXXSetup|FALSE
   ```

   B. Set PcdSioXXXXUpdateAsl to patch ASL code for SIO, its default value is TRUE.

   Note that if you are using multiple SIO, it must be TRUE.

   ```
   gSioGuid.PcdSioXXXXUpdateAsl|TRUE
   ```

   C. The variable name storing SIO data for SCU.

   ```
   gSioGuid.PcdSioXXXXSetupStr|L"SioXXXXSetup00"
   ```

   D. Configure the parameters for your SIO chip in `gSioGuid.PcdSioXXXXCfg`.

   ```
   #
   # Device Number: Com:0x01, Floppy:0x02, LPT:0x3, KYBD:0x04, MOUSE:0x05,
   HardWare Monitor:0x10
   #               CIR:0x07
   #
   # TYPEH: SIO ID High Byte
   # TYPEL: SIO ID Low Byte
   # SI:    SIO Instance
   # D:     SIO Device
   ```

```
# DI:    SIO Device Instance

# DE:    SIO Device Enable

# DBASE: SIO Device Base Address

# SIZE: SIO Device Size

# LDN:   SIO Device LDN

# DIRQ:  SIO Device IRQ

# DDMA:  SIO Device DMA

#

gSioGuid.PcdSioIt8728fCfg|{ \

#SIO TYPE | SI   | D    | DI   | DE   | DBASE          | SIZE| LDN | DIRQ| DDMA

#-------------------------------------------------------------------------------
-

0x87, 0x28, 0x00, 0x01, 0x00, 0x01, UINT16(0x03F8), 0x00, 0x00, 0x04, 0x00,
\ # Com1

0x87, 0x28, 0x00, 0x01, 0x01, 0x00, UINT16(0x02F8), 0x00, 0x00, 0x03, 0x00,
\ # Com2

0x87, 0x28, 0x00, 0x02, 0x00, 0x00, UINT16(0x03F0), 0x00, 0x00, 0x06, 0x00,
\ # Floppy

0x87, 0x28, 0x00, 0x03, 0x00, 0x00, UINT16(0x0378), 0x00, 0x00, 0x07, 0x00,
\ # Lpt

0x87, 0x28, 0x00, 0x04, 0x00, 0x01, UINT16(0x0060), 0x00, 0x00, 0x01, 0x00,
\ # KYBD

0x87, 0x28, 0x00, 0x05, 0x01, 0x01, UINT16(0x0060), 0x00, 0x00, 0x0C, 0x00,
\ # MOUSE

0x87, 0x28, 0x00, 0x07, 0x00, 0x00, UINT16(0x0320), 0x01, 0x00, 0x0B, 0x00,
\ # CIR

0x87, 0x28, 0x00, 0x10, 0x00, 0x00, UINT16(0x0290), 0x00, 0x00, 0x00, 0x00,
\ # HWM

0x87, 0x28, 0x00, 0x7F, 0x00, 0x00, UINT16(0x002E), 0x00, 0x00, 0x00, 0x00,
\ # CFG

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, UINT16(0x0000), 0x00, 0x00, 0x00, 0x00
\ # End Entry

}
```

9.2Related files

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc
```

```
SIO package (SioXXXXPkg)/*.*

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.fdf

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Override/SioXXXXPkg/SioXXXXDxe /SioAsl/*.*
```

9.3Setting

1. Project.dsc

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc

```
…
!import SioIt8728fPkg/Package.dsc
…
[PcdsFeatureFlag]
  gSioGuid.PcdSioIt8728fSetup|FALSE
  #
  # If you initialize UpdateAsl variable to FALSE, remembering checked mailbox struct in
SioAsl\$(SioName).asl.
  #
  gSioGuid.PcdSioIt8728fUpdateAsl|TRUE

[PcdsFixedAtBuild]
  #
  # Device Number: Com:0x01, Floppy:0x02, LPT:0x3, KYBD:0x04, MOUSE:0x05, HardWare
Monitor:0x10, CIR:0x07
  #
  # TYPEH: SIO ID High Byte
  # TYPEL: SIO ID Low Byte
  # SI:    SIO Instance
  # D:     SIO Device
  # DI:    SIO Device Instance
  # DE:    SIO Device Enable
  # DBASE: SIO Device Base Address
  # SiIZE: SIO Device Size
  # LDN:   SIO Device LDN
  # DIRQ:  SIO Device IRQ
  # DDMA:  SIO Device DMA
  #
  gSioGuid.PcdSioIt8728fCfg|{ \
    # SIO TYPE | SI | D | DI | DE | DBASE | SIZE | LDN | DIRQ | DDMA
    # ------------------------------------------------------------------------------
    0x87, 0x28, 0x00, 0x01, 0x00, 0x01, UINT16(0x03F8), 0x00, 0x00, 0x04, 0x00,    \ #
Com1
    0x87, 0x28, 0x00, 0x01, 0x01, 0x00, UINT16(0x02F8), 0x00, 0x00, 0x03, 0x00,    \ #
Com2
    0x87, 0x28, 0x00, 0x02, 0x00, 0x00, UINT16(0x03F0), 0x00, 0x00, 0x06, 0x00,    \ #
Floppy
    0x87, 0x28, 0x00, 0x03, 0x00, 0x00, UINT16(0x0378), 0x00, 0x00, 0x07, 0x00,    \ # Lpt
    0x87, 0x28, 0x00, 0x04, 0x00, 0x01, UINT16(0x0060), 0x00, 0x00, 0x01, 0x00,    \ #
KYBD
```

```
    0x87, 0x28, 0x00, 0x05, 0x01, 0x01, UINT16(0x0060), 0x00, 0x00, 0x0C, 0x00,    \ #
MOUSE
    0x87, 0x28, 0x00, 0x07, 0x00, 0x00, UINT16(0x0320), 0x01, 0x00, 0x0B, 0x00,    \ # CIR
    0x87, 0x28, 0x00, 0x10, 0x00, 0x00, UINT16(0x0290), 0x00, 0x00, 0x00, 0x00,    \ # HWM
    0x87, 0x28, 0x00, 0x7F, 0x00, 0x00, UINT16(0x002E), 0x00, 0x00, 0x00, 0x00,    \ # CFG
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, UINT16(0x0000), 0x00, 0x00, 0x00, 0x00     \ # End
Entry
  }
  gSioGuid.PcdSioIt8728fSetupStr|L"SioIt8728fSetup00"
```

2. Project.fdf

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.fdf
```

```
…
!import SioDummyPkg/Package.fdf
…
```

10 PCISkipTable

This section explains where to set a skip table.

10.1 Related files

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/DxeOemSvcKernelLib/OemSvcInstallPciSkipTable.c
```

10.2 Setting

1. OemSvcInstallPciSkipTable.c

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/DxeOemSvcKernelLib/OemSvcInstallPciSkipTable.c
```

```
…
//
// module variables
//
PCI_SKIP_TABLE      mPciSkipTable[] = {
  {
    0xffff,
    0xffff
  }
};
…
```

11 Program SsidSvid

This section provides information on where to set a device's SSID/SVID.

OEM can use a PcdDefaultSsidSvid to update SSID default value.

The SSID/SVID of the devices within the SoC will be programmed in the PEI phase via the Intel FSP. For these devices, we provide the PcdDefaultSsidSvidPeiTable for OEM to update the SSID/SVID values. OEM can also use OemSvcUpdateFspsUpd() to modify SsidTable stored in UPD. Changing SsidTable in Fsp-S UPD can override all previous modification.

For devices which are not within the SoC, we have the OemSvcUpdateSsidSvidInfo() to customize the SSID/SVID values.

11.1 Related files

$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Package.dsc

$(PLATFORMSAMPLE_PACKAGE)/Library/PeiPolicyUpdateLib/PeiSiPolicyUpdate.c

$(CHIPSET_REL_PATH)/Library/PeiOemSvcChipsetLib/OemSvcUpdateFspsUpd.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/DxeOemSvcKernelLib/OemSvcUpdateSsidSvidInfo.c

11.2 Setting

1. Package.dsc.c

$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Package.dsc

```
…
#
# Define the devices which SSID/SVID must be programmed before init.
# If SsidSvid is 0 means use PcdDefaultSsidSvid as default SsidSvid value.
# If SsidSvid is not 0, use the value as device SsidSvid.
#
gChipsetPkgTokenSpaceGuid.PcdDefaultSsidSvidPeiTable|{ \
        #   Bus,    Dev,    Func, SsidSvid,
            0x00, 0x14, 0x00, UINT32(0x00000000), \ # XHCI Controller
            0x00, 0x1F, 0x03, UINT32(0x00000000)}   # HDA Controller
…
```

Add SSID/SVID of devices which you want to update in PEI phase into table.

We suggest that use this way to update devices within the SoC. The devices outside of the SoC will be programmed via DXE phase interface, OemSvcUpdateSsidSvidInfo().

Note: We found some devices' SSID/SVID cannot be programmed in the SiProgramSsid() function provided by Intel RC. We suggest programming these device via DXE phase interface. Ex: iTBT devices, VGA.

In some devices, the SSID/SVID is read only and cannot be programmed. Ex: CNVi, iTBT DMA.

2. PeiSiPolicyUpdate.c

$(PLATFORMSAMPLE_PACKAGE)/Library/PeiPolicyUpdateLib/PeiSiPolicyUpdate.c

```
...
STATIC
VOID
UpdateSsidPolicy (
... )
{
...
    DefaultSsidSvid = PcdGet32 (PcdDefaultSsidSvid);
    UPDATE_POLICY (((FSPS_UPD *)FspsUpd)->FspsConfig.SiCustomizedSvid, SiConfig->CustomizedSvid, (UINT16)(DefaultSsidSvid
& 0xFFFF));
    UPDATE_POLICY (((FSPS_UPD *)FspsUpd)->FspsConfig.SiCustomizedSsid, SiConfig->CustomizedSsid, (UINT16)((DefaultSsidSvid
>> 16) & 0xFFFF));
    CustomizedSsidSvidTable=(SSID_SVID_PEI_CONFIG*)PcdGetPtr(PcdDefaultSsidSvidPeiTable);

...
}
```

PcdDefaultSsidSvid is a default SSID/SVID for all devices within the SoC that didn't defined in PcdDefaultSsidSvidPeiTable.

We pass this PCD to the FspsUpd(in API mode) or SiConfig(in Dispatch mode) and let FSP program SSID/SVID according to it.

If you want to modify some devices, you can use PcdDefaultSsidSvidPeiTable to update.

The program flow is below:

A. Pass all device that in the PcdDefaultSsidSvidPeiTable to SsidTablePtr:

B. SiProgramSsid() will be called by SiInit(), it takes PcdDefaultSsidSvid as SSID/SVID default.

C. SiProgramSsid() will call OverrideSvidSsidValue() to override SSID/SVID with PcdDefaultSsidSvidPeiTable.

D. If the device exist in PcdDefaultSsidSvidPeiTable:

Override SSID/SVID by value in PcdDefaultSsidSvidPeiTable.

else

Use PcdDefaultSsidSvid to program it

3. OemSvcUpdateFspsUpd.c

$(CHIPSET_REL_PATH)/Library/PeiOemSvcChipsetLib/OemSvcUpdateFspsUpd.c

```
...
/**
 This function offers an interface to modify FSPS_UPD data before the FSP-S API be called.

 @param[in,out]     FspUpdDataPtr                A pointer to the UPD data region data
structure address.
```

```
                                                    It must convert to FSPS_UPD structure
for update FSPS UPD.

                                                    If you set this pointer to NULL, FSP API
will use the
                                                    default FSP-S UPD settings in the
binary.

 @retval              EFI_UNSUPPORTED               Returns unsupported by default.
 @retval              EFI_MEDIA_CHANGED             Alter the Configuration Parameter.
 @retval              EFI_SUCCESS                   The function performs the same operation
as caller.
                                                    The caller will skip the specified
behavior and assuming
                                                    that it has been handled completely by
this function.
**/
EFI_STATUS
OemSvcUpdateFspsUpd (
  IN OUT VOID                                       **FspUpdDataPtr
  )
{
  FSPS_UPD                                          *FspsUpd;
  FspsUpd = (FSPS_UPD *)(*FspUpdDataPtr);

  //
  // Set this pointer to NULL, use the default FSP-S UPD settings in the binary.
  // BUGBUG: When set the FspUpdDataPtr to NULL, will cause the system to hang in FSP
  // binary, still can't use the default UPD data in binary to boot successfully.
  //
//  *FspUpdDataPtr = NULL;

  return EFI_UNSUPPORTED; }
```

Note: OemSvcUpdateFspsUpd() will not be called in dispatch mode.

4. OemSvcUpdateSsidSvidInfo.c

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/DxeOemSvcKernelLib/OemSvcUpdateSsidSvidInfo.c
...
/**
  The OemSvc is used to update SSID/SVID value by OEM.

  @param[in]      Bus                   Bus number.
  @param[in]      Dev                   Device number.
  @param[in]      Func                      Function number.
  @param[in]      VendorID              Vendor ID.
  @param[in]      DeviceID              Device ID.
  @param[in]      ClassCode             Class Code.
  @param[in out]  SsidSvid              Pointer to SSID/SVID.
```

```
  @retval     EFI_UNSUPPORTED     Returns unsupported by default.
  @retval     EFI_SUCCESS                 OEM handled SSID/SVID programming on this PCI
device. Skip default kernel programming
                                                                    mechanism.
  @retval     EFI_MEDIA_CHANGED     Updated SsidSvid value and returned this value for
default programming mechanism.
**/
EFI_STATUS
OemSvcUpdateSsidSvidInfo
(
  IN      UINT8     Bus,
  IN      UINT8     Dev,
  IN      UINT8     Func,
  IN      UINT16    VendorId,
  IN      UINT16    DeviceId,
  IN      UINT16    ClassCode,
  IN OUT UINT32   *SsidSvid
  )
{
  /*++
    Todo:
      Add project specific code in here.
  --*/
#if 0
// Sample code for OEM project.
// The code is for BIOS to configure SSID&SSVID for NVIDA/AMD add-in card.
// The address of SSID&SSVID is different with normal PCIE device's address 0x2C.
  UINT8       Index;
  EFI_STATUS  Status;

  Status = EFI_UNSUPPORTED;
  for (Index = 0; SsidTable[Index].SpecialSsidSvidFunction != NULL; Index++ ) {
    if (SsidTable[Index].VendorId == VendorId) {
      if ((SsidTable[Index].DeviceId == DEVICE_ID_DONT_CARE)
        || (SsidTable[Index].DeviceId == DeviceId)){
        Status = SsidTable[Index].SpecialSsidSvidFunction (Bus, Dev, Func, SsidSvid);
        break;
      }
    }
  }

  return Status;
#endif
  return EFI_UNSUPPORTED;
}
```

12 Define LPC/eSPI Interface

This section describes how to check Platform Control Hub, which is essential to the success of porting Insyde BIOS source code.

Although Alder Lake only support eSPI and don't support LPC, we keep the "PchLpcIoEnableDecoding" for compatibility.

12.1 Related files

```
$(CHIPSET_REL_PATH)/$(PLATFORMSAMPLE_PACKAGE)/PlatformPkg.dec

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc
```

12.2 Setting

1. Project.dsc

   If your project setting is different from CRB, please modify
   `gPlatformModuleTokenSpaceGuid.PchLpcIoEnableDecoding`

   If you don't see it on Project.dsc, please copy it from
   `$(CHIPSET_REL_PATH)/$(PLATFORMSAMPLE_PACKAGE)/PlatformPkg.dec`.

`$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc`

```
…
  #
  # PcdPchLpcDecodeRange, used to set LPC Component Decode Range (LPC device offset:80h -
81h)
  #
  # bit 2:0      : COMA Decode Range
  # bit 6:4      : COMB Decode Range
  # Value        : 000     = 3F8h -3FFh (COM1)      001     = 2F8h -2FFh (COM2)
  #              : 010     = 220h -227h             011     = 228h -22Fh
  #              : 100     = 238h -23Fh             101     = 2E8h -2EFh (COM4)
  #              : 110     = 338h -33Fh             111     = 3E8h -3EFh (COM3)
  # bit 9:8       : LPT Decode Range
  # Value        : 00 = 378h-37fh and 778h-77fh     01 = 278h -27Fh (port 279h is ready
only) and 678h -67Fh
  #              : 10 = 3BCh -3BEh and 7BCh -7BEh
  # bit 12         : FDD Decode Range
  # Value         : 0 = 3F0h-3F5h, 3F7h (Primary)     1= 370h -375h, 377h (Secondary)
  gPlatformModuleTokenSpaceGuid.PcdLpcIoDecodeRange|0x0010
  #
  # PchLpcEnableList, used to enable/disable LPC Component (LPC device offset:82h-83h)
  #
  # Bit  0      : COMA_LPC_EN        Bit  1      : COMB_LPC_EN
  # Bit  2      : LPT_LPC_EN          Bit  3      : FDD_LPC_EN
  # Bit  8      : GAMEL_LPC_EN       Bit  9      : GAMEH_LPC_EN
  # Bit 10      : KBC_LPC_EN         Bit 11      : MC_LPC_EN
```

```
  # Bit 12      : CNF1_LPC_EN        Bit 13      : CNF2_LPC_EN
  gPlatformModuleTokenSpaceGuid.PchLpcIoEnableDecoding|0x3c03
…
```

13 Adjust PCI-e MMIO Size Address

This section describes how to adjust the size and the address of PCI-e MMIO.

13.1 Related files

```
$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Package.env
```

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc
```

13.2 Setting

1. Package.env

   To modify PCI_EXPRESS_SIZE of Package.env file as below.

   A. To set PCI_EXPRESS_SIZE = 256 for PCI-e MMIO size is 256Mbytes

   B. To set PCI_EXPRESS_SIZE = 128 for PCI-e MMIO size is 128Mbytes

   C. To set PCI_EXPRESS_SIZE = 64 for PCI-e MMIO size is 64Mbytes

$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Package.env

```
…
#
# 256MB, BAR = 0xE0000000, Max bus = 0xFF
# 128MB, BAR = 0xF0000000, Max bus = 0x7F
# 64MB,  BAR = 0xF0000000, Max bus = 0x3F
#
EDK_GLOBAL  PCI_EXPRESS_SIZE                    = 256
…
```

14 FlashMap

This section describes how to modify the BIOS size of description region.

14.1 Introduction

If BIOS image size does not match the BIOS size of description region, please DO NOT USE null binary image appending BIOS image to meet BIOS size of description region in your project, otherwise it will cause BIOS R/W variable error during POST. We suggest you to follow one of the two scenarios as below:

1. Adjust setting of description region to meet your BIOS image size.
2. Modify Project.fdf to adjust BIOS image size.

14.2 EC share ROM (BIOS and EC in the same flash part)

1. Put project EC binary in this path

   `$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Binary/Insyde/Ec`

2. Modify project EC binary path and filename in Project.fdf

   `FILE = $(PROJECT_PKG)/Binary/Insyde/Ec/EmuEc.bin`

3. Set these switches in Project.env

   Set `EDK_GLOBAL  EC_IDLE_PER_WRITE_BLOCK` to `YES`

   Set `EDK_GLOBAL  EC_SHARED_FLASH_SUPPORT` to `YES`

4. Implement and modify EC related Kernel and Chipset OEM services in

   `$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcKernelLib/*.*`

   `$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcChipsetLib/*.*`

   Follow the steps below to implement your project own BaseOemSvcKernelLib

   A. If the relative files do not exist in `$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcKernelLib` folder, copy BaseOemSvcKernelLib.inf and the *.c files that you want to implement from `Insyde/InsydeOemServicesPkg/Library/BaseOemSvcKernelLib` into `$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcKernelLib`.

   B. Delete the *.c filename (what you do not need) from `[Sources]` section in BaseOemSvcKernelLib.inf.

   C. Add the following description to `[LibraryClasses]` section in Project.dsc.

   ```
   BaseOemSvcKernelLib|$(PROJECT_PKG)/Library/BaseOemSvcKernelLib/BaseOemSvcKernelLib.inf
   ```

D. Modify the source *.c file for your own project.

5. Follow the steps below to implement your project own BaseOemSvcChipsetLib

   A. If the relative files do not exist in `$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcChipsetLib` folder, copy BaseOemSvcChipsetLib.inf and the *.c files that you want to implement from `$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Library/BaseOemSvcChipsetLib` into `$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcChipsetLib`.

   B. Delete the *.c filename (what you do not need) from `[sources]` section in BaseOemSvcChipsetLib.inf

   C. Add the following description to the `[LibraryClasses]` section in Project.dsc.

   ```
   BaseOemSvcChipsetLib|$(PROJECT_PKG)/Library/BaseOemSvcChipsetLib/BaseOemSvcChipsetLib.inf
   ```

Modify the source *.c file for your own project.

14.3 Non EC share ROM

If your project does not support EC then you should follow the steps listed below:

1. Set `EDK_GLOBAL  EC_IDLE_PER_WRITE_BLOCK` to `NO` in Project.env
2. Set `EDK_GLOBAL  EC_SHARED_FLASH_SUPPORT` to `NO` in Project.env
3. Remove SioDummyPkg:

   Remove `!import SioDummyPkg/Package.dsc` from Project.dsc.

   Remove `!import SioDummyPkg/Package.fdf` from Project.fdf.

14.4 Related files

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.env

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.fdf

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcKernelLib/BaseOemSvcKernelLib.inf

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcKernelLib/OemSvcEcAcpiMode.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcKernelLib/OemSvcEcInit.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcChipsetLib/BaseOemSvcChipsetLib.inf

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcChipsetLib/OemSvcEcDetectEcPresent.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcChipsetLib/OemSvcEcGetLidState.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcChipsetLib/OemSvcEcGetPcieDockSta
```

```
tus.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcChipsetLib/OemSvcEcPowerState.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcChipsetLib/OemSvcEcReadEcRam.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcChipsetLib/OemSvcEcSaveRestoreKbc
.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcChipsetLib/OemSvcEcSetCriticalThe
rmal.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcChipsetLib/OemSvcEcSetDswMode.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcChipsetLib/OemSvcEcSetLowPowerMod
e.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcChipsetLib/OemSvcEcVersion.c
```

14.5 Setting

1. Project.env

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.env
…
EDK_GLOBAL  EC_IDLE_PER_WRITE_BLOCK          = NO
EDK_GLOBAL  EC_SHARED_FLASH_SUPPORT          = NO
…
```

2. Project.fdf

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.fdf
…
[Defines]
!if $(FSP_WRAPPER_SUPPORT) == YES
!if gInsydeTokenSpaceGuid.PcdH2OMultiBoardSupported == FALSE
DEFINE FLASH_BASE                                          = 0xFF400000
DEFINE FLASH_SIZE                                          = 0x00C00000
!else
DEFINE FLASH_BASE                                          = 0xFF400000
DEFINE FLASH_SIZE                                          = 0x00C00000
!endif

DEFINE BLOCK_SIZE                                          = 0x00001000
DEFINE NUM_BLOCKS                                          =
$(FLASH_SIZE)/$(BLOCK_SIZE)

DEFINE FLASH_REGION_FVEC_OFFSET                            = 0x00000000
!if $(EC_SHARED_FLASH_SUPPORT) == YES
  DEFINE FLASH_REGION_FVEC_SIZE                            = 0x00010000
!else
  DEFINE FLASH_REGION_FVEC_SIZE                            = 0x00000000
!endif
…
```

Current Flash rom area allocation:

A. FLASH_REGION_FVEC_OFFSET/ FLASH_REGION_FVEC_SIZE

Define the EC binary start address and size for BIOS and EC in the same flash part.

B. FLASH_REGION_FVMAIN_OFFSET/ FLASH_REGION_FVMAIN_SIZE

Define the DXE driver start address and size, all drivers are compressed in this area.

For more details, please refer **Build/$(PROJECT_PKG)/RELEASE_DEVTLSxVC14/FV/DXEFV.FV.txt** and **FVMAIN_COMPACT.FV.txt**.

C. FLASH_REGION_FVUNSIGNED_OFFSET/ FLASH_REGION_FVUNSIGNED_SIZE

Define the UNSIGNEDFV area start address and size.

UNSIGNEDFV area can be found on **$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.fdf [FV.UNSIGNEDFV]**

D. FLASH_REGION_FV_MICROCODE_OFFSET / FLASH_REGION_FV_MICROCODE_SIZE

Define the MicroCode start address and size.

E. FLASH_REGION_NV_COMMON_STORE_SUBREGION_OEM_DMI_STORE_OFFSET/
FLASH_REGION_NV_COMMON_STORE_SUBREGION_OEM_DMI_STORE_SIZE

Define DMI table start address and size.

F. FLASH_REGION_NV_COMMON_STORE_SUBREGION_NV_BVDT_OFFSET/
FLASH_REGION_NV_COMMON_STORE_SUBREGION_NV_BVDT_SIZE

Define BIOS version information start address and size.

G. FLASH_REGION_NV_COMMON_STORE_SUBREGION_NV_MSDM_DATA_OFFSET/
FLASH_REGION_NV_COMMON_STORE_SUBREGION_NV_MSDM_DATA_SIZE

Define MSDM key start address and size.

H. FLASH_REGION_NVSTORAGE_XXX/ FLASH_REGION_NVSTORAGE_XX_SIZE

Define NVSTORAGE start address and size.

I. FLASH_REGION_FV_RECOVERY_OFFSET/FLASH_REGION_FV_RECOVERY_SIZE

Define Recovery start address and size.

PEI driver put in here.

For more details, please refer **Build\$(PROJECT_PKG)\RELEASE_DEVTLSxVC14\FV\RECOVERYFV.FV.txt**.

3. BaseOemSvcKernelLib.inf

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcKernelLib/BaseOemSvcKernelLib.inf

```
…
[Sources]
  OemSvcEcInit.c
```

```
  OemSvcEcAcpiMode.c

[Packages]
  MdePkg/MdePkg.dec
  InsydeModulePkg/InsydeModulePkg.dec
  InsydeOemServicesPkg/InsydeOemServicesPkg.dec
  MdeModulePkg/MdeModulePkg.dec
  $(CHIPSET_PKG)/$(CHIPSET_PKG).dec
  $(PROJECT_PKG)/Project.dec
…
```

4. OemSvcEcInit.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcKernelLib/OemSvcEcInit.c

```
…
EFI_STATUS
OemSvcEcInit (
  )
{
  EFI_STATUS        Status;

  Status = SendEcCommand1 (EC_C_LAN_ON);
  Status = SendEcCommand1 (EC_C_ACPI_DISABLE);
  Status = SendEcCommand1 (EC_C_SMI_NOTIFY_DISABLE);

  return EFI_UNSUPPORTED;
}
…
```

5. OemSvcEcAcpiMode.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcKernelLib/OemSvcEcAcpiMode.c

```
…
EFI_STATUS
OemSvcEcAcpiMode (
  IN  BOOLEAN         Enable
  )
{

  EnableEcAcpiMode (Enable);

  return EFI_UNSUPPORTED;
}…
```

6. OemSvcEcIdle.c

Insyde/InsydeOemServicesPkg/Library/BaseOemSvcKernelLib/OemSvcEcIdle.c

```
…
EFI_STATUS
OemSvcEcIdle (
  IN  BOOLEAN         EnableEcIdle
  )
{
  /*++
```

```
    Todo:
      Add project specific code in here.
  --*/

  return EFI_UNSUPPORTED;
}
…
```

7. OemSvcEcWait.c

Insyde/InsydeOemServicesPkg/Library/BaseOemSvcKernelLib/OemSvcEcWait.c

```
…
EFI_STATUS
OemSvcEcWait (
  IN  BOOLEAN         EnableEcWait
  )
{
  /*++
    Todo:
      Add project specific code in here.
  --*/

  return EFI_UNSUPPORTED;
}
…
```

8. BaseOemSvcChipsetLib.inf

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcChipsetLib/BaseOemSvcChipsetLib.inf

```
…
[Sources]
  OemSvcEcDetectEcPresent.c
  OemSvcEcGetLidState.c
  OemSvcEcGetPcieDockStatus.c
  OemSvcEcPowerState.c
  OemSvcEcReadEcRam.c
  OemSvcEcSaveRestoreKbc.c
  OemSvcEcSetCriticalThermal.c
  OemSvcEcSetDswMode.c
  OemSvcEcSetLowPowerMode.c
  OemSvcEcVersion.c

[Packages]
  MdePkg/MdePkg.dec
  $(CHIPSET_PKG)/$(CHIPSET_PKG).dec
  InsydeModulePkg/InsydeModulePkg.dec
  $(PROJECT_PKG)/Project.dec
  ClientSiliconPkg/ClientSiliconPkg.dec
  $(CHIPSET_REF_CODE_PKG)/SiPkg.dec
  $(PLATFORM_SAMPLE_CODE_DEC_NAME)/$(PLATFORM_SAMPLE_CODE_DEC_NAME).dec

[LibraryClasses]
```

```
  CommonEcLib
  IoLib
  InsydeChipsetEcLib
  DebugLib
…
```

9. OemSvcEcSaveRestoreKbc.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcChipsetLib/OemSvcEcSaveRestoreKbc.c

```
…
EFI_STATUS
OemSvcEcSaveRestoreKbc (
  IN  BOOLEAN                       SaveRestoreFlag
  )
{
  /*++
    Todo:
      Add project specific code in here.
  --*/

  UINT8                             KbdIrqState = 0;
  UINT8                             SaveKbdIrqState = 0;

  if (SaveRestoreFlag) {
    //
    // Restore Keyboard command byte
    //
    IoWrite8 (KEY_CMD_STATE, KBC_WRITE_CMD_BYTE);
    WaitKbcIbe (KEY_CMD_STATE);

    IoWrite8 (KEY_DATA, KbcCmdByte);
    WaitKbcIbe (KEY_CMD_STATE);
  } else {
    //
    // Save Keyboard command byte
    //

    // Disable KBD IRQ
    KbdIrqState = IoRead8 (IRQ_8259_MASK);
    SaveKbdIrqState = KbdIrqState;
    KbdIrqState |= 2;
    IoWrite8 (IRQ_8259_MASK, KbdIrqState);

    WaitKbcIbe (KEY_CMD_STATE);

    // Send read command
    IoWrite8 (KEY_CMD_STATE, KBC_READ_CMD_BYTE);

    WaitKbcObf (KEY_CMD_STATE);

    KbcCmdByte = IoRead8 (KEY_DATA);
```

```
    IoWrite8 (IRQ_8259_MASK, SaveKbdIrqState);
  }

  return EFI_UNSUPPORTED;
}
…
```

10. OemSvcEcPowerState.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcChipsetLib/OemSvcEcPowerState.c

```
…
EFI_STATUS
OemSvcEcPowerState (
  IN OUT  BOOLEAN                         *PowerState
  )
{

  *PowerState = PowerStateIsAc ();

  return EFI_MEDIA_CHANGED;
}
…
```

11. OemSvcEcVersion.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcChipsetLib/OemSvcEcVersion.c

```
…
EFI_STATUS
OemSvcEcVersion (
  OUT EFI_STATUS    *ReadEcVersionStatus,
  OUT UINT8         *MajorNum,
  OUT UINT8         *MinorNum
  )
{
  EFI_STATUS        Status;
  UINT8             EcMajorRevision;
  UINT8             EcMinorRevision;

  Status = SendEcCommand (EC_C_EC_REVISION);
  if (Status == EFI_SUCCESS) {
    Status = ReceiveEcData (&EcMajorRevision);
    if (Status == EFI_SUCCESS) {
      Status = ReceiveEcData (&EcMinorRevision);
      if (Status == EFI_SUCCESS) {
        *ReadEcVersionStatus = Status;
        *MajorNum = EcMajorRevision;
        *MinorNum = EcMinorRevision;
      }
    }
  }

  return EFI_MEDIA_CHANGED;
```

```
}
…
```

12. OemSvcEcSetDswMode.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcChipsetLib/OemSvcEcSetDswMode.c

```
…
EFI_STATUS
OemSvcEcSetDswMode (
  OUT EFI_STATUS    *SetDswModeStatus,
  IN  UINT8         DswMode
  )
{
  *SetDswModeStatus = SendEcCommand (SMC_DEEPSX_CMD);
  if (*SetDswModeStatus == EFI_SUCCESS) {
    switch (DswMode) {
      case PchDpS4S5BatteryEn:
      case PchDpS4S5AlwaysEn:
        *SetDswModeStatus = SendEcData (0x03);
        break;
#ifdef EMBEDDED_FLAG
      case PchDpS3S4S5BatteryEn:
      case PchDpS3S4S5AlwaysEn:
        *SetDswModeStatus = SendEcData (0x05);
        break;
#endif
      case PchDeepSxPolDisable:
      default:
        *SetDswModeStatus = SendEcData (0x00);
        break;
    }
  }

  return EFI_MEDIA_CHANGED; }
…
```

13. OemSvcEcGetLidState.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcChipsetLib/OemSvcEcGetLidState.c

```
…
EFI_STATUS
OemSvcEcGetLidState (
  OUT EFI_STATUS    *EcGetLidState,
  OUT UINT8         *LidIsOpen
  )
{
  UINT8                     PortDataOut;
  UINT8                     DataBuffer[1];
  UINT8                     CmosEcGetLidStateFlag;

  CmosEcGetLidStateFlag = 0;
```

```
  //
  // If the platform does not support a lid, the function must return EFI_UNSUPPORTED
  //
  CmosEcGetLidStateFlag = ReadExtCmos8 (R_XCMOS_INDEX, R_XCMOS_DATA, EcGetLidStateFlag);

  if (((CmosEcGetLidStateFlag & PLATFORM_TYPE_MASK) == PLATFORM_TYPE_TRAD) &&
      ((CmosEcGetLidStateFlag & PLATFORM_FLAVOR_MASK) == PLATFORM_FLAVOR_DESKTOP)) {
    DEBUG ((DEBUG_INFO, "Returning Lid status as unsupported to GOP for DT/AIO board\n"));
    return EFI_UNSUPPORTED;
  }

  if ((CmosEcGetLidStateFlag & EC_PRESENT_MASK) == EC_PRESENT_FLAG) {
    DataBuffer[0] = EC_D_LID_STATE;
    *EcGetLidState = ReadEcRam (DataBuffer);
    if (*EcGetLidState == EFI_SUCCESS) {
      PortDataOut = DataBuffer[0];
      if ((PortDataOut & EC_B_LID_STATUS_OPEN) == EC_B_LID_STATUS_OPEN) {
        *LidIsOpen = LidOpen;
      } else {
        *LidIsOpen = LidClosed;
      }
      return EFI_MEDIA_CHANGED;
    }
  }

  return EFI_UNSUPPORTED;
}
…
```

14. OemSvcEcSetCriticalThermal.c

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcKernelLib/OemSvcEcSetCriticalThermal.c
```

```
…
EFI_STATUS
OemSvcEcSetCriticalThermal (
  IN UINT8  Temperature
  )
{
  EFI_STATUS Status;

  //
  // Get the Critical shutdown temperature and pass it to the EC so that in ACPI mode the
OS will shut the system down.
  //
  Status = SetEcCriticalShutdownTemperature (Temperature);

  return EFI_MEDIA_CHANGED;
}…
```

15. OemSvcEcSetLowPowerMode.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcKernelLib/OemSvcEcSetLowPowerMode.c

```
…
EFI_STATUS
OemSvcEcSetLowPowerMode (
  IN  BOOLEAN       LowPowerMode
  )
{
  UINT8             Data;

  Data = 0;
  if (LowPowerMode) {
    Data |= EC_DEBUG_LOW_POWER_ENABLE;
  }

  SendEcCommand (EC_POWER_FEATURES_CMD);
  SendEcData (Data);

  return EFI_MEDIA_CHANGED;
}
…
```

16. OemSvcEcDetectEcPresent.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcKernelLib/OemSvcEcDetectEcPresent.c

```
…
EFI_STATUS
OemSvcEcDetectEcPresent (
  OUT EFI_STATUS    *CommandStatus,
  OUT BOOLEAN       *Present
  )
{
  /*++
    Todo:
      Add project specific code in here.
  --*/
  RETURN_STATUS     Status;
  UINT8             Retry;
  UINT8             DataBuffer[2];

  //
  // Detect EC Revision
  //
  Retry = 5;
  do {
    Status = DetectEcRevision ((UINT8 *)DataBuffer);
    Retry--;
    if ((RETURN_ERROR (Status)) && (Retry != 0)) {
      MicroSecondDelay (STALL_TIME);
    }
  } while ((RETURN_ERROR (Status)) && (Retry != 0));
```

```
  if (RETURN_ERROR (Status)) {
    *Present = FALSE;
  } else {
    *Present = TRUE;
  }

  *CommandStatus = Status;

  return EFI_MEDIA_CHANGED;
}…
```

17. OemSvcEcGetPcieDockStatus.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcChipsetLib/OemSvcEcGetPcieDockStatus.c

```
…
EFI_STATUS
OemSvcEcGetPcieDockStatus (
  OUT EFI_STATUS    *CommandStatus,
  OUT UINT8         *Dock
  )
{
  UINT8             PortData;

  *CommandStatus = GetPcieDockStatus (&PortData);
  *Dock = PortData;

  return EFI_MEDIA_CHANGED;
}
…
```

18. OemSvcEcReadEcRam.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/BaseOemSvcKernelLib/OemSvcEcReadEcRam.c

```
…
EFI_STATUS
OemSvcEcReadEcRam (
  OUT EFI_STATUS *ReadEcRamStatus,
  IN OUT UINT8   *DataBuffer
  )
{
  UINT8       Data[1];

  *ReadEcRamStatus = ReadEcRam(Data);
  *DataBuffer = Data[0];

  return EFI_MEDIA_CHANGED;
}
…
```

15 HotKey

This section describes how to modify the hotkey to fit in the OEM project feature.

15.1 Related files

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/DxeOemSvcKernelLib/OemSvcInstallPostKeyTable.c
```

15.2 Setting

1. Project.dsc

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc
```

```
…
#
# Hot key Configuration
# Platform Hot key Define
# ScanCode, ShiftKey, AltKey, CtrlKey
# ex:
#    0x54, 0x0, 0x1, 0x0      F1(Combination Key ScanCode) + ShiftKey
#    0x68, 0x0, 0x2, 0x0      F1(Combination Key ScanCode) + AltKey
#    0x5f, 0x0, 0x4, 0x0      F1(Combination Key ScanCode) + CtrlKey
#
gInsydeTokenSpaceGuid.PcdPlatformKeyList|{ \
  0x3b, 0x0, 0x0, 0x0,                     \ # F1_KEY
  0x3c, 0x0, 0x0, 0x0,                     \ # F2_KEY
  0x53, 0x0, 0x0, 0x0,                     \ # DEL_KEY
  0x44, 0x0, 0x0, 0x0,                     \ # F10_KEY
  0x86, 0x0, 0x0, 0x0,                     \ # F12_KEY
  0x01, 0x0, 0x0, 0x0,                     \ # ESC_KEY
  0x40, 0x0, 0x0, 0x0,                     \ # UP_ARROW_KEY_BIT
  0x3d, 0x0, 0x0, 0x0,                     \ # F3_KEY
  0x43, 0x0, 0x0, 0x0,                     \ # F9_KEY
  0x00, 0x0, 0x0, 0x0}                       # EndEntry
…
```

2. OemSvcInstallPostKeyTable.c

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/DxeOemSvcKernelLib/OemSvcInstallPostKeyTable.c
```

```
…
/**
  This OemService provides OEM to define the post key corresponding to the behavior of
utility choosing (SCU or Boot Manager).

  @param[in]    KeyDetected           A bit map of the monitored keys found.
                                      Bit N corresponds to KeyList[N] as provided by the
                                      GetUsbPlatformOptions () API of UsbLegacy protocol.
```

```
  @param[in]    ScanCode              The Scan Code.
  @param[out]   *PostOperation        Point to the operation flag which imply the behavior
of utility choosing in post time. For example: choose the SCU or Boot Manager.

  @retval       EFI_MEDIA_CHANGED     Get post operation success.
  @retval       EFI_SUCCESS           Get post operation failed.
  @retval       Others                Base on OEM design.
**/
EFI_STATUS
OemSvcInstallPostKeyTable (
  IN  UINTN                           KeyDetected,
  IN  UINT16                          ScanCode,
  OUT UINTN                           *PostOperation
  )
{
  UINTN                                 Index;
  UINTN                                 PostOperationCount;
  SCAN_TO_OPERATION                     *PostKeyToOperation;

  InitialPostkeyTable ();
  PostOperationCount = mPostOperationCount;
  PostKeyToOperation = mPostKeyToOperation;

  for (Index = 0; Index < PostOperationCount; Index++) {
    //
    // Search table
    //
    if ((KeyDetected & PostKeyToOperation[Index].KeyBit) ==
PostKeyToOperation[Index].KeyBit &&
         ScanCode == PostKeyToOperation[Index].ScanCode){
      *PostOperation = (UINTN)PostKeyToOperation[Index].PostOperation;
      if (PostKeyToOperation[Index].PostOperation == NO_OPERATION) {
        break;
      }

      return EFI_MEDIA_CHANGED;
    } else if (KeyDetected == 0xffff && ScanCode == PostKeyToOperation[Index].ScanCode) {
      //
      // Pure EFI enviorment, no monitor key driver
      //
      *PostOperation = (UINTN)PostKeyToOperation[Index].PostOperation;
      if (PostKeyToOperation[Index].PostOperation == NO_OPERATION) {
        break;
      }

      return EFI_MEDIA_CHANGED;
    }
  }

  return EFI_SUCCESS;
```

```
}
...
```

16 Modify RC Policy

This section describes how to set the RC policy to fit in the OEM project feature.

16.1 Introduction

The policy drivers configure chipset settings in the Intel Reference Code.

Platform policy is in following location:

```
$(CHIPSET_REL_PATH)/$(PLATFORMSAMPLE_PACKAGE)/PlatformInitAdvanced/PlatformInitAdvancedPei

$(CHIPSET_REL_PATH)/$(PLATFORMSAMPLE_PACKAGE)/Library/PeiPolicyInitLib

$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Library/PeiPolicyBoardConfigLib

$(CHIPSET_REL_PATH)/$(PLATFORMSAMPLE_PACKAGE)/Library/PeiPolicyUpdateLib

$(CHIPSET_REL_PATH)/$(PLATFORMSAMPLE_PACKAGE)/Library/PeiPolicyDebugLib
```

Dxe policy:

```
$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/PlatformInit/PolicyInitDxe

$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Library/DxePolicyBoardConfigLib

$(CHIPSET_REL_PATH)/$(PLATFORMSAMPLE_PACKAGE)/Library/DxePolicyUpdateLib
```

Each policy default value is setting by setup variable or PCD. OEM can modify it by your project requirement.

Each PPI or Protocol have a PEI/DXE OemSvcChipsetLib function for OEM to customize. This function is called before installing the PPI or Protocol.

16.2 Related files

```
$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/PlatformInit/PlatformInitPei/*.*

$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Library/PeiPolicyBoardConfigLib/*.*

$(CHIPSET_REL_PATH)/$(PLATFORMSAMPLE_PACKAGE)/Library/PeiPolicyDebugLib/*.*

$(CHIPSET_REL_PATH)/$(PLATFORMSAMPLE_PACKAGE)/Library/PeiPolicyInitLib/*.*

$(CHIPSET_REL_PATH)/$(PLATFORMSAMPLE_PACKAGE)/Library/PeiPolicyUpdateLib/*.*

$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/PlatformInit/PolicyInitDxe/*.*

$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Library/DxePolicyBoardConfigLib/*.*

$(CHIPSET_REL_PATH)/$(PLATFORMSAMPLE_PACKAGE)/Library/DxePolicyUpdateLib/*.*
```

17 Skip Boot Mode

This section describes how to skip boot mode detection.

17.1 Introduction

How do I set boot mode to always be in "cold boot" mode when I bring up board on my project?

1. Set boot mode as BOOT_WITH_FULL_CONFIGURATION.

   In OemSvcChangeBootMode() routine of OemSvcChangeBootMode.c file

   ```
   *SkipPriorityPolicy == TRUE

   *BootMode = BOOT_WITH_FULL_CONFIGURATION

   Return EFI_SUCCESS.
   ```

2. OemSvcChangeBootMode.c is in below location

   ```
   $(PROJECT_REL_PATH)/$(PROJECT_PKG)/PeiOemSvcKernelLib/OemSvcChangeBootMode.c
   ```

17.2 Related files

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/PeiOemSvcKernelLib/OemSvcChangeBootMode.c
```

17.3 Setting

1. OemSvcChangeBootmode.c

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/PeiOemSvcKernelLib/OemSvcChangeBootMode.c
```

```
…
/**
  This OemService is queried to assign the default boot mode.
  OEM can use this service to assign default boot mode,
  and use the parameter SkipPriorityPolicy to control normal boot mode identification.
  If SkipPriorityPolicy be set as TRUE, the final boot mode is the same as the parameter
"bootmode" which is assigned by this service.

  @param[in, out]  *BootMode             Assign default boot mode.
  @param[in, out]  *SkipPriorityPolicy   If SkipPriorityPolicy == TRUE, then normal boot
mode identification will be skipped.

  @retval          EFI_SUCCESS           Similar to set *SkipPriorityPolicy = TRUE.
  @retval          EFI_UNSUPPORTED       Kernel should go through boot mode policy code
below.
**/
EFI_STATUS
OemSvcChangeBootMode (
```

```
  IN OUT EFI_BOOT_MODE                    *BootMode,
  OUT    BOOLEAN                          *SkipPriorityPolicy
  )
{
  EFI_STATUS                              Status;
  CHIPSET_CONFIGURATION                   SystemConfiguration;
  EFI_PRE_POST_HOTKEY                     PrePostHotkey;
  UINTN                                   VariableSize;
  EFI_GUID                                SystemConfigurationGuid =
SYSTEM_CONFIGURATION_GUID;
  EFI_PEI_READ_ONLY_VARIABLE_PPI          *Variable;

  Status = PeiServicesLocatePpi (
             &gEfiPeiReadOnlyVariablePpiGuid,
             0,
             NULL,
             (VOID **)&Variable
             );
  if (EFI_ERROR (Status)) {
    return EFI_UNSUPPORTED;
  }

  //
  // Change boot mode only if Quick boot is enabled.
  //
  VariableSize = PcdGet32 (PcdSetupConfigSize);
  Status = Variable->PeiGetVariable (
                       (EFI_PEI_SERVICES **)GetPeiServicesTablePointer (),
                       L"Setup",
                       &SystemConfigurationGuid,
                       NULL,
                       &VariableSize,
                       &SystemConfiguration
                       );
  if (EFI_ERROR (Status)) {
    return EFI_UNSUPPORTED;
  }
  if (SystemConfiguration.QuickBoot == 0) {
    return EFI_UNSUPPORTED;
  }

  //
  // Change boot mode if no Pre-POST hotkey pressed
  //
  VariableSize = sizeof (EFI_PRE_POST_HOTKEY);
  Status = Variable->PeiGetVariable (
                       (EFI_PEI_SERVICES **)GetPeiServicesTablePointer (),
                       EFI_PRE_POST_HOTKEY_NAME,
                       &gEfiGenericVariableGuid,
                       NULL,
```

```
                        &VariableSize,
                        &PrePostHotkey
                        );
  if ((Status == EFI_SUCCESS) && (PrePostHotkey.KeyBit != PRE_POST_HOTKEY_NOT_EXIST)) {
    return EFI_UNSUPPORTED;
  }

  if (SystemConfiguration.SetupVariableInvalid == 1) {
    //
    // Setup Variable is invalid, force run in BOOT_WITH_FULL_CONFIGURATION.
    //
    return EFI_UNSUPPORTED;
  }

  //
  // return EFI_SUCCESS only when setting *SkipPriorityPolicy = TRUE
  //

  return EFI_UNSUPPORTED;
}
```

18 Set OEM Global ACPI NVS Region

This section describes how to set OEM global ACPI NVS region detection.

18.1 Introduction

1. Add system variable into **SYSTEM_CONFIGURATION** in SetupConfig.h.
2. If this variable must pass value to the ACPI table, please set it into **OEM_PLATFORM_NVS_AREA** in OemPlatformNvsArea.h and set it into **OGNS** in OemPlatformNvs.asl.

   Note: The order of the variable must have the same sequence between **OEM_PLATFORM_NVS_AREA** and **OGNS**.
3. Set the variable data by SetupVariable or Pcds, or OemSvcUpdatePlatformNvs routine of OemSvcUpdatePlatformNvs.c file.
4. When you finish the setting above, the system will set the global ACPI NVS region successfully.

The definition of OemPlatformNvsArea.h must match OemPlatformNvs.asl

18.2 Related files

$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/ChipsetSvcDxe/UpdateAcpiTable.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Include/OemPlatformNvsArea.h

$(PROJECT_REL_PATH)/AlderLakeBoardPkg/Acpi/AcpiTables/Dsdt/OemPlatformNvs.asl

$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Library/DxeOemSvcChipsetLib/OemSvcUpdatePlatformNvs.c

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Include/SetupConfig.h

18.3 Setting

1. Project.dsc
   A. Before modify OemPlatformNvs.asl, copy all the files in AcpiTables.inf module from **$(PROJECT_REL_PATH)/AlderLakeBoardPkg/Acpi/AcpiTables** to **$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Acpi/AcpiTables**.

   B. Use **!disable** command to disable AcpiTables.inf and re-define your ACPI table module in **$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc**.

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc
… [Components.X64] …

```
  !disable $(PLATFORM_BOARD_PACKAGE)/Acpi/AcpiTables/AcpiTables.inf
  $(PROJECT_PKG)/Acpi/AcpiTables/AcpiTables.inf
…
```

2. UpdateAcpiTable.c

$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/ChipsetSvcDxe/UpdateAcpiTable.c

```
…
/**
  ACPI Platform driver installation function.

  @param[in] ImageHandle     Handle for this drivers loaded image protocol.
  @param[in] SystemTable     EFI system table.

  @retval EFI_SUCCESS        The driver installed without error.
  @retval EFI_ABORTED        The driver encountered an error and could not complete
installation of
                             the ACPI tables.

**/
EFI_STATUS
InstallAcpiPlatform (
//  IN EFI_HANDLE          ImageHandle,
//  IN EFI_SYSTEM_TABLE    *SystemTable
  )
…
```

3. OemSvcUpdatePlatformNvs.c

$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Library/DxeOemSvcChipsetLib/OemSvcUpdatePlatformNvs.c

```
…
/**
 This function provides an interface to update GlobalNvs table content.

 @param[in, out]    mPlatformNvsArea           On entry, points to PLATFORM_NVS_AREA
instance.
                                                                          On exit,
points to updated PLATFORM_NVS_AREA instance.
 @param[in, out]    mOemPlatformNvsArea   On entry, points to OEM_PLATFORM_NVS_AREA
instance.
                                                                           On exit,
points to updated OEM_PLATFORM_NVS_AREA instance.

 @retval            EFI_UNSUPPORTED     Returns unsupported by default.
 @retval            EFI_MEDIA_CHANGED   Alter the Configuration Parameter.
 @retval            EFI_SUCCESS         The function performs the same operation as
caller.
                                        The caller will skip the specified behavior and
assuming
                                        that it has been handled completely by this
function.
*/
```

```
EFI_STATUS
OemSvcUpdatePlatformNvs (
  PLATFORM_NVS_AREA                          * mPlatformNvsArea,
  OEM_PLATFORM_NVS_AREA                   * mOemPlatformNvsArea
  ) {
/*++
  Todo:
  Add project specific code in here.
--*/

  return EFI_UNSUPPORTED;
}
```

19 Update OemTable ID of ACPI tables

This section describes how to update OemTable ID of ACPI tables.

We use the Check Point mechanism to customize the OemTable ID, if you want to use this mechanism, please set "`gInsydeTokenSpaceGuid.PcdH2OBdsCpUpdateAcpiDescHdrSupported`" to `TRUE`.

We will use the "`gEfiMdeModulePkgTokenSpaceGuid.PcdAcpiDefaultOemId`" and "`gEfiMdeModulePkgTokenSpaceGuid.PcdAcpiDefaultOemTableId`" to update.

You can change these PCDs in the Project.dsc.

If you want to set different values for different tables, you can register your Check Point.

19.1 Related files

`Insyde/InsydeModulePkg/Universal/Acpi/AcpiPlatformDxe/AcpiDescriptionHeader.c`

`$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/OemAcpiPlatformDxe/UpdateOemTableID.c`

`$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc`

19.2 Setting

1. AcpiDescriptionHeader.c

   `UpdateAcpiDescriptionHeader()` will trigger the Check Point when "`gInsydeTokenSpaceGuid.PcdH2OBdsCpUpdateAcpiDescHdrSupported`" is `TRUE`

`Insyde/InsydeModulePkg/Universal/Acpi/AcpiPlatformDxe/AcpiPlatform.c`

```
AcpiPlatformEntryPoint () {
  …
  if (PcdGetBool(PcdH2OBdsCpUpdateAcpiDescHdrSupported)) {
    EFI_EVENT                       Event;

    Status = EfiCreateEventReadyToBootEx (
               TPL_CALLBACK-1,
               UpdateAcpiDescriptionHeader,
               NULL,
               &Event
               );
    ASSERT_EFI_ERROR (Status);
  }
  …
}
```

`Insyde/InsydeModulePkg/Universal/Acpi/AcpiPlatformDxe/AcpiDescriptionHeader.c`

```
UpdateAcpiDescriptionHeader () {
  …
    H2O_BDS_CP_UPDATE_ACPI_DESC_HDR_DATA  BdsCpUpdateAcpiDescHdrData;
    BdsCpUpdateAcpiDescHdrData.Size = sizeof (H2O_BDS_CP_UPDATE_ACPI_DESC_HDR_DATA);
    BdsCpUpdateAcpiDescHdrData.Status = H2O_BDS_TASK_NORMAL;
    BdsCpUpdateAcpiDescHdrData.AcpiTableHeader = Table;
    DEBUG ((DEBUG_INFO, "Checkpoint Trigger: %g\n", &gH2OBdsCpUpdateAcpiDescHdrGuid));
    H2OCpTrigger (&gH2OBdsCpUpdateAcpiDescHdrGuid, &BdsCpUpdateAcpiDescHdrData);
    DEBUG ((DEBUG_INFO, "Checkpoint Result: %x\n", BdsCpUpdateAcpiDescHdrData.Status));
    if (BdsCpUpdateAcpiDescHdrData.Status == H2O_CP_TASK_SKIP) {
      AcpiSdtInfo[Index].Skip = TRUE;
    }
  …
}
```

2. UpdateOemTableID.c

   We will register UpdateAcpiDescHdrCallback in `OemUpdateOemTableID()` if "`gInsydeTokenSpaceGuid.PcdH2OBdsCpUpdateAcpiDescHdrSupported`" is `TRUE`.

   The callback routine will use "`gEfiMdeModulePkgTokenSpaceGuid.PcdAcpiDefaultOemId`" and "`gEfiMdeModulePkgTokenSpaceGuid.PcdAcpiDefaultOemTableId`"

`$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/OemAcpiPlatformDxe/UpdateOemTableID.c`

```
…
EFI_STATUS
OemUpdateOemTableID (
  VOID
  )
{
  EFI_STATUS  Status;

  Status = EFI_SUCCESS;
  if (PcdGetBool (PcdH2OBdsCpUpdateAcpiDescHdrSupported)) {
    H2O_CP_HANDLE   CpHandle;

    Status = H2OCpRegisterHandler (
               &gH2OBdsCpUpdateAcpiDescHdrGuid,
               UpdateAcpiDescHdrCallback,
               H2O_CP_MEDIUM_HIGH,
               &CpHandle
               );
    if (EFI_ERROR (Status)) {
      DEBUG ((EFI_D_ERROR, "Checkpoint Register Fail: %g (%r)\n",
&gH2OBdsCpUpdateAcpiDescHdrGuid, Status));
      return Status;
    }
    DEBUG ((EFI_D_INFO, "Checkpoint Registered: %g (%r)\n",
&gH2OBdsCpUpdateAcpiDescHdrGuid, Status));
  }
  return EFI_SUCCESS;
}
```

```
UpdateAcpiDescHdrCallback () {
…
  AcpiDefaultOemId = (CHAR8 *) PcdGetPtr (PcdAcpiDefaultOemId);
…
  AcpiDefaultOemTableId = PcdGet64 (PcdAcpiDefaultOemTableId);
…
}
…
```

20 OEMBadgingSupport

This section describes how to add an OEM logo during POST

20.1 Introduction

1. The logo `BadgingData` and `OemVidoeModeScreenStringData` structure is defined in OEMBadgingSupportDxe.c. The OemBadgingString structure is defined in OEMBadgingString.c
2. InsydeH2O 5.4 offers the DxeOemSvcChipsetLib for OEMs to update this data.
   A. Copy `$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Library/DxeOemSvcChipsetLib/OemSvcUpdateOemBadgingLogoData.c` to `$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/DxeOemSvcChipsetLib`
   B. Put OemSvcUpdateOemBadgingLogoData.c into `[Sources]` section in `$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/DxeOemSvcChipsetLib/DxeOemSvcChipsetLib.inf`
3. New string is put on `$(PROJECT_REL_PATH)/$(PROJECT_PKG)/project.uni`

20.2 Related files

`$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/DxeOemSvcChipsetLib/DxeOemSvcChipsetLib.inf`

`$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/DxeOemSvcChipsetLib/OemSvcUpdateOemBadgingLogoData.c`

`$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.uni`

20.3 Setting

1. DxeOemSvcChipsetLib.inf

`$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/DxeOemSvcChipsetLib/DxeOemSvcChipsetLib.inf`

```
…
[Sources]
  OemSvcModifyAcpiDescriptionHeader.c
  OemSvcSetIgdOpRegion.c
  OemSvcHookPlatformDxe.c
  OemSvcSetUsbLegacyPlatformOptions.c
  OemSvcHookPlatformReset.c
  OemSvcUpdateOemBadgingLogoData.c
…
```

2. OemSvcUpdateOemBadgingLogoData.c

`$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Library/DxeOemSvcChipsetLib/OemSvcUpdateOemBadgingLogoData.c`

```
…
```

```
#if 0 // Sample Implementation
#include <Uefi.h>
#include <Guid/Pcx.h>

#define EFI_INSYDE_BOOT_BADGING_GUID \
  { \
    0x931F77D1, 0x10FE, 0x48bf, 0xAB, 0x72, 0x77, 0x3D, 0x38, 0x9E, 0x3F, 0xAA \
  }

EFI_OEM_BADGING_LOGO_DATA gOemBadgingData[] = {
  //
  // BIOS Vendor Insyde Badge
  //
  { EFI_INSYDE_BOOT_BADGING_GUID,
   EfiBadgingSupportFormatBMP,
   EfiBadgingSupportDisplayAttributeCustomized,
   0,
   0,
   NULL,
   EfiBadgingSupportImageBoot
  },
  { EFI_DEFAULT_PCX_LOGO_GUID,
   EfiBadgingSupportFormatPCX,
   EfiBadgingSupportDisplayAttributeCenter,
   0,
   0,
   NULL,
   EfiBadgingSupportImageLogo
  }
};

OEM_BADGING_STRING gOemBadgingString[] = {
  //
  // OEM can modify the background and foreground color of the OEM badging string through
through the below data
  // for example:
  //   { 50, 280, { 0x00, 0x00, 0x00, 0x00 }, { 0xFF, 0xFF, 0xFF, 0x00 }, STRING_TOKEN
( STR_OEM_BADGING_STR_CPUID ), GetCpuId },
  //     { 0x00, 0x00, 0x00, 0x00 } indicate the foreground color { Blue, Green, Red,
Reserved }
  //     { 0xFF, 0xFF, 0xFF, 0x00 } indicate the background color { Blue, Green, Red,
Reserved }
  //
  { 30, 30, { 0xFF, 0xFF, 0xFF, 0x00 }, { 0x00, 0x00, 0x00, 0x00 }, STRING_TOKEN
( STR_OEM_BADGING_STR_ESC   ), NULL },
  { 50,  0, { 0xFF, 0xFF, 0xFF, 0x00 }, { 0xFF, 0x00, 0x00, 0x00 }, STRING_TOKEN
( STR_OEM_BADGING_STR_ESC   ), NULL }
};
#endif
/**
```

```
 This function provides an interface to modify OEM Logo and POST String.

 @param[in out]     *EFI_OEM_BADGING_LOGO_DATA                    On entry, points to a
structure that specifies image data.
                                                                  On exit , points to
updated structure.
 @param[in out]     *BadgingDataSize    On entry, the size of EFI_OEM_BADGING_LOGO_DATA
matrix.
                                        On exit , the size of updated
EFI_OEM_BADGING_LOGO_DATA matrix.
 @param[in out]     *OemBadgingString   On entry, points to OEM_BADGING_STRING matrix.
                                        On exit , points to updated OEM_BADGING_STRING
matrix.
 @param[in out]     *OemBadgingStringInTextMode                   On entry, points to
OEM_BADGING_STRING matrix in text mode.
                                                                  On exit , points to
updated OEM_BADGING_STRING matrix in text mode.
 @param[in out]     *StringCount        The number is POST string count.
                                        On entry, base on SetupVariable->QuietBoot
                                                  1 : The number of entries in
OemBadgingString,
                                                  0 : The number of entries in
OemBadgingStringInTextMode.
                                        On exit , base on SetupVariable->QuietBoot
                                                  1 : The number of entries in updated
OemBadgingString,
                                                  0 : The number of entries in updated
OemBadgingStringInTextMode.
 @param[in out]     *OemBadgingStringAfterSelectWithMe            On entry, points to
OEM_BADGING_STRING matrix after selected.
                                                                  On exit , points to
updated OEM_BADGING_STRING matrix after selected.
 @param[in out]     *OemBadgingStringAfterSelectWithMeInTextMode On entry, points to
OEM_BADGING_STRINGmatrix after selected  in text mode.
                                                                  On exit , points to
updated OEM_BADGING_STRING matrix after selected  in text mode.

 @retval            EFI_UNSUPPORTED     Returns unsupported by default.
 @retval            EFI_MEDIA_CHANGED   Alter the Configuration Parameter.
 @retval            EFI_SUCCESS         The function performs the same operation as
caller.
                                        The caller will skip the specified behavior and
assuming
                                        that it has been handled completely by this
function.
*/
EFI_STATUS
OemSvcUpdateOemBadgingLogoData (
  IN OUT EFI_OEM_BADGING_LOGO_DATA            **EfiOemBadgingLogoData,
  IN OUT UINTN                                *BadgingDataSize,
```

```
  IN OUT OEM_BADGING_STRING                    **OemBadgingString,
  IN OUT OEM_BADGING_STRING                    **OemBadgingStringInTextMode,
  IN OUT UINTN                                 *StringCount,
  IN OUT OEM_BADGING_STRING                    **OemBadgingStringAfterSelectWithMe,
  IN OUT OEM_BADGING_STRING
**OemBadgingStringAfterSelectWithMeInTextMode
  )
{
/*++
  Todo:
  Add project specific code in here.
--*/
#if 0 // Sample Implementation
  (*EfiOemBadingLogoData) = gOemBadgingData;
  *BadgingDataSize = sizeof (gOemBadgingData) / sizeof (EFI_OEM_BADGING_LOGO_DATA);
  (*OemBadgingString) = gOemBadgingString;
  (*OemBadgingStringInTextMode)[0].X =60;
#endif
  return EFI_UNSUPPORTED;
}
```

3. Project.uni

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.uni
```

```
…
#string STR_OEM_BADGING_STR_F2          #language eng  "Press F2 go to Setup Utility"
...
#string STR_OEM_BADGING_STR_CPUID       #language eng  "CPUID : "
…
```

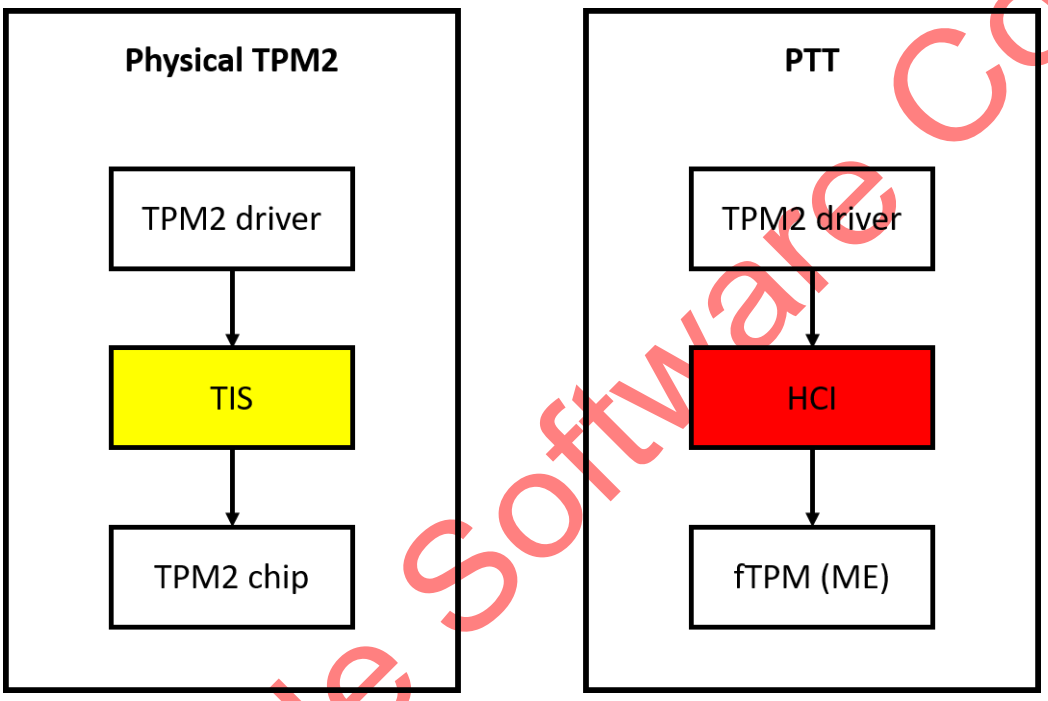
21 Platform Trust Technology Support

This section describes how to enable Platform Trust Technology in the Alder Lake code base.

21.1 Introduction

PTT is fTPM and it is also a subset of TPM2.0.

The BIOS will provide interface (HCI) to communicate with fTPM.



21.2 Related files

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.env
```

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc
```

21.3 Setting

1. Project.env

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.env
…
EDK_GLOBAL   PTT_SUPPORT                                    = YES
…
```

2. Project.dsc

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc
…
#
# TPM2_SUPPORT
#
  gInsydeTokenSpaceGuid.PcdH2OTpm2Supported|TRUE
…
```

3. FW image enable (ME setting):

   There are 3 items for Intel? PTT enabled:

   Platform Protection -> Intel(R) PTT Configuration -> Intel(R) PTT initial power-up state -> Enabled

   Platform Protection -> Intel(R) PTT Configuration -> Intel(R) PTT Supported -> YES

   Platform Protection -> Intel(R) PTT Configuration -> Intel(R) PTT Supported [FPF] -> YES

21.4 Reference

Intel IBP#513196 Host Controller Interface for Intel? Platform Trust Technology v1.2

Intel IBP#544345 Intel? Platform Protection Technology with Platform Trust (Intel? PTT) v1.0

22 BIOS Guard Support

This section describes how to enable BIOS Guard in the Alder Lake codebase, and how to create Bios Guard update image.

22.1 Related files

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.env
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc
$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/$(CHIPSET_PKG).dec
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/PlatformConfig/BiosGuard/BiosGuardSetting.ini
```

22.2 Settings

1. Project.env

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.env
…
EDK_GLOBAL BIOS_GUARD_SUPPORT                    = YES
…
```

2. Project.dsc

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dsc
…
[PcdsFeatureFlag]
.....
gChipsetPkgTokenSpaceGuid.PcdBiosGuardAcmSupport|$(BIOS_GUARD_SUPPORT)
!if gChipsetPkgTokenSpaceGuid.PcdBiosGuardAcmSupport
  gInsydeTokenSpaceGuid.PcdSecureFlashSupported|TRUE
!endif
…
```

3. $(CHIPSET_REL_PATH)/$(CHIPSET_PKG).dec

```
$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/$(CHIPSET_PKG).dec
…
[PcdsFeatureFlag]
  gChipsetPkgTokenSpaceGuid.PcdBiosGuardAcmSupport|FALSE|BOOLEAN|0x40000022
  #
  # Please set PcdBiosGuardEcSupport as TURE if Project use Non-Shared rom EC which
support Bios Guard requested Command.
  # For Bios Guard requested EC Command, please reference Intel Platform Protection
Technology with BIOS Guard Mobile Embedded Controller Design Guide
  #
  gChipsetPkgTokenSpaceGuid.PcdBiosGuardEcSupport|FALSE|BOOLEAN|0x40000023
…
[PcdsFixedAtBuild]
…
  gChipsetPkgTokenSpaceGuid.PcdBiosGuardConfigBgupHeaderPkgAttributes|0x00000000|UIN
  T16|0x20000714
```

```
  gChipsetPkgTokenSpaceGuid.PcdBiosGuardConfigBgpdtHash0|0x55AF29CFF1C7F07F|UINT64|0
  x20000717
  gChipsetPkgTokenSpaceGuid.PcdBiosGuardConfigBgpdtHash1|0x9EB381EAEB8AEDDD|UINT64|0
  x20000718
  gChipsetPkgTokenSpaceGuid.PcdBiosGuardConfigBgpdtHash2|0x1AAD0BF117F8BC91|UINT64|0
  x20000719
  gChipsetPkgTokenSpaceGuid.PcdBiosGuardConfigBgpdtHash3|0x868DE39714FC0A5D|UINT64|0
  x20000720
  gChipsetPkgTokenSpaceGuid.PcdBiosGuardConfigBgupHeaderVendorSpecific|0x808655AA|UI
  NT32|0x20000721
  #
  # Bios Guard prevent back-flashing.
  # The ESRT Version is similar as the BIOS version.
  # If Project.dsc didn't re-defined the PcdBiosGuardConfigBgpdtBiosSvn.
  # Default will reference ESRT Version for Bios Guard to do the back-flashing
compare.
  # PcdBiosGuardConfigBgpdtBiosSvn and BiosSvn in BiosGuardSetting.ini will be
updated autometically in the build time.
  #
  gChipsetPkgTokenSpaceGuid.PcdBiosGuardConfigBgpdtBiosSvn|0x00000000|UINT32|0x20000
  724
  gChipsetPkgTokenSpaceGuid.PcdBiosGuardConfigBgpdtEcCmd|0x66|UINT32|0x20000725
  gChipsetPkgTokenSpaceGuid.PcdBiosGuardConfigBgpdtEcData|0x62|UINT32|0x20000726
  gChipsetPkgTokenSpaceGuid.PcdBiosGuardConfigEcCmdDiscovery|0xB0|UINT8|0x20000727

gChipsetPkgTokenSpaceGuid.PcdBiosGuardConfigEcCmdProvisionEav|0xB1|UINT8|0x20000728
  gChipsetPkgTokenSpaceGuid.PcdBiosGuardConfigEcCmdLock|0xB2|UINT8|0x20000729

gChipsetPkgTokenSpaceGuid.PcdBiosGuardConfigBgpdtEcCmdGetSvn|0xB3|UINT32|0x2000072A
  gChipsetPkgTokenSpaceGuid.PcdBiosGuardConfigBgpdtEcCmdOpen|0xB4|UINT32|0x2000072B
  gChipsetPkgTokenSpaceGuid.PcdBiosGuardConfigBgpdtEcCmdClose|0xB5|UINT32|0x2000072C
  gChipsetPkgTokenSpaceGuid.PcdBiosGuardConfigBgpdtEcCmdPortTest|0xB6|UINT32|0x20000
  72D
  gChipsetPkgTokenSpaceGuid.PcdBiosGuardConfigBgupHeaderEcSvn|0x00000000|UINT32|0x20
  00072E
  gChipsetPkgTokenSpaceGuid.PcdBiosGuardConfigBgpdtPublicKeySlot0|{0x4F, 0xF7, 0x7D,
0x32, 0x56, 0x6C, 0x4C, 0x70, 0x67, 0x44, 0x5B, 0xF3, 0xCA, 0xF7, 0x26, 0x5A, 0x15,
0xD8, 0xF4, 0x3E, 0xAF, 0x5F, 0x97, 0xD6, 0xB8, 0xC0, 0x47, 0x45, 0xDE, 0x72, 0x9E,
0xD5}|VOID*|0x2000072F
  gChipsetPkgTokenSpaceGuid.PcdBiosGuardConfigBgpdtPublicKeySlot1|{0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00}|VOID*|0x20000730
  gChipsetPkgTokenSpaceGuid.PcdBiosGuardConfigBgpdtPublicKeySlot2|{0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00}|VOID*|0x20000731
…
```

4. BiosGuardSetting.ini

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/PlatformConfig/BiosGuardSetting.ini
...
[Platform]
 ;;
 ;; The BIOS Guard module will compare the PLAT_ID found in the BGPDT with the
 ;; PLAT_ID found in the BIOS Guard update package header to ensure a given update
 ;; package is appropriate for the platform. This prevents cross flashing of a platform
 ;; image to the wrong platform when a single platform signing key is used.
 ;;
 PlatId = xxxxx
 SpiSize = 0x02000000
 BiosSize = 0x1000000

[Image]
 ;;
 ;;  FileName : Image file name, (ex.AlderLakeP.fd, BiosGaurdUcodeFv.bin)
 ;;  BiosGuardScriptFile : BIOS Guard Script Language File(Use Intel BIOSGuardSL2Bin.exe, transfer *.bgsl to *.bin)
 ;;
 FileName = XXXX.fd
 BiosGuardScriptFile = BGSL\InsydeBiosGuardScript.bin    (Note2)
 ;;
 ;;  PackageMode:
 ;;     0x1 : Package BIOS Image or Full Image.
 ;;     0x2 : Package EC (Non-Shared flash rom)
 ;;     0x3 : Package Firmware (ex: microcode)          (Note3)
 ;;  PackageMode = 0x1

 ;;
 ;; Used default BIOS Script
 ;;
 ;; Granularity : Size of update data in 1 flash cycle.
 ;; Must set multiple of 0x1000, Maxmum is 0x10000.
 ;; For firmware update, this item will be ignored and use default size 0x10000.
 ;;
 Granularity = 0x00001000

 ;;
 ;;  PackageMode= 0x2, Package EC (Non-Shared Rom)
 ;;
 ;;  EcSize : EC Image file size.
 ;;  EcOffset : SPI rom address of EC firmware.
 ;;
 EcSize = 0x80000
 EcOffset = 0x1000
 ;;
 ;;  PackageMode= 0x3, Package Firmware (ex: microcode full mode only)
 ;;
 ;;  FwSize : Fw size.
 ;;  FwOffset : SPI rom address of Fw firmware.
```

```
  ;;          (Chasmfalls gen2 support microcode update, and default bios size 16M)
  ;;
  FwSize = 0x00080000
  FwOffset = 0x1C80000


[ChasmFalls]
  ;;
  ;; The data in this section will auto be updated in Postbuild Process
  ;;
  ;; ChasmFallsType : Chasm Fall is Disabled , or type is Gen1 or Gen2. (PcdChasmFallsSupport)
  ;;                          (Disable = 0, Gen1 = 0x1, Gen2 = 0x2. ChaseFalls didn't support Full Image, please set type as 0x0.)
  ;; PbbOffset      : PBB BIOS Offset.  (PcdFlashPbbBase  - PcdBiosAreaBaseAddress)
  ;; PbbSize        : PBB/PBBr size.    (PcdFlashPbbSize)
  ;;
  ChasmFallsType = 0x0
  PbbOffset = 0x00000000
  PbbSize = 0x0


[BiosGuardHeader]
  ;;
  ;; Version of the update package header.This field must be 0x0001.
  ;;
  Version = 0x0002

  ;;
  ;; Indicates the PSL major, minor version of the script associated with this update
  ;; package. For this version of the specification this should indicate version 1.0.
  ;;
  PslMajorVer = 0x02
  PslMinorVer = 0x00

  ;;
  ;; If UPDPKG_ATTR[0] == 1, indicates the Security Version Number of the BIOS
  ;; update contained within this update package.The BIOS Guard module must verify
  ;; BGUP.Header.BIOS_SVN >=  BGPDT.BIOS__SVN. if UPDPKG_ATTR[0] == 0, BIOS_SVN checking is not done
  ;; and this field is ignored by the BIOS Guard module.
  ;;
  BiosSvn = 0x0    (Note4)

  ;;
  ;; If UPDPKG_ATTR[1] == 1, indicates the Security Version Number of the EC FW
  ;; update if there is one contained within this update package. If the update
  ;; package contains protected EC operations, but does not contain an EC
  ;; firmware update, the BGUP.Header.EC_SVN >= EC_SVN retrieved from the EC via
  ;; ECCMD_GetSecurityVersionNumber. If UPDPKG_ATTR[1] == 0, EC_SVN checking is
  ;; not done and this files is ignored by the BIOS Guard module.
  ;;
  EcSvn = 0x00000000
```

```
;;
;; 4 bytes available to script writer. This field is not referenced by the BIOS Guard
;; module. It is, however, included in digital signature of signed scripts. It is expected
;; that it will be used for vendor specific versioning or script identification.
;;
VendorSpecific = 0x00
…
```

Note1 : In Build Code Process, BiosSize, FwSize, FwOffset and [Chasm Falls] Section will be auto update by BiosGuardSettingUpdate.exe

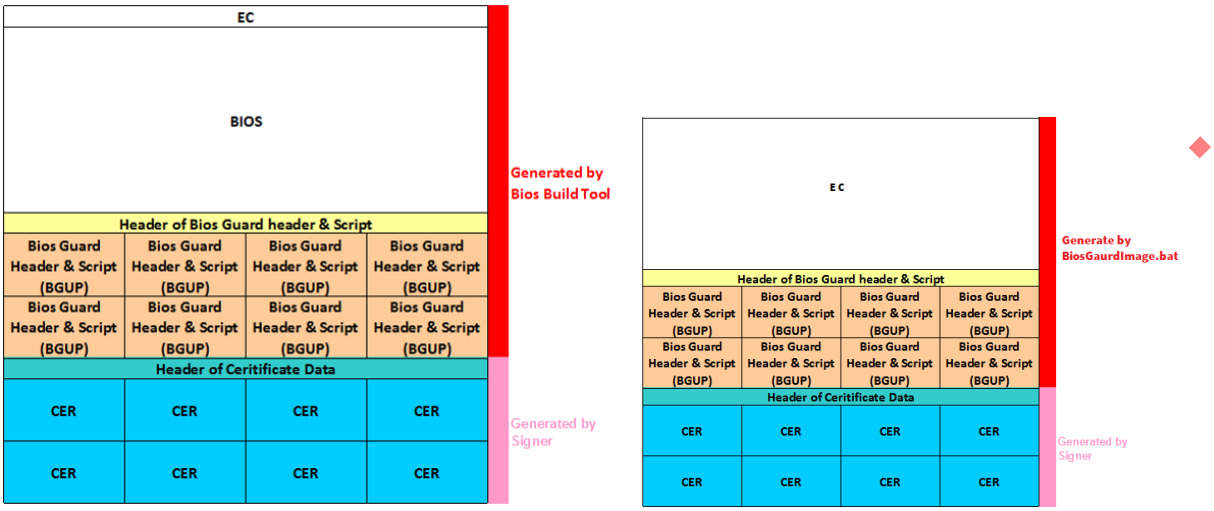
Note2 : Insyde provided 2 BGSL to use, InsdyeBiosGuardRomTranferScript.bin is using for Chasm Falls Gen2 PBB/PBBr sync and rollback. For others request, please still using InsydeBiosGuardScript.bin. And the Binary can be reference *.bgsl file.

Note3 : Package 0x3, Currently only support Chasm Falls Ge2 uCode update to use.

Note4 : Will be auto updated as same as PcdBiosGuardConfigBgpdtBiosSvn. Please make sure the setting of PcdBiosGuardConfigBgpdtBiosSvn is equal or greater than the setting of BiosSvn in BiosGuardSettings.ini.

22.3 Create BIOS Guard Update Image

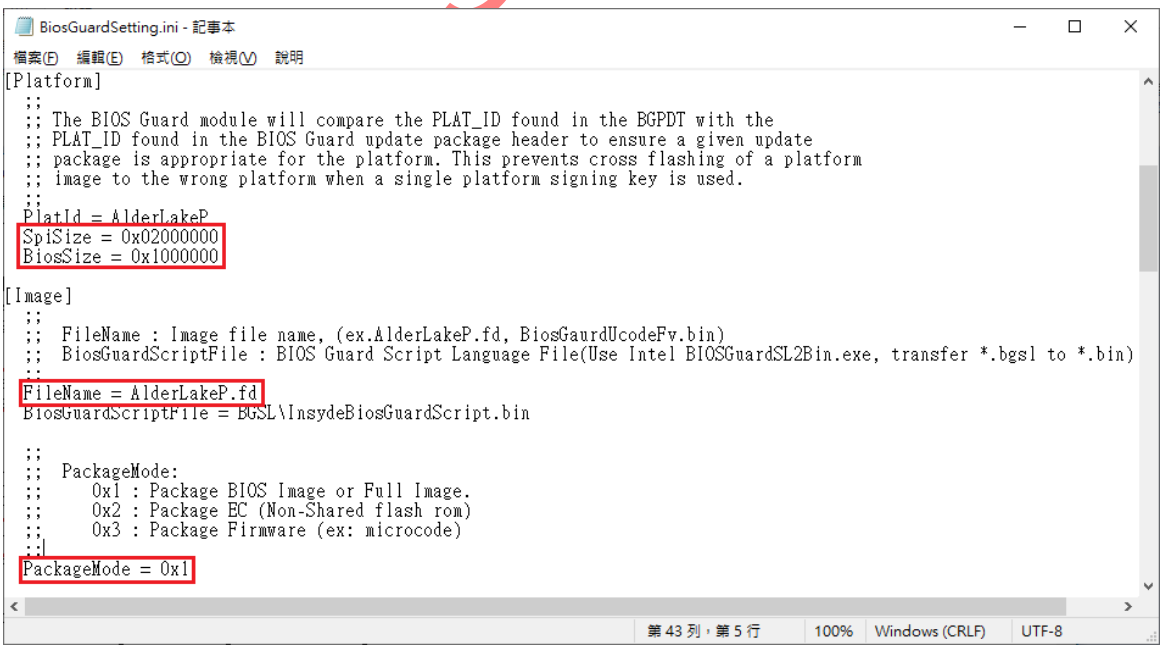
The following image describes the structure of "BIOS Guard Update Image".



29.2.1 BIOS Only Image

Combine "BIOS Image" with "Insyde Bios Guard Header", gen "BiosGuard_BIOS.fd"

1. Prepare pure BIOS image, AlderLakeX.fd, and copy to $(PROJECT_REL_PATH)\$(PROJECT_PKG)\PlatformConfig\BiosGaurd\
2. Update BiosGuardSetting.ini :
   (1) SpiSize = 0x02000000 **; CRB SPI ROM size is 32MB = 0x02000000**
   (2) BiosSize = 0x01000000 **; CRB SPI ROM size is 16MB = 0x01000000**
   (3) FileName= AlderLakeX.fd
   (3) PackageMode = 0x1



3. Execute BiosGuardImage.bat to create (1) InsydeBiosGuardHeader.bin (2) BiosGuard_BIOS.fd

BiosGuardImage.bat file And Command
$(PROJECT_REL_PATH)\$(PROJECT_PKG)\PlatformConfig\BiosGuard\BiosGuardImage.bat
copy /b "%BinFileName%" + /b "InsydeBiosGuardHeader.bin" "BiosGuard_BIOS.fd"

※ InsydeBiosGuarHeader.bin included (1)Header of BiosGaurd header & Script (2) many of BGUP , as above Picture.
※ BiosGuardImage.bat will call genBiosGuardHdr.exe which will create suitable InsydeBiosGuardHeader.bin then merge with BIOS image to BiosGuad_BIOS.fd

4. Sign the "Bios Guard Update Image" by Insyde Sign Tool

Sign Tool Command
iEFIFlashSigner.exe sign -n [OEM Private Key] -bios BiosGuard_BIOS.fd -ini platform.ini

29.2.2 Full Image

Combine "Full Image" with "Insyde Bios Guard Header", gen "BiosGuard_BIOS.fd"
1. Prepare pure BIOS image, AlderLakeX.fd, use Intel FIT tool to create Full Image, "Image.bin"
2. Copy fullImage to $(PROJECT_REL_PATH)\$(PROJECT_PKG)\PlatformConfig\BiosGaurd\
3. Update BiosGuardSetting.ini :
   (1) SpiSize = 0x02000000
   (2) BiosSize = 0x01000000
   (3) PackageMode = 0x1
   (4) BinFileName = Image.bin

4. Execute BiosGuardImage.bat to create (1) InsydeBiosGuardHeader.bin (2) BiosGuard_BIOS.fd

BiosGuardImage.bat file And Command
$(PROJECT_REL_PATH)\$(PROJECT_PKG)\PlatformConfig\BiosGuard\BiosGuardImage.bat
copy /b "%BinFileName%" + /b "InsydeBiosGuardHeader.bin" "BiosGuard_BIOS.fd"

※ InsydeBiosGuarHeader.bin included (1)Header of BiosGaurd header & Script (2) many of BGUP , as above Picture.
※ BiosGuardImage.bat will call genBiosGuardHdr.exe which will create suitable InsydeBiosGuardHeader.bin then merge with BIOS image to BiosGuad_BIOS.fd

5. Sign the "Bios Guard Update Image" by Insyde Sign Tool
Note:H2OFFT and Sign tool support full Image update via BIOS Guard from
Insyde H2OFFT x86 SHELL Package v.2.00.10.00 (Security Flash Package Version 2.00.16)
Insyde H2OFFT x86 WIN Package v2.01.06 (Security Flash Package Version 2.00.16)

Sign Tool Command
iEFIFlashSigner.exe sign -n [OEM Private Key] -bios BiosGuard_BIOS.fd -ini platform.ini

29.2.3 ME + BIOS Image

Combine "BIOS Image" with "Insyde Bios Guard Header", gen "BiosGuard_BIOS.fd" and sign with ME binary together.
1. Follow "Section 22.3.1 Bios Only Image" to create BiosGuard_BIOS.fd
2. Use Intel FIT tool to get updated ME image, FWUpdate.bin.
3. Sign the "Bios Guard Update Image" by Insyde Sign Tool

Sign Tool Command
iEFIFlashSigner.exe sign -n [OEM Private Key] -bios BiosGuard_BIOS.fd -ini Platform.ini -me FWUpdate.bin –oemid "00000000-0000-0000-0000-000000000000"

29.2.4 BIOS + EC Image

Combine "BIOS Image" with "Insyde Bios Guard Header", gen "BiosGuard_BIOS.fd" and sign with EC binary together.
EC means NON-shared ROM EC which is merge by Intel FIT tool.
1. Follow "Section 22.3.1 Bios Only Image", to create BiosGuard_BIOS.fd
2. Prepare pure EC binary.
3. Sign the "Bios Guard Update Image" by Insyde Sign Tool

Note: H2OFFT and Sign tool support BiosGuard BIOS + EC Image update via BIOS Guard from Insyde H2OFFT x86 SHELL Package v.2.00.10.00 (Security Flash Package Version 2.00.16)
Insyde H2OFFT x86 WIN Package v2.01.06 (Security Flash Package Version 2.00.16)

Sign Tool Command
iEFIFlashSigner.exe sign -n [OEM Private Key] -bios BiosGuard_BIOS.fd -ini Platform.ini -ec EC.bin

29.2.5 ME + BIOS + EC Image

Combine "BIOS Image" with "Insyde Bios Guard Header", gen "BiosGuard_BIOS.fd" and sign with EC and ME binary together.
EC means NON-shared ROM EC which is merge by Intel FIT tool.
1. Follow "Section 22.3.1 Bios Only Image", to create BiosGuard_BIOS.fd.
2. Use Intel FIT tool to get updated ME image, FWUpdate.bin.
3. Prepare pure EC binary.
4. Sign the "Bios Guard Update Image" by Insyde Sign Tool

Note: H2OFFT and Sign tool support ME + BiosGaurd BIOS + EC Image update via BIOS Guard from Insyde H2OFFT x86 SHELL Package v.2.00.10.00 (Security Flash Package Version 2.00.16)
Insyde H2OFFT x86 WIN Package v2.01.06 (Security Flash Package Version 2.00.16)

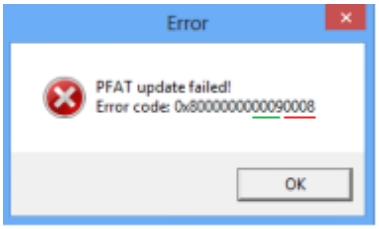
Sign Tool Command
iEFIFlashSigner.exe sign -n [OEM Private Key] -bios BiosGuard_BIOS.fd -ini Platform.ini -me FWUpdate.bin -oemid "00000000-0000-0000-0000-000000000000" -ec EC.bin

22.4 How to debug when BIOS Guard update fail

When “BIOS Guard Update Fail”, an error code will return by the flash tool. This error code is returned by BIOS Guard ACM directly, so you will know which verification step fails. Refer to IBP #574325 “Intel? Platform Protection Technology with BIOS Guard 2.0 – BIOS Specification”

Example:

“PFAT Update Fail” with following error code.



1. 0008 indicates the “BIOS Guard Error Code”
2. 0009 indicates the “ADDITIONAL_DATA”

   By referring to "Appendix A BIOS Guard Error Codes", we can see that this is a “ERR_BAD_BGUPC” error.

   Search the keyword “ERR_BAD_BGUPC” in the document, and you will find following table:

Since the “ADDITIONAL_DATA” is “0009”, it indicates your “Key Hash” is incorrect.

Appendix A BIOS Guard Error Codes

Table 14 – BIOS Guard Error Code Convention

Range	Convention
0x0000	No error
0x0001 – 0x0FFF	Errors that are detected before script execution
0x1000 – 0x8000	Reserved for future use
0x8001 – 0x8FFF	Errors that are may be detected after script execution begins
0x9000 – 0xEFFF	Reserved for future use
0xF000 – 0xFFFE	Allocated to the implementation for implementation specific use
0xFFFF	CPU detected error prior to BIOS Guard module execution

Table 15 – BIOS Guard Error Codes

Name	Value	Condition
ERR_OK	0x0000	Operation completed without error
ERR_UNSUPPORTED_CPU	0x0001	BIOS Guard module detected an incompatibility with the installed CPU. ADDITIONAL_DATA indicates the exact check that failed.
ERR_BAD_DIRECTORY	0x0002	BG_DIRECTORY check failed. ADDITIONAL_DATA indicates the exact check that failed.
ERR_BAD_BGPDT	0x0003	A pre-execution check of the BGPDT failed. ADDITIONAL_DATA indicates which specific check failed. ADDITIONAL_DATA indicates the exact check that failed.
ERR_BAD_BGUP	0x0004	An inconsistency was found in the update package. ADDITIONAL_DATA indicates which check failed.
ERR_SCRIPT_SYNTAX	0x0005	Unknown operator or name, or invalid syntax found in script
ERR_UNDEFINED_FLASH_OBJECT	0x0006	An unimplemented flash object was referenced

ERR_BAD_BGUPC		
ERR_BAD_BGUPC	0x0001	BGUPC overlaps AC_RAM
ERR_BAD_ BGUPC	0x0002	BGUPC is not in addressable DRAM
ERR_BAD_ BGUPC	0x0003	BGUPC overlaps another declared range
ERR_BAD_BGUPC	0x0004	Incorrect version
ERR_BAD_BGUPC	0x0005	Non-zero reserved field in BGUPC
ERR_BAD_BGUPC	0x0006	Unsupported crypto algorithm
ERR_BAD_BGUPC	0x0007	BGUPC is not in DMA protected range
ERR_BAD_BGUPC	0x0008	BGUP attributes are all zero
ERR_BAD_BGUPC	0x0009	Bad key hash
ERR_BAD_BGUPC	0x000A	Bad signature

22.5 How to update Bios Guard Platform Data Table (BGPDT) and Bios Guard EC Command

Please use H2O_PEI_CP_BIOSGUARD_EC_SUPPORT or H2O_PEI_CP_BIOSGUARD_UPDATE_BGPDT checkpoint, base on EC support BiosGuard Command to separate, to update before Bgpdt (Bios Guard Package Data Table) Hash complete in PEI Phase (PeiCpuPolicyBoardConfig.c).

For Checkpoint usage, please reference InsydeH2O® Chipset Technical Reference. And reference *22.7 Reference* document for BGPDT and EC detail description

22.5.1 Bios Guard EC Command

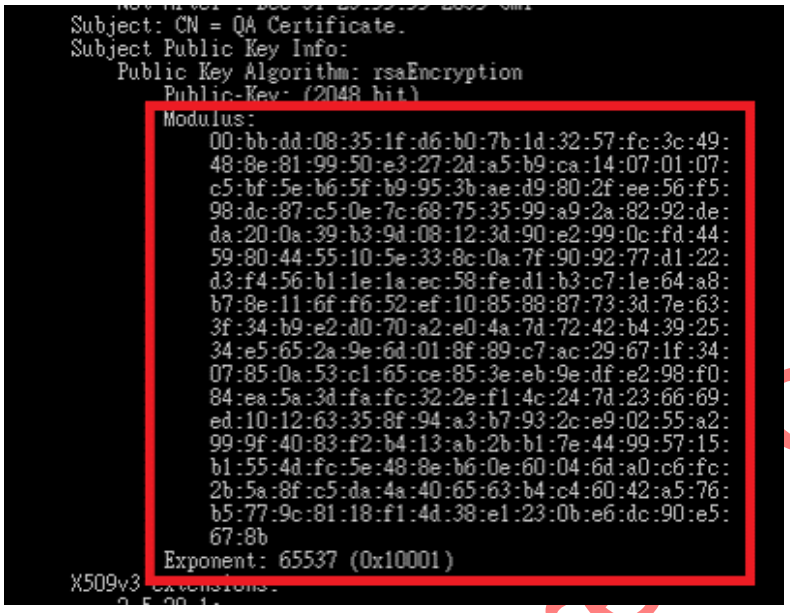
Area of the BIOS region of flash that is to be protected by BIOS Guard, only signed updates are allowed to this address range, need to be described in SFAM.

Should base on Project Flash map design and protect area requirement to describe in SPI ROM

22.5.2 Signed Flash Address Map

1. address basePrepare Tools:Key (ex.QA.cer)
   A. OpenSSL.exe
   B. Binary Edit Tool (ex.HxD)
2. Create Public Key Hash Steps:
   A. Install OpenSSL
   B. Put the Key.cer in folder where have OpenSSL.exe
   C. Use "Command Prompt" execute Sample command and get the message

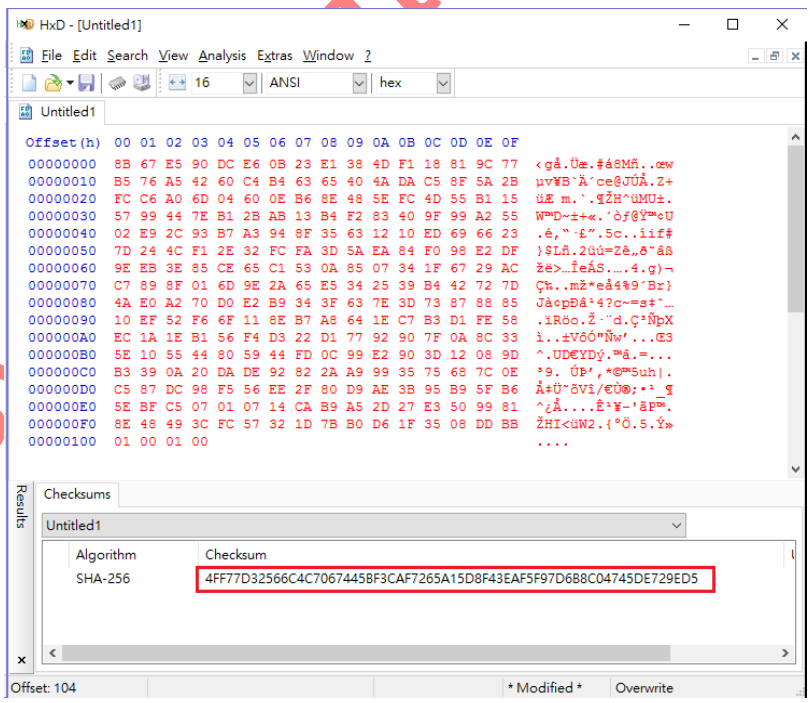
      "openssl.exe x509 -inform DER -in QA.cer -text"



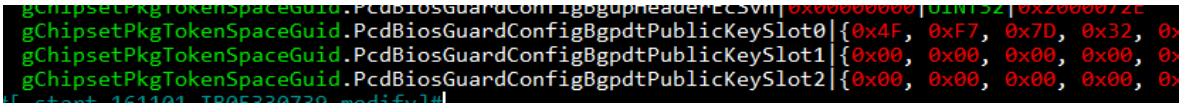
D. Take Modulus, upside down and get 256 bytes, and Exponent number 0x10001), get 4 bytes(0x00010001) and upside down.

※Because ":" is MSB notation . We need to use LSB notation.

E. Use Binary Edit Tool get the CheckSums (Analysis->CheckSums) value by SHA-256.



F. Set the Checksum value in One of PcdBiosGuardConfigBgpdtPublicKeySlot0/1/2 at **`$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.dec`** and build BIOS Binary.

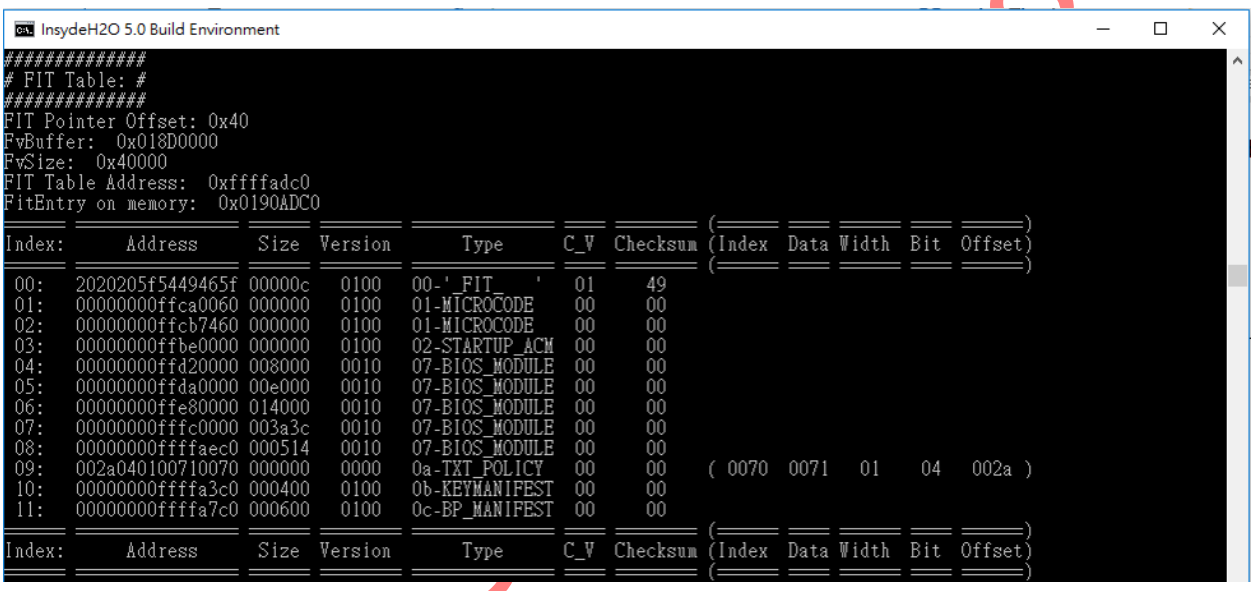


23 Boot Guard Support

23.1 Introduction

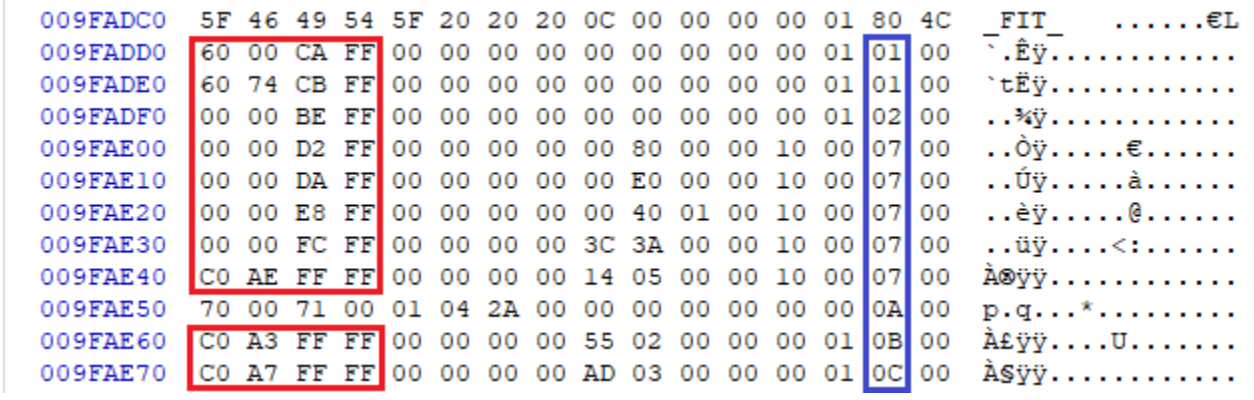
Please refer "Converged Boot Guard and Trusted Execution Technologies" (IBP#575506) and "Converged Boot Guard and Trusted Execution Technologies BIOS Writers Guide" (IBP#575623).

23.2 How to find FIT table information



Red: Memory address of each region.

Blue: Type of each region.



23.3 Related files

$(PLATFORM_BOARD_PACKAGE)/BiosInfo/*.*

%PLATFORM_FULL_PACKAGE%/Tools/BpmGen2/BpmGen2.exe

$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)/Tools/Bin/Win32/FitGen.exe

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Binary/BootGuardAcm/*.*

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Binary/Txt/*.*

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Include/BootGuardTableDefinition.h

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/PlatformConfig/bpmgen2.params

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/PlatformConfig/keyprivkey.pem

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/PlatformConfig/keypubkey.pem

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/PlatformConfig/privkey.pem

$(PROJECT_REL_PATH)/$(PROJECT_PKG)/PlatformConfig/pubkey.pem

23.4 Related tools

Before end of build process, ProjectPostBuild.bat will call FitGen.exe and BpmGen2.exe.

We need to input the GUIDs of BiosInfo.inf and ACM to FitGen.exe, then this tool will generate a BIOS image which includes FIT table.

BpmGen2.exe can generate KeyManifest.bin and Manifest.bin according to your key pairs (please refer 22.5), and package these manifests into BIOS image.

23.5 How to change key pairs for KM and BPM

Intel tools are designed to work together with the open source OpenSSL tool (version 1.0.2b or higher), follow Intel Reference Code setting, AlderLake Platform default use 2 key paris **keyprivkey.pem/keypubkey.pem(3k key)** and **privkey.pem/pubkey.pem(2k key).**

1. Use OpenSSL to generate a key pair:

   Command of generate a private key (3k key):

   ```
   openssl.exe genrsa -out keyprivkey.pem 3072
   ```

   Command of generate a private key (2k key):

   ```
   openssl.exe genrsa -out privkey.pem 2048
   ```

   Command of extract a public key from a private key:

   ```
   openssl.exe rsa -in privkey.pem -pubout -out pubkey.pem
   ```

2. Replace **keyprivkey.pem/keypubkey.pem** and **privkey.pem/pubkey.pem** in **$(PROJECT_REL_PATH)/$(PROJECT_PKG)/PlatformConfig**

3. Build code.

23.6 How to use MEU tool to generate key hash and manifest binary

1. meu.exe -gen meu_config (Generate meu_config.xml for later step).
2. meu.exe -gen OEMKeyManifest (Generate OEMKeyManifest.xml for later step).
3. Modify the meu_config.xml from step 1, change the OpenSSL tool path (Ex: SigningToolPath value="C:\OpenSSL-Win32\bin\openssl.exe")
4. meu.exe -keyhash pubkey_hash -key OEMkey.pem (Extract public key hash (.bin and .txt format) from OEMkey.pem).

   Note: Please use `$(PROJECT_REL_PATH)/$(PROJECT_PKG)/PlatformConfig/keyprivkey.pem` as OEMkey.pem
5. Modify the OEMKeyManifest.xml from step 2, change the HashBinary filename if you don't use "pubkey_hash" in step 4 (Ex: HashBinary value="pubkey_hash.bin")
6. meu.exe -f OEMKeyManifest.xml -o OEMKeyManifest.bin -key OEMkey.pem
7. Use the pubkey_hash.txt from step 4 and the OEMKeyManifest.bin from step 6 for ME setting.

23.7 BIOS settings

For Boot Guard enable:

`$(PROJECT_REL_PATH)/$(PROJECT_PKG)\Project.env` → `BOOT_GUARD_SUPPORT = YES`

For PTT enable:

`$(PROJECT_REL_PATH)/$(PROJECT_PKG)\Project.env` → `PTT_SUPPORT = YES`

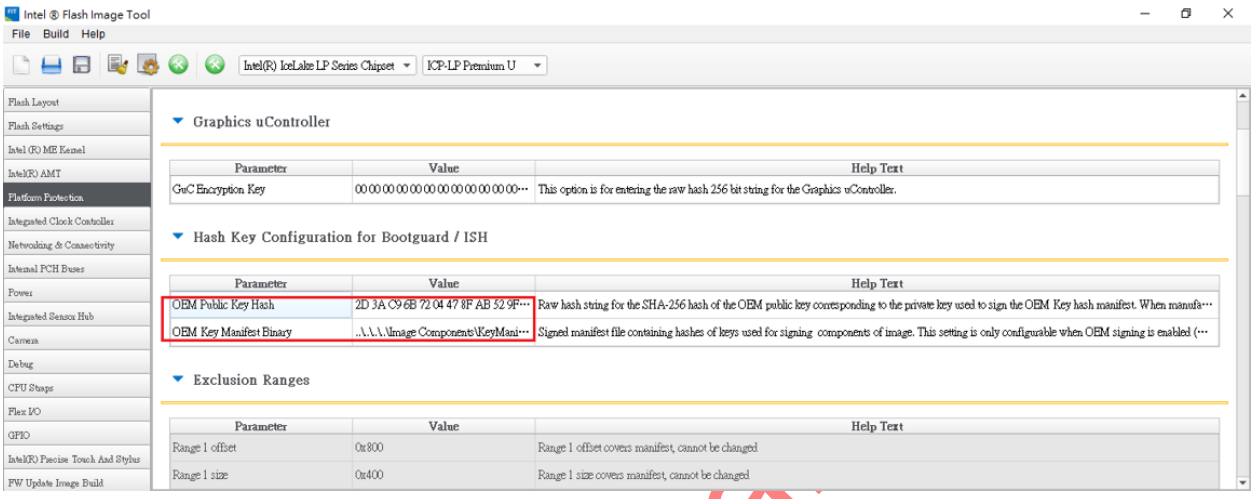
Front Page → Setup Utility → Chipset Configuration → Platform Trust Technology → Enabled

Check the Hardware version of CPU/PCH, and set `$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.env → PRODUCTION_SIGNED_ACM = YES` if necessary.

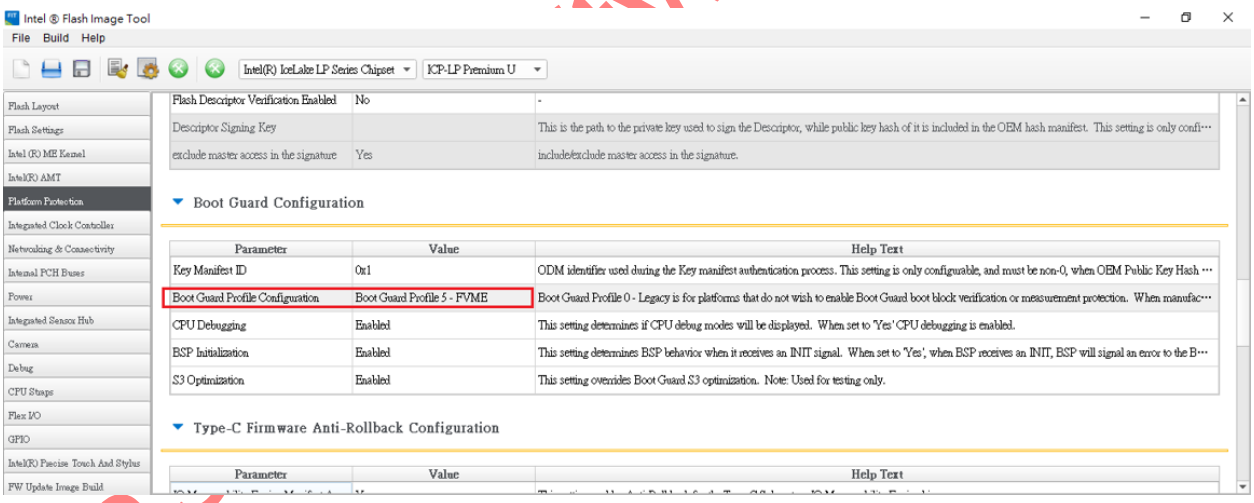
23.8 CSME settings

For Boot Guard enable:

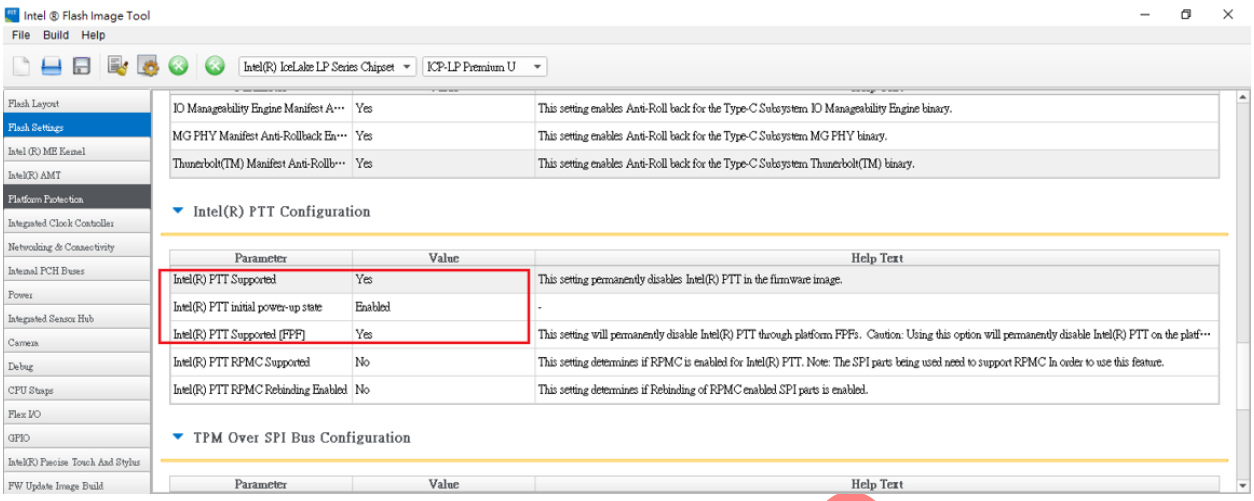
Make sure the OEM Public Key Hash and OEM Key Manifest Binary are correct (Please refer 23.6).



Set Boot Guard Profile Configuration to Boot Guard Profile 3 – VM, Boot Guard Profile 4 – FVE or Boot Guard Profile 5 – FVME (by request).



For PTT enable:



23.9 Chain of Trust

In $(PROJECT_PKG)/Project.dsc, set Chain of trust enabled with bootguard by default

```
gInsydeTokenSpaceGuid.PcdH2OFdmChainOfTrustSupported|$(BOOT_GUARD_SUPPORT)
```

23.10 Reference

Converged Boot Guard and Trusted Execution Technologies (IBP#575506).

Converged Boot Guard and Trusted Execution Technologies BIOS Writers Guide (IBP#575623).

InsydeH2O Build Technical Reference

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/Project.env
```

```
$(PROJECT_REL_PATH)/$(PROJECT_PKG)/$(CHIPSET_PKG).dec
```

24 RMT Data Dump

This section describes how to enable RMT (Rank Margin Test) in the Alder Lake codebase.

RMT data is usually used by ODM to test if the board layout for memory is suitable. If the board layout is OK, the test analyzed tool will return passed.

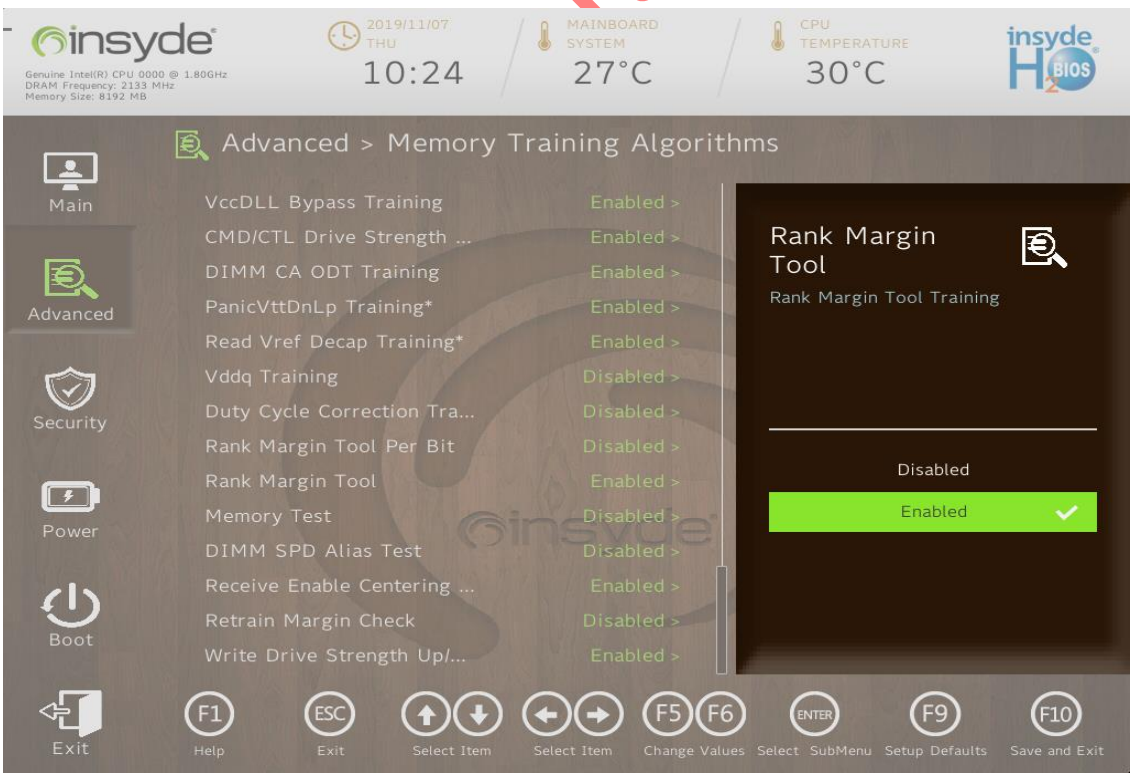
24.1 Related files/tools

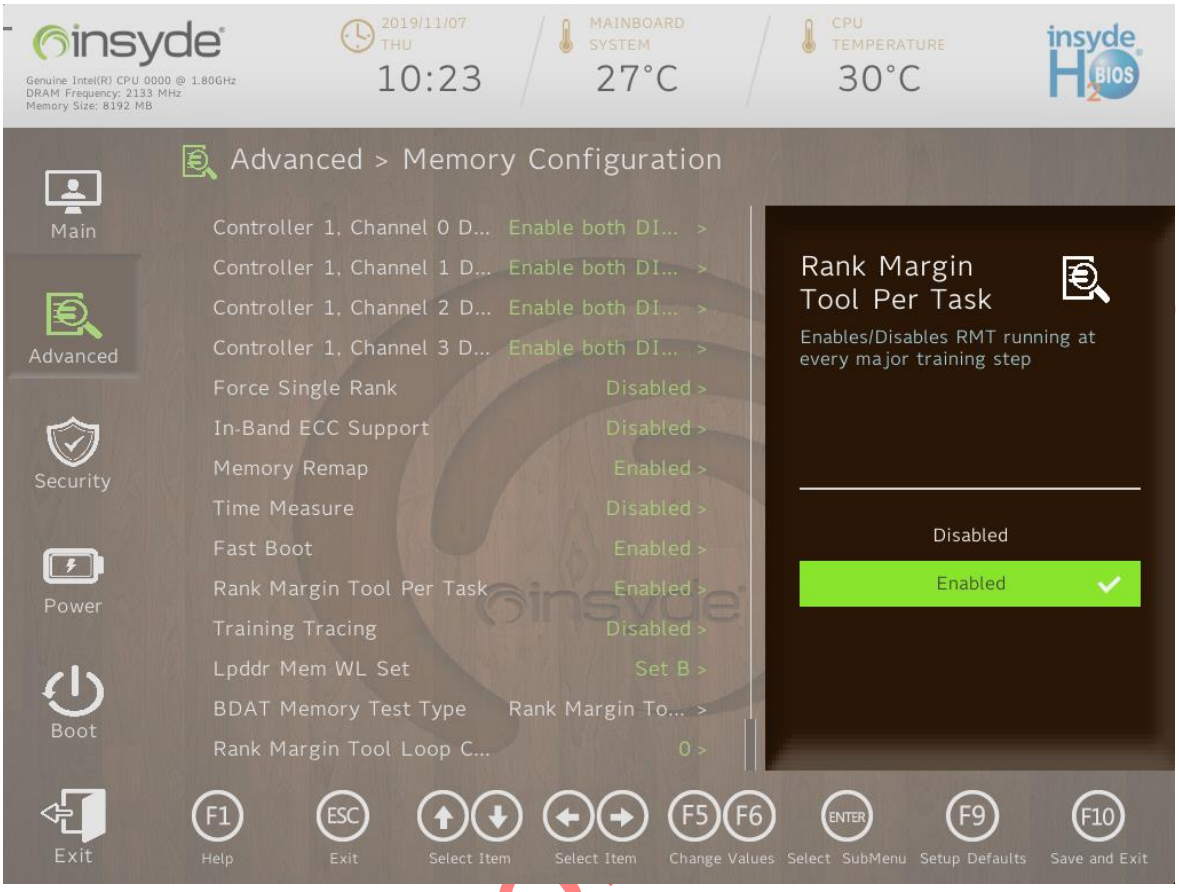
1. Tool to capture RMT dump data (Ex: Putty)
2. 574682_MemoryRMT_Tools_Rev2p05.xlsm (Tool to calculate RMT data)
3. Cable to capture the efidebug log.

24.2 How to build RMT data

If want to do **RMT test**, please follow the steps below.

1. Build bootleg by build command "nmake efidebug"
2. Enter SCU, switch “Rank Margin Tool” and "Rank Margin Tool Per Task" to Enabled
3. Then next boot, it will dump RMT data





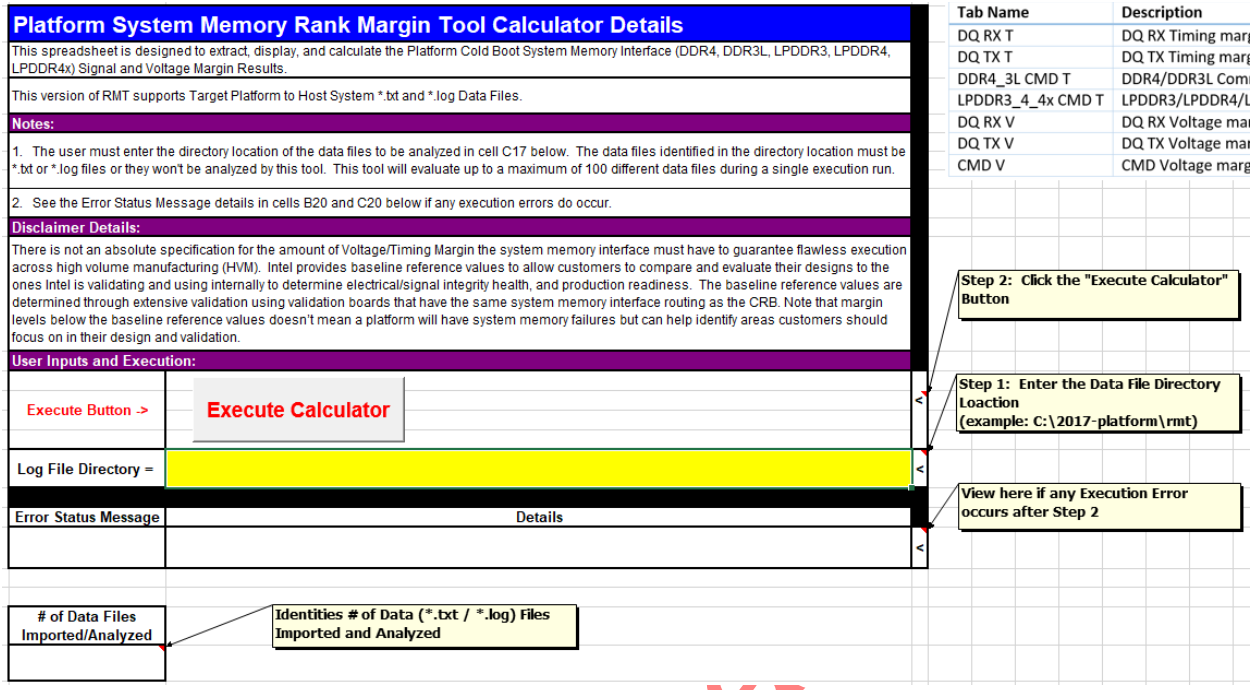
24.3 How to verify if RMT data is OK

This is the RMT calculatori¦s job to calculate if the board layout fulfilled the requirement.

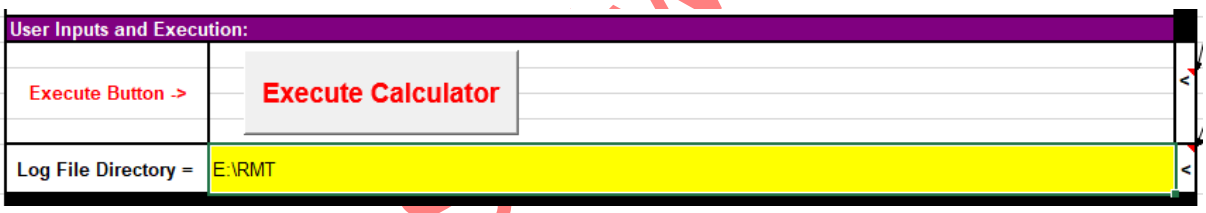
In this case, ***574682_Memory_RMT_Tools_Rev2p05.xlsm*** (excel file) is the calculator.

This tool can be obtained from Intel IBL website or your customer has a copy of it for you to do the job.

1. Place your log file (ex: ***your_textFileName.txt***) into the **same folder** where the excel file is located.
2. Opened the excel file and click on ¡§Instri¨ <Tab> at the bottom of the page. See the following figure:



3. Give it the absolute path of your RMT data file.



4. Click on "**Execute Calculator**" to run the calculator.

5. When the job is completed, click on "***Summary***" <Tab> and you will probably see the following figure:

Worse Case System Memory Margin Result Summary

System Memory Speed	3200	MT/s	SA-GV High	Timing per Step			4.88	pS/Step	
DRAM Device Type	LPDDR4		Gear2	RD VREF Voltage per Step			2.86	mV/Step	
System Memory Voltage	1.10	V		WR VREF Voltage per Step			2.86	mV/Step	
Total Data Files Analyzed	1	Files		CMD VREF Voltage per Step			2.86	mV/Step	
Timing Parameter	Left Side	Right Side	Total	Units	Left Side	Right Side	Total	Units	PASS/FAIL
Min RD/WR Baseline	10	10	20	Steps	48.0	48.0	97.0	pS	
Read Timing Margin	20	19	39	Steps	97.7	92.8	190.4	pS	PASS
Write Timing Margin	21	21	42	Steps	102.5	102.5	205.1	pS	PASS
Min CMD Baseline	10	10	20	Steps	48.0	48.0	97.0	pS	
CMD Timing Margin	49	32	81	Steps	239.3	156.3	395.5	pS	PASS
Voltage Parameter	Low Side	High Side	Total	Units	Low Side	High Side	Total	Units	
Min RD VREF Baseline	18	18	36	Steps	51.0	51.0	103.0	mV	
RD VREF Voltage Margin	46	51	97	Steps	131.8	146.1	277.9	mV	PASS
Min WR VREF Baseline	12	12	24	Steps	52.0	52.0	105.0	mV	
WR VREF Voltage Margin	40	40	80	Steps	114.6	114.6	229.2	mV	PASS
Min CMD VREF Baseline	12	12	24	Steps	52.0	52.0	105.0	mV	
CMD VREF Voltage Margin	36	35	71	Steps	103.1	100.3	203.4	mV	PASS

6. This indicates that your test platform layout has fulfilled the RMT requirement.

25 Build FSP binary

This section describes how to re-build the FSP binary from the FSP source code in the Alder Lake code base.

In most case, you can use the FSP binary directly. For debug purpose, you may need to re-build the FSP binary from the FSP source code.

After re-building the FSP binary, you can build the final BIOS binary with this new FSP binary.

25.1 How to build FSP binary

If you want to build FSP binary, please follow the steps below:

1. Install Python first, if it is not installed before. Required Python version: 2.5 ~ 2.7.
   Can go to https://www.python.org/downloads/
   Make sure Python path is installed under C:. For example, C:\Python2X

2. Run "ProjectBuild.bat" to enter build environment

3. Run "BuildFsp.cmd /option" in build environment. Available options are:

   a) "BuildFsp.cmd /r" for release FSP binary.

   b) "BuildFsp.cmd /d" for efidebug FSP binary.

   c) "BuildFsp.cmd /ddt" for DDT FSP binary.

   d) "BuildFsp.cmd /clean" for removing Build and Conf\.cache directories.

   e) "BuildFsp.cmd /h" or "BuildFsp.cmd /?" for option usage

After build complete, the FSP binary will be copied to xxxxxFspBinPkg automatically.

26 Board ID feature

This section describes AlderLake new board ID feature.
Use PcdBoardId to directly control AlderLake Intel platform BoardID.

Use PcdH2OBoardId (PcdSkuId) for support multi-configuration (ex: Project.var).

Note: It should check the PcdH2OBoardId value is whether declared in both [SkuIds] and SKUID_IDENTIFIER of Project.dsc otherwise, the system will hang with the POST code **0x8E**.

26.1 Related files and parameters

1. ProjectSetup.bat
   1.1 **CrbBuild** - for select CRB or OEM project.

2. $(PROJECT_PKG)/Project.dsc

   2.1 **PcdBoardId** - for select Intel Platform BoardIds

   2.2 **PcdCrbSkuId** - for customize Intel RC SKUIDs

3. $(PROJECT_PKG)/Project.env

   3.1 **USE_INTEL_CRB_H8_EC –** select use CRB EC lib or OEM EC lib

26.2 How to use

1. For OEM project, please setting **CrbBuild**=NO

2. Setting **PcdBoardId** to proper Intel Platform BoardIds (BoardID define in PlatformBoardId.h)

3. Setting **PcdCrbSkuId** to correct value.

   AlderLake RC use 1 SKUID:

   3.1 (0x03) SkuIdAdlS, for AlderLake-S

4. Setting **USE_INTEL_CRB_H8_EC**

   4.1 YES:Use CRB EC lib

   4.2 NO: Use OEM EC lib (OEM need to implement related EC libs which comment out by **PcdUseCrbEcFlag**)

26.3 Compare to old BOARDID feature

Old	**New**

Set **PcdDefaultBoardId** = 0xFF for OEM project	Set **CrbBuild** = NO for OEM project
Use **PcdOemProjectReferenceIntelCrb** to select Intel Platform BoardIds	Use **PcdBoardId** to select Intel Platform BoardIds

27 Chasm Falls

This section describes how to open Insyde Chasm Falls on AlderLake and how to create update image.

27.1 Introduction

AlderLake support gen1 & gen2 and Insyde Chasm falls is designed base on Intel Chasm falls. For more detail, please refer documents in below:

Gen1: 611532-chasmfalls-gen1-capsuleupdate-sas-rev0p9.pdf
611910-chasmfalls-gen1-recovery-sas-rev1p2.pdf
Gen2: 611644-chasmfalls-gen2-capsule-update-sas-rev0p54.pdf
615670-chasmfalls-gen2-recovery-resillency-sas-rev0-5.pdf
615670_chasmfalls_gen2_recovery_resiliency_sas_rev1p0.pdf

27.2 How to open feature

This chapter will introduce how to open Chasm falls feature by setting related pcds.

27.2.1 Main pcd

gChipsetPkgTokenSpaceGuid.PcdChasmFallsSupport:

Enable/Disable Chasm falls feature, and it also controllers which flash map is used. Default is 0.
(0 - Not support ChasmFalls feature, 1 - Support ChasmFalls gen 1, 2 - Support ChasmFalls gen 2)

```
# Defaule layout not support Chasm Falls
  gChipsetPkgTokenSpaceGuid.PcdChasmFallsSupport|0
```

AlderLake has integrated all related pcds to Project.dsc with pcd “PcdChasmFallsSupport”. Basically project only need to open this pcd.

(1)PcdsFixedAtBuild:

```
#
# Enable/Disable Insyde/Intel ChasmFalls
#
!if gChipsetPkgTokenSpaceGuid.PcdChasmFallsSupport == 1 || gChipsetPkgTokenSpaceGuid.PcdChasmFallsSupport == 2
  # Enable Intel ChasmFalls (Default chasmfalls feature will not enable Intel chasmfalls)
  #gPlatformModuleTokenSpaceGuid.PcdCapsuleEnable|TRUE
  # ME
  gPlatformModuleTokenSpaceGuid.PcdMeResiliencyEnable|TRUE
  # Monolithic
!if gChipsetPkgTokenSpaceGuid.PcdMonolithicCapsuleUpdateSupported == TRUE
!if gChipsetPkgTokenSpaceGuid.PcdChasmFallsSupport == 1
  # Default support case is BIOS(Gen1 12M)+ ME(Max size: Corporate 9.8M) total 23M.
  # Since the flash layout of Gen 1 is the same as Gen 2, the setting was changed to the same as Gen 2
  gChipsetPkgTokenSpaceGuid.PcdMeFirmwareUpdateReservedMemorySize|0x1A00000
!else
  # Default support case is BIOS(Max size: Gen2 16M)+ ME(Max size: Corporate 9.8M) total 26M.
  gChipsetPkgTokenSpaceGuid.PcdMeFirmwareUpdateReservedMemorySize|0x1A00000
!endif
!endif
!endif
```

(2) PcdsFeatureFlag:

```
#
# Enable/Disable Insyde ChasmFalls
#
!if gChipsetPkgTokenSpaceGuid.PcdChasmFallsSupport == 1
  # BIOS
  gInsydeTokenSpaceGuid.PcdH2OBiosUpdateFaultToleranceEnabled|TRUE
  gInsydeTokenSpaceGuid.PcdH2OBiosUpdateFaultToleranceResiliencyEnabled|FALSE
  gChipsetPkgTokenSpaceGuid.PcdSpiReadByMemoryMapped|FALSE
  # Monolithic
  gChipsetPkgTokenSpaceGuid.PcdMonolithicCapsuleUpdateSupported|TRUE
  # ME
  gChipsetPkgTokenSpaceGuid.PcdMeCapsuleUpdateSupported|TRUE
!elseif gChipsetPkgTokenSpaceGuid.PcdChasmFallsSupport == 2
  # BIOS
  gInsydeTokenSpaceGuid.PcdH2OBiosUpdateFaultToleranceEnabled|TRUE
  gInsydeTokenSpaceGuid.PcdH2OBiosUpdateFaultToleranceResiliencyEnabled|TRUE
  gChipsetPkgTokenSpaceGuid.PcdSpiReadByMemoryMapped|FALSE
  gInsydeTokenSpaceGuid.PcdH2OBaseCpVerifyFvSupported|TRUE
  # Microcode
  gChipsetPkgTokenSpaceGuid.PcdUcodeCapsuleUpdateSupported|TRUE
  # Monolithic
  gChipsetPkgTokenSpaceGuid.PcdMonolithicCapsuleUpdateSupported|TRUE
  # ME
  gChipsetPkgTokenSpaceGuid.PcdMeCapsuleUpdateSupported|TRUE
!endif

!if gPlatformModuleTokenSpaceGuid.PcdTdsEnable
  gInsydeTokenSpaceGuid.PcdH2OBdsCpBootDeviceEnumCheckBootOptionSupported|TRUE
!endif
```

27.2.2 Bios

This section will introduce more detail for Insyde Chasm Fall Bios related pcds. **Gen2 is the enhancement of Gen1. If user want to enable gen2 feature, it also needs to enable gen1 feature.**

1. Feature pcds:

   (1) gInsydeTokenSpaceGuid.PcdH2OBiosUpdateFaultToleranceEnabled:
   Enable/Disable Gen1 Insyde Chasm falls bios.

   (2) gInsydeTokenSpaceGuid.PcdH2OBiosUpdateFaultToleranceResiliencyEnabled:
   Enable/Disable Gen2 Insyde Chasm falls bios.

2. Other related pcds:

   (1) gChipsetPkgTokenSpaceGuid.PcdSpiReadByMemoryMapped:
   Enable top swap will change the memory map content automatically, so Chasm falls doesn't use SpiReadByMemoryMapped.

   (2) gInsydeTokenSpaceGuid.PcdH2OBaseCpVerifyFvSupported:
   Gen2 only. This checkpoint will help to detect if SBB(OBB) is corrupt or not.

27.2.3 Microcode

Microcode update only supported in Gen2. **Also microcode FV belong to BIOS region, the related pcds for BIOS needs to be opened too.** You can refer to chapter 27.2.2 for more detail.

Ucode capsule feature pcds:

   (1) gChipsetPkgTokenSpaceGuid.PcdUcodeCapsuleUpdateSupported:
   Enable/Disable Intel or Insyde Microcode update.

27.2.4 ME

This section will introduce more detail for Insyde Chasm Fall Microcode related pcds.

1. Me capsule feature pcd:

(1) gChipsetPkgTokenSpaceGuid.PcdMeCapsuleUpdateSupported:
Enable/Disable Insyde normal ME capsule service.

(2) gPlatformModuleTokenSpaceGuid.PcdMeResiliencyEnable:
Enable/Disable Insyde Chasm falls ME resiliency service.

27.2.5 Monolithic

This section will introduce more detail for Insyde Chasm Fall Monolithic related pcds. Default combination is BIOS + ME, therefore it needs to open the related pcds of both BIOS and ME. You can refer to chapter 27.2.2 and chapter 27.2.4 for more detail.

1. Monolithic capsule feature pcd:

(1) gChipsetPkgTokenSpaceGuid.PcdMonolithicCapsuleUpdateSupported:
Enable/Disable Insyde Monolithic capsule service.

2. Other related pcd:

(1) gChipsetPkgTokenSpaceGuid.PcdMeFirmwareUpdateReservedMemorySize:
ME firmware update reserved memory size. This size depends on monolithic update supported case.

Example:
Default support case is BIOS(Max size: 16M)+ ME(Max size: Corporate 9.8M) total 26M.

27.2.6 Boot Guard Acm (BtGAcm)

BtGAcm update only supported in Gen2.**Also microcode FV belong to BIOS region, the related pcds for BIOS needs to be opened too.** You can refer to chapter 27.2.2 for more detail.

Unmark BtGacm capsule feature pcds:

(1) gPlatformModuleTokenSpaceGuid.PcdCapsuleEnable|TRUE

27.3 How to adjust top swap size (PBB/PBBR)

Top swap size is same as the size of PBB (IBB). Also top swap size is PBBR (IBBR). If project chose other top swap size, the following need to be modified.

1. Project.fdf

(1) gInsydeTokenSpaceGuid.PcdFlashPbbSize

Default PBB(IBB) size is 4MB.

```
# FW_BINARIES ~ FV_RECOVERY0
# Must same as Top Swap Block Size in ME setting and FLASH_REGION_FW_RESILIENCY_RESERVED_SIZE
SET gInsydeTokenSpaceGuid.PcdFlashPbbsize = $(FLASH_REGION_FV_RECOVERY0_SIZE) + $(FLASH_REGION_FLASH_DEVICE_
```

(2) FLASH_REGION_FW_RESILIENCY_RESERVED_SIZE

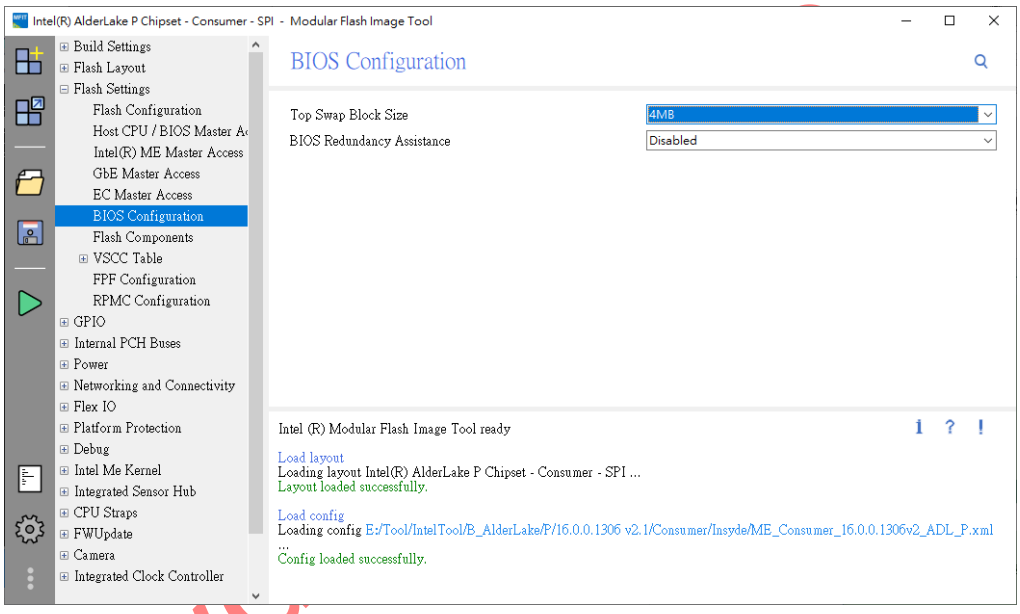
Gen2 only. This is PBBR(IBBR) region, which is top swap region.

```
!if gChipsetPkgTokenSpaceGuid.PcdChasmFallsSupport == 2
DEFINE FLASH_REGION_FW_RESILIENCY_RESERVED_OFFSET                = $(FLASH_REGION_FV_RESERVED_OFFSET) + $(FLASH_REGION_FV_RESERVED_SIZE
# Must same as gInsydeTokenSpaceGuid.PcdFlashPbbSize
DEFINE FLASH_REGION_FW_RESILIENCY_RESERVED_SIZE                  = 0x00400000
!endif
```

2. ME setting

(1) Top swap Block Size

Tops swap block size equal to PBBR(IBBR) size.



27.4 How to adjust SBB

Basically the SBB (OBB) size will calculate automatically once the pcd "PcdFlashPbbSize"is set. AlderLake follow demo bios design to program these regions and also bios will set variable during Chasm falls update.Therefor, Chasm falls shouldn't put variable region into SBB (OBB). However, some of projects do have some FV in front of variable region. Insyde Chasm Falls adjust SBB range to the space where was out of PBB(IBB) and PBBr(IBBr), and use Pcd VariableFlashReservedBase and VariableFlashReservedSize/VARIABLE_REGION_SZIE to deal with Variable region be skipped in hash and verify mechanism.

Variable region:

Default variable region is NV_VARIABLE_STORE ~ NV_FACTORY_COPY.

```
: DEFINE FLASH_REGION_NVSTORAGE_SUBREGION_NV_FACTORY_COPY_SIZE        = 0x00020000
: !if gChipsetPkgTokenSpaceGuid.PcdChasmFallsSupport == 1 || gChipsetPkgTokenSpaceGuid.PcdChasmFallsSupport == 2
: # NV_VARIABLE_STORE ~ NV_FACTORY_COPY
: DEFINE VARIABLE_REGION_SIZE                                  = $(FLASH_REGION_NVSTORAGE_SUBREGION_NV_VARIABLE_STORE_SIZE) + $(FLASH_REGION_NVSTORAGE_SUBR
: !endif
```

```
: # NV_COMMON_STORE_SUBREGION_NV_BVDT ~ DXE_FV_RESERVED
: # Since the flash layout of Gen 1 is the same as Gen 2, the ChasmFalls bios flash layout of Gen 1 was changed to the same as Gen 2
: !if gChipsetPkgTokenSpaceGuid.PcdChasmFallsSupport == 1
: SET gInsydeTokenSpaceGuid.PcdFlashSbbSize = $(gInsydeTokenSpaceGuid.PcdFlashAreaSize) - $(gInsydeTokenSpaceGuid.PcdFlashPbbSize) - $(gInsydeTokenSpaceGuid.PcdFlashPbbSize)
: !else
: SET gInsydeTokenSpaceGuid.PcdFlashSbbSize = $(gInsydeTokenSpaceGuid.PcdFlashAreaSize) - $(gInsydeTokenSpaceGuid.PcdFlashPbbSize) - $(gInsydeTokenSpaceGuid.PcdFlashPbbSize)
: SET gChipsetPkgTokenSpaceGuid.VariableFlashReservedBase = $(gSiPkgTokenSpaceGuid.PcdBiosAreaBaseAddress) + $(FLASH_REGION_NVSTORAGE_SUBREGION_NV_VARIABLE_STORE_OFFSET)
: SET gChipsetPkgTokenSpaceGuid.VariableFlashReservedSize = $(VARIABLE_REGION_SIZE)
: !endif
: !endif
```

27.4.1 How to describe Reserved Area

When system crash then start rollback action, some FV data should be keep in, ex. MSDM data. These FV area also need to skip hash and verify mechanism.
Must NOT change the FAULTTOLERANCE_SKIP_TABLE place to other file, otherwise, calculate SBB hash mechanism will be incorrect.

MultBoardPkg\Include\ChasmFallsReservedDefine.h

```
typedef struct {
  UINT32    SkipAddress;
  UINT32    SkipSize;
} FAULTTOLERANCE_SKIP_TABLE;

//
// Skip table
//
FAULTTOLERANCE_SKIP_TABLE mSkipHashInSbbTable[] = {
//  {FixedPcdGet32 (PcdFlashNvStorageMsdmDataBase), FixedPcdGet32 (PcdFlashNvStorageMsdmDataSize)}
  {0,0}
};
```

Notice : When BIOS Guard Enabled, only skipped ONE continuous area which described in FAULTTOLERANCE_SKIP_TABLE. It can use calculation to describe the region.

```
//
// Skip table
//
FAULTTOLERANCE_SKIP_TABLE mSkipHashInSbbTable[] = {
//  Example :
  {FixedPcdGet32 (PcdFlashNvStorageDmiBase), FixedPcdGet32(PcdFlashNvStorageDmiSize) + FixedPcdGet32 (PcdFlashNvStorageMsdmDataSize)},
  {0,0}
};
```

If use Pcds in the FAULTTOLERANCE_SKIP_TABLE, also need to add these Pcds in BiosUpdateFaultToleranceDxe.inf, CrisisRecoveryPei.inf and which driver include the ChasmFallsReservedDefine.h.

27.5 How to modify microcode version/lsv/version string

MicrocodeVersion.data instead pcd is used to report information for ESRT table. This file include microcode version, lowest supported version and firmware version string.

Default MicrocodeVersion.data in Microcde FV have these values:

(1) Microcode version = 1
(2) Lowest supported version = 1
(3) Firmware version string = "Version 0.0.0.1"

You can generate MicrocodeVersion.data by using GenMicrocodeVersion.exe. Or you can directly edit MicrocodeVersion.data. ReadMe.txt for GenMicrocodeVersion.exe under PyScript folder describe how to use this tool.
(Intel\AlderLake\AlderLakePlatSamplePkg\Features\CapsuleUpdate\Tools\NewGenCap\PyScript\GenMicrocodeVersion.exe)

27.6 Size of ESP Partition for Chasm Falls

Chasm Falls will use ESP partition to save the backup file. This chapter will list all the backup files in each case and the file size for RVP. OEM needs to be checked these files if it has limitation for the space of ESP partition.

27.6.1 For Gen 1

(1) BIOS update case:

1. CapsuleUpdateFile1000.bin (After flash completed, this file will be Deleted. BIOS capsule image. For RVP: the Gen1 isflash.bin is about 13.5M, the Gen1 isflash.bin with BIOS Guard enabled is about 13.7M)

2. BackupSBB.bin (After flash completed, this file will be deleted. SBB backup file. For RVP, the file size is 7.6M)

The total max size is about 21.3M.

(2) Me update case:

1. MeRecov.Cap (ME backup image. For Consumer ME, the isflash.bin size is about 5.33M Corporate ME, the isflash.bin size is about 12.2M)

2. UxCap.Cap (Windows UX image. A graphic shown in Me resiliency. The size is about 405K)

3. MeRecovN.Cap (After flash completed, this file will be deleted. ME capsule as temp ME backup image. The file size is same as MeRecov.Cap)

4. CapsuleUpdateFile1000.bin (After flash completed, this file will be deleted. ME capsule image. The file size is same as MeRecov.Cap)

5. CapsuleUpdateFile0001.bin (After flash completed, this file will be deleted. Windows UX capsule image. The file size is same as UxCap.Cap)

The total max size is about 37.4M.

Notice that: If OEM doesn't want ME resiliency function, the file 1, 2, 3 will not exist.

(3) Monolithic (Default: BIOS + ME) update case:

1. MeRecov.Cap (Monolithic (Default: BIOS + ME) capsule as ME backup image. For BIOS+Consumer ME, the total file size is about 17.3M. BIOS with Bios guard enabled + Consumer ME, the total file size is about 17.5M. BIOS + Corporate ME, the total file size is about 24.2M. BIOS with Bios guard enabled + Corporate ME, the total file size is about 24.4M)

2. UxCap.Cap (Windows UX image. The graphic for Me resiliency. The size is about 405K.)

27.6.2 For Gen 2

(1) BIOS update case:

1. CapsuleUpdateFile1000.bin (After flash completed, this file will be Deleted. BIOS capsule image. For RVP: the Gen 2 isflash.bin is about 17.5M. the Gen 2 isflash.bin with BIOS Guard enabled is about 17.7M)

2. BackupSBB.bin (SBB backup file. For RVP, the file size is 7.6M) The total max size is about 25.3M.

(2) Me update case:

All the files and size are same as Gen1 Me update case. You can refer to Chapter 27.6.1 (2).

(3) Microcode update case:

1. CapsuleUpdateFile1000.bin (After flash completed, this file will be deleted. Microcode capsule image. For RVP, the size depends on microcode FV)

2. CapsuleUpdateFile0001.bin (After flash completed, this file will be deleted. Windows UX capsule image. The file size is same as UxCap.Cap)

3. BackupSBB.bin (SBB backup file. For RVP, the file size is 7.6M) The total max size is above 7.6M.

(4) Monolithic (Default: BIOS + ME) update case:

1. MeRecov.Cap (Monolithic (Default: BIOS + ME) capsule as ME backup image.
For BIOS + Consumer ME, the total file size is about 21.3M. BIOS with Bios guard enabled + Consumer ME, the total file size is about 21.5M. BIOS + Corporate ME, the total file size is about 28.2M. BIOS with Bios guard enabled + Corporate ME, the total file size is about 28.4M.)

2. UxCap.Cap (Windows UX image. The graphic for Me resiliency. The size is about 405K.)

27.7 Build Bios with Bios Guard Enabled

Chasm Falls and Bios Guard can be supported in the same time, no matter Gen1 or Gen2. You can refer to chapter 22 to support Bios Guard Feature

This section will introduce more detail for Build Chasm Fall **Gen2** Bios image with Bios Guard.

In post build process, Bios Guard Update Package Header and Bios Guard Update Package Certificate will be created for each PBB (IBB), PBBR (IBBR) and SBB(OBB). In order to do Chasm Falls sync and rollback flow.

For Bios Guard Update Package Certificate, please check below setting.

(1) BiosGuardPostBuild.bat

Insyde iEFIFlashSigner provide command to create suitable Bios Guard certificate format, please change your own key.
If you have own tool to create certificate, these command can be replaced.

Bios Guard Update Package Certificate format detail can be reference IBP #574325 "Intel? Platform Protection Technology with BIOS Guard 2.0 – BIOS Specification, *Chapter 3.4 BIOS Guard Update Package Certificate*

Table 3-6. BIOS Guard Update Package Certificate

Name	Offset	Size	Description
Version	0	2	Version of the BGUPC structure. Must be 0x0001.
Reserved	2	2	Must be 0.
Algorithm	4	4	Indicates the signing algorithm and key size. Algorithm == 1 indicates PKCS1 1.5 digital signature (SHA-256 hash, RSA 2048 key) Algorithm == 2 indicates PKCS1 v2.1 EMSA/PSS digital signature (SHA-256, RSA 2048)
Cert	8	Depends on Algorithm.	Format and size depends on Algorithm. Format for Algorithm == 1 or 2 is defined below.

(2) Microsoft Signtool
When Insyde iEFIFlashSigner created certificate will called Microsoft Signtool, please make sure your DEVTLS Revision is newer then 977. Or download and installed, reference InsydeH2O 5.4 Alder Lake Power On Technical Reference Guide, *Chatper 2.1.5.2 SignTool Caution* and *Chatper 2.3 Signtool.*

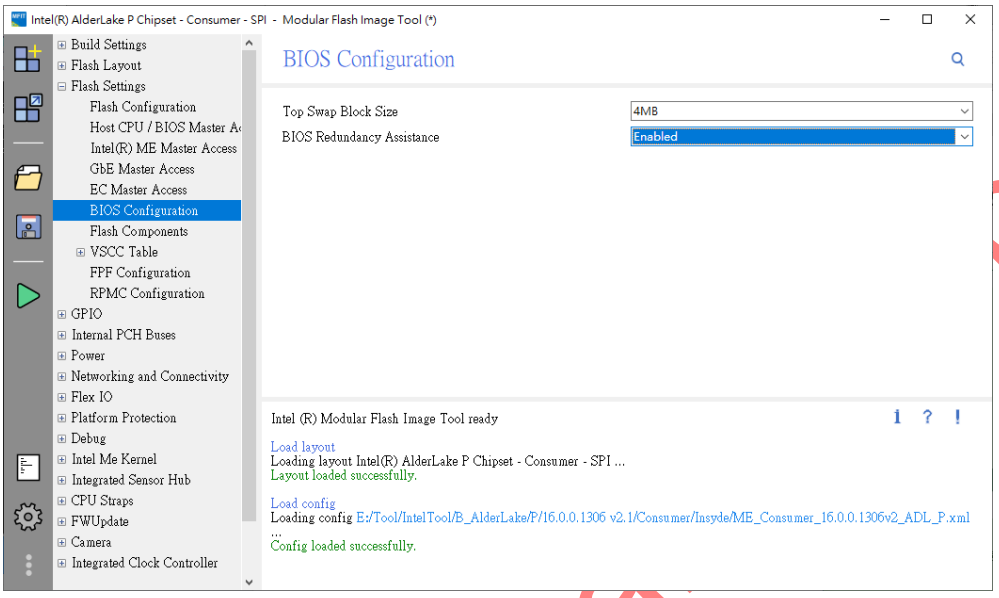
27.8 CSME Setting

CSME needs to enable some items if Chasm falls enable.

27.8.1 Bios

Bios Redundancy Assistance:

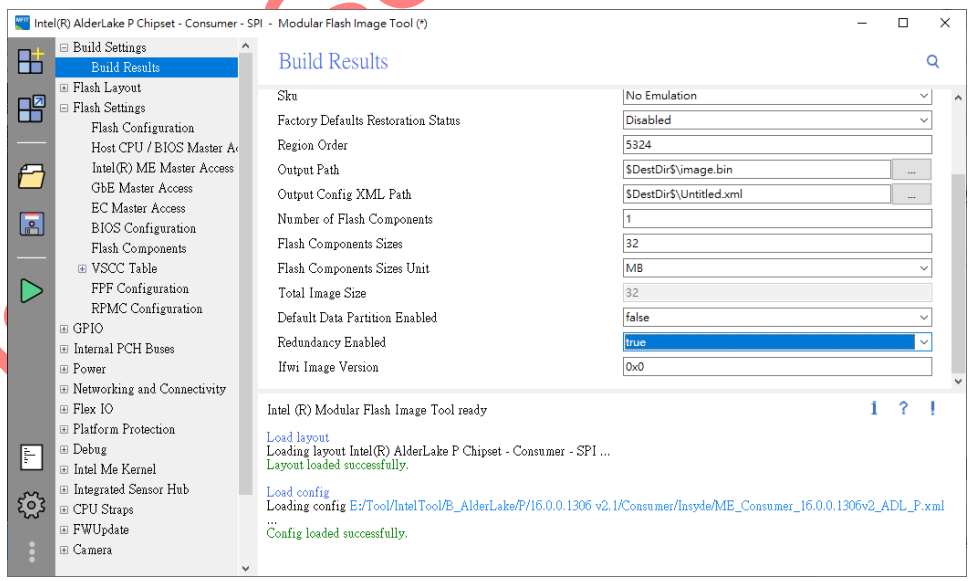
This item **only for Gen 2**. In case of BIOS boot failure, ME will configure the platform to boot with backup BIOS using Top Swap. **This option is only available when "Boot Guard" is enabled.**



27.8.2 ME

Redundancy Enabled:

This item needs to be opened if pcd "PcdMeResiliencyEnable" is enabled.This item will allow to build two boot critical partitions in ME. If boot critical 1 broken, it can boot from boot critical 2.



27.9 How to create update image (Bios Guard Disabled)

27.9.1 Bios

Same as normal capsule/secure flash image. The H2OFFT version at lest above InsydeH2OFFT_x86_WIN_Package_2.01.08.00.zip.

Platform.ini setting:

(i) For capsule update, it is same as normal capsule.

(ii) **For Secure flash, "viaESP" only support 1. (Write image to ESP)**

```
; Supports on WIN Shell flash.
[SecureUpdate]
viaESP=1
ViaESPCopyFileWaitingTime=0
FreeEspSpaceWhenNeed=1
DeviceOrder=eMMC, NVMe, SATA, ATAPI, USB
PhysicalMemoryAllocateFailRetryTimes=3
PhysicalMemoryAllocateFailRetryDelay=500
```

27.9.2 Microcode

Microcode has two update mode, which are Slot mode and Full mode.

Slot mode: To flash **One** microcode that it is match with board.
(ex: If the CPU is b-stepping, the capsule microcode must b-stepping.)
Full mode : To flash entire microcode FV.

1. Generate Microcode capsule FV binary:

(1) Put Microcode binary into "Board\Intel\AlderLakeXMultiBoardPkg\CapsuleUpdate\Binary". **The file extension of microcode needs to use "mcb".** MicrocodeConverter.py can convert "inc" to "pdb". Then manually change "pdb" to "mcb".

(2) Update MicrocodeFv.fdf (Board\Intel\AlderLakeXMultiBoardPkg\CapsuleUpdate)
To build microcode capsule FV. Notice that, the uCode FFS file GUID and MicrocodeVersion.data FFS file GUID are same as [FV.MICROCODEFV] in Project.fdf. If OEM changed these two FFS file GUID in Project.fdf, MicrocodeFv.fdf must be changed too.

For example:

(i) Slot mode:

```
#
#  1. uCode FFS file GUID {197DB236-F856-4924-90F8-CDF12FB875F3} is specific.
#     to help binary tools to locate uCode Array.
#  2. Starting address of Microcode Array is aligned at 4KB alignment.
#     PcdFlashMicrocodeOffset must also set to 4KB accordingly.
#  3. Platform use Slot mode to avoid future FIT update in case of uCode size
#     change, each uCode patch below is placed in a Slot with reserved space.
#  4. uCode patches defined below will be padded and packed for slot mode
#     GenMicrocodeVersion.py in NewGenCap.py performs padding job, with default SLOT_SIZE set to 0x3B000.
#  5. Use MicrocodeConverter.py to comvert ucode from inc to pdb file extension,
#     then modify pdb file extension to mcb.
#
FILE RAW = 197DB236-F856-4924-90F8-CDF12FB875F3 {
    #
    # gIntelMicrocodeArrayFfsFileGuid
    #
    Align=4K $(PROJECT_PKG)\CapsuleUpdate\Binary\m_82_90671_00000017.mcb # ADL-P J1
  }
```

(ii) Full mode:

```
#
#  1. uCode FFS file GUID {197DB236-F856-4924-90F8-CDF12FB875F3} is specific.
#     to help binary tools to locate uCode Array.
#  2. Starting address of Microcode Array is aligned at 4KB alignment.
#     PcdFlashMicrocodeOffset must also set to 4KB accordingly.
#  3. Platform use Slot mode to avoid future FIT update in case of uCode size
#     change, each uCode patch below is placed in a Slot with reserved space.
#  4. uCode patches defined below will be padded and packed for slot mode
#     GenMicrocodeVersion.py in NewGenCap.py performs padding job, with default SLOT_SIZE set to 0x3B000.
#  5. Use MicrocodeConverter.py to comvert ucode from inc to pdb file extension,
#     then modify pdb file extension to mcb.
#
FILE RAW = 197DB236-F856-4924-90F8-CDF12FB875F3 {
    #
    # gIntelMicrocodeArrayFfsFileGuid
    #
    Align=4K $(PROJECT_PKG)\CapsuleUpdate\Binary\m_82_90671_00000017.mcb # ADL-P J1
    Align=16 $(PROJECT_PKG)\CapsuleUpdate\Binary\m_82_906a0_0000000f.mcb # ADL-P J0/J1
  }
```

(3) Put OpenSSL into "AlderLakePlatSamplePkg\Features\CapsuleUpdate\Tools\GenerateCapsule\OpenSSL".

(4) Setting Microcode Capsule information in "BuildCapsule.bat".
(Board\Intel\AlderLakeXMultiBoardPkg\CapsuleUpdate)

Only needs to modify these three items in below. **"**LSV" at onboard MicrocodeVersion.data is the major one. "SLOT_SIZE " **must same as define in MicrocodeUpdatesRes.inf's.** Therefor, this two items usually same as onboard's.

(i) FW_VERSION : New version of MicrocodeVersion.data.
(ii) FW_VERSION_STRING : New microcode version string.
(iii) UCODE_MODE : Microcode update Mode. (ucodefull or ucodeslot)

```
@REM
@REM  Microcode setting
@REM
set SLOT_SIZE=0x3B000
set FW_VERSION=0x0002
set LSV=0x0001
set FW_VERSION_STRING="Version 0.0.0.2"
@REM
@REM UC0DE_MODE = (ucodefull | ucodeslot)
@REM If bios guard enable, the UC0DE_MODE only support ucodefull mode.
@REM
set UCODE_MODE=ucodefull
```

(5) Press ProjectBuild.bat in "Board\Intel\AlderLakeXMultiBoardPkg".

(6) Cd "Board\Intel\AlderLakeXMultiBoardPkg\CapsuleUpdate folder"

Notice that, if the platform isn't AlderLake, below item needs to be changed in BuildCapsule.bat.

```
set PLATFORMSAMPLE_PACKAGE=AlderLakePlatSamplePkg
```

(7) Command: BuildCapsule.bat ucode

Microcde capsule image will be created under "Board\Intel\AlderLakeXMultiBoardPkg\CapsuleUpdate\Output"

Slot mode: UcodeSlotModeFv.bin
Full mode : UcodeFullModeFv.bin

2. Sign Microcode capsule image:

Insyde H2OFFT at lest above InsydeH2OFFT_x86_EFI_Package_2.00.12.00.zip.

(1) Slot mode:

Signed command: "iEFIFlashSigner.exe signbin -n "QA Certificate" -microcode UcodeSlotModeFv.bin"

(2) Full mode:

Signed command: "iEFIFlashSigner.exe signbin -n "QA Certificate" -microcode UcodeFullModeFv.bin"

27.9.3 ME

Same as normal ME capsule image.

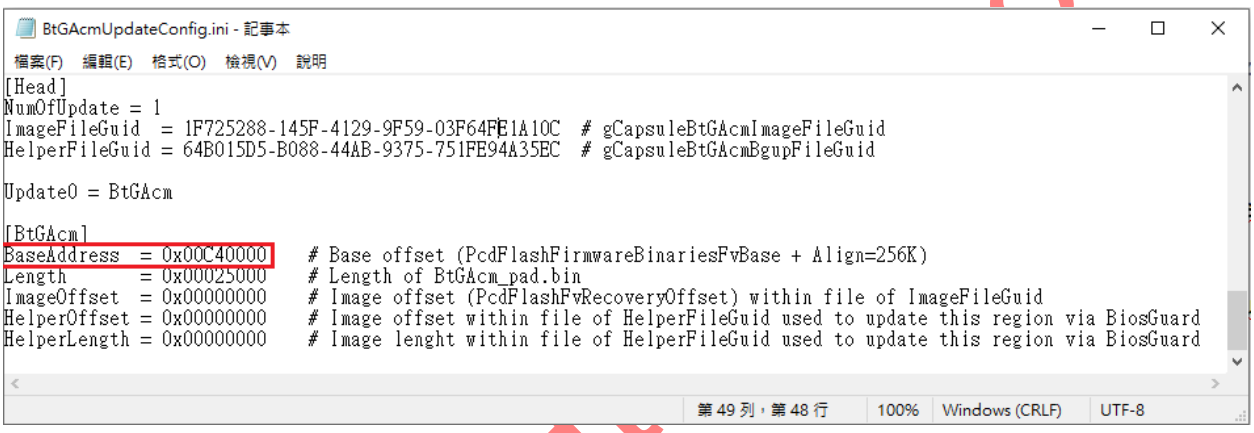
27.9.4 Monolithic

Default combination is BIOS + ME. Therefor, it is same as normal secure flash with BIOS + ME.
The H2OFFT version at lest above InsydeH2OFFT_x86_WIN_Package_2.01.08.00.zip.

27.9.5 Boot Guard Acm (BtGAcm)

(1) Put BtGAcm binary into "Board\Intel\AlderLakeXMultiBoardPkg\CapsuleUpdate\Binary" and rename to BiosAc.bin

(2) Update BaseAddress with BtGAcm Address at BtGAcmUpdateConfig.ini (Board\Intel\AlderLakeXMultiBoardPkg\CapsuleUpdate)



(3) Put OpenSSL into "AlderLakePlatSamplePkg\Features\CapsuleUpdate\Tools\GenerateCapsule\OpenSSL".

(4) Press ProjectBuild.bat in "Board\Intel\AlderLakeXMultiBoardPkg".

(5) Cd "Board\Intel\AlderLakeXMultiBoardPkg\CapsuleUpdate folder"
Notice that, if the platform isn't AlderLake, below item needs to be changed in BuildCapsule.bat.

```
set PLATFORMSAMPLE_PACKAGE=AlderLakePlatSamplePkg
```

(6) Command: BuildCapsule.bat btgacm

BtgAcm capsule image will be created under "Board\Intel\AlderLakeXMultiBoardPkg\CapsuleUpdate\Output"
SystemFwBtGAcm.Cap

27.10 How to create update image (Bios Guard Enabled)

H2OFFT for windows version at lest above InsydeH2OFFT_x86_WIN_Package_2.01.11.00.
H2OFFT for shell version at lest above InsydeH2OFFT_x86_SHELL_Package_2.00.13.00.

27.10.1 Bios

1. Put AlderLakeP.fd\AlderLakeS.fd into "Board\Intel\AlderLakeXMultiBoardPkg\PlatformConfig\BiosGuard".

2. Update BiosGuardSetting.ini (Board\Intel\AlderLakeXMultiBoardPkg\PlatformConfig\BiosGuard)

(1) The setting of bios part are same as normal bios update. You can refer to chapter 22.

(2) Blow list the items for Chasm Falls.

For Gen1:
(I) ChasmFallsType : Gen1 = 0x1.
(II) PbbOffset : The offset of PBB. According to Project.fdf, the PBB offset is 0xC00000.
(III) PbbSize : The size of PBB. According to Project.fdf, the PBB size is 0x400000.

```
[ChasmFalls]
  ;;
  ;; The data in this section will auto be updated in Postbuild Process
  ;;
  ;; ChasmFallsType : Chasm Fall is Disabled , or type is Gen1 or Gen2. (PcdChasmFallsSupport)
  ;;                  (Disable = 0, Gen1 = 0x1, Gen2 = 0x2. ChaseFalls didn't support Full Image, please set type as 0x0.)
  ;; PbbOffset      : PBB BIOS Offset.  (PcdFlashPbbBase  - PcdBiosAreaBaseAddress)
  ;; PbbSize        : PBB/PBBr size.    (PcdFlashPbbSize)
  ;;
  ChasmFallsType = 0x1
  PbbOffset = 0xc00000
  PbbSize = 0x400000
```

For Gen2:
(I) ChasmFallsType : Gen2 = 0x2.
(II) PbbOffset : The offset of PBB. According to Project.fdf, the PBB offset is 0xC00000.
(III) PbbSize : The size of PBB. According to Project.fdf, the PBB size is 0x400000.

```
[ChasmFalls]
  ;;
  ;; The data in this section will auto be updated in Postbuild Process
  ;;
  ;; ChasmFallsType : Chasm Fall is Disabled , or type is Gen1 or Gen2. (PcdChasmFallsSupport)
  ;;                  (Disable = 0, Gen1 = 0x1, Gen2 = 0x2. ChaseFalls didn't support Full Image, please set type as 0x0.)
  ;; PbbOffset      : PBB BIOS Offset.  (PcdFlashPbbBase  - PcdBiosAreaBaseAddress)
  ;; PbbSize        : PBB/PBBr size.    (PcdFlashPbbSize)
  ;;
  ChasmFallsType = 0x2
  PbbOffset = 0xc00000
  PbbSize = 0x400000
```

3. Run BiosGuardImage.bat in "Board\Intel\AlderLakeXMultiBoardPkg\PlatformConfig\BiosGuard".
The bios image with Bios Guard header is BiosGuard_BIOS.fd.

4. Sign Bios capsule:
Signed command: "iEFIFlashSigner.exe signbin -n "QA Certificate" -bios BiosGuard_BIOS.fd"

27.10.2 Microcode

1. Generate Microcode capsule FV binary. This is same as Bios Guard disables, you can refer to chapter 27.8.2.

2. Add Bios Guard header to microcode capsule FV binary:

(1) Put BiosGuardUcodeFv.bin into "Board\Intel\AlderLakeXMultiBoardPkg\PlatformConfig\BiosGuard".

BiosGuardUcodeFv.bin is created by step 1. The file is under "Board\Intel\AlderLakeXMultiBoardPkg\CapsuleUpdate\Output".

(2) Update BiosGuardSetting.ini (Board\Intel\AlderLakeXMultiBoardPkg\PlatformConfig\BiosGuard)

The setting of bios part are same as normal bios update. You can refer to chapter 22.

Blow list the items for Chasm Falls:

(i) FileName : The image file name which need to add Bios Guard header, BiosGuardUcodeFv.bin.
(ii) PackageMode : This image is bios or EC or other firmware. For microcode, it is package firmware.
(ii) FwSize : Fw size. For microcode, this is FLASH_REGION_FV_MICROCODE_SIZE.
(iii) FwOffset : SPI address of firmware. For microcode, this address is FLASH_BASE + FLASH_REGION_FV_MICROCODE_OFFSET.

```
[Image]
  ;;
  ;;  FileName : Image file name, (ex.TigerLake.fd, BiosGaurdUcodeFv.bin)
  ;;  BiosGuardScriptFile : BIOS Guard Script Language File(Use Intel BIOSGuardSL2Bin.exe, transfer *.bgsl to *.bin)
  ;;
  FileName = BiosGaurdUcodeFv.bin
  BiosGuardScriptFile = BGSL\InsydeBiosGuardScript.bin

  ;;
  ;;  PackageMode:
  ;;     0x1 : Package BIOS Image or Full Image.
  ;;     0x2 : Package EC (Non-Shared flash rom)
  ;;     0x3 : Package Firmware (ex: microcode)
  ;;
  PackageMode = 0x3
```

```
  ;;
  ;;  PackageMode= 0x3, Package Firmware (ex: microcode full mode only)
  ;;
  ;;  FwSize : Fw size.
  ;;  FwOffset : Fw firmware address (base on BIOS region)
  ;;             (Chasmfalls gen2 support microcode update)
  ;;
  FwSize = 0x00080000
  FwOffset = 0x0C80000
```

(3) Run BiosGuardImage.bat in "Board\Intel\AlderLakeXMultiBoardPkg\PlatformConfig\BiosGuard".

The microcode FV with Bios Guard header is "BiosGuard_FW.bin".

3. Sign Microcode capsule:

Signed command: "iEFIFlashSigner.exe signbin -n "QA Certificate" -microcode BiosGuard_FW.bin"

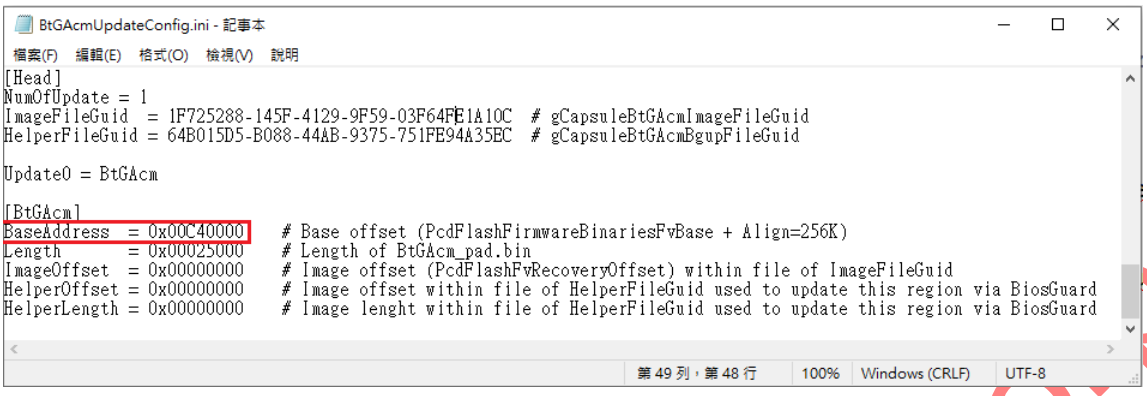
27.10.3 Monolithic

Default combination is BIOS + ME. Only needs to add Bios Guard header to Bios image. You can refer to chapter 27.9.1. The ME firmware don't needs to add Bios Guard header.

27.10.4 Boot Guard Acm (BtGAcm)

28 Put BtGAcm binary into "Board\Intel\AlderLakeXMultiBoardPkg\CapsuleUpdate\Binary" and rename to BiosAc.bin

29 Update BaseAddress with BtGAcm Address at BtGAcmUpdateConfig.ini (Board\Intel\AlderLakeXMultiBoardPkg\CapsuleUpdate)



30 Put OpenSSL into "AlderLakePlatSamplePkg\Features\CapsuleUpdate\Tools\GenerateCapsule\OpenSSL".

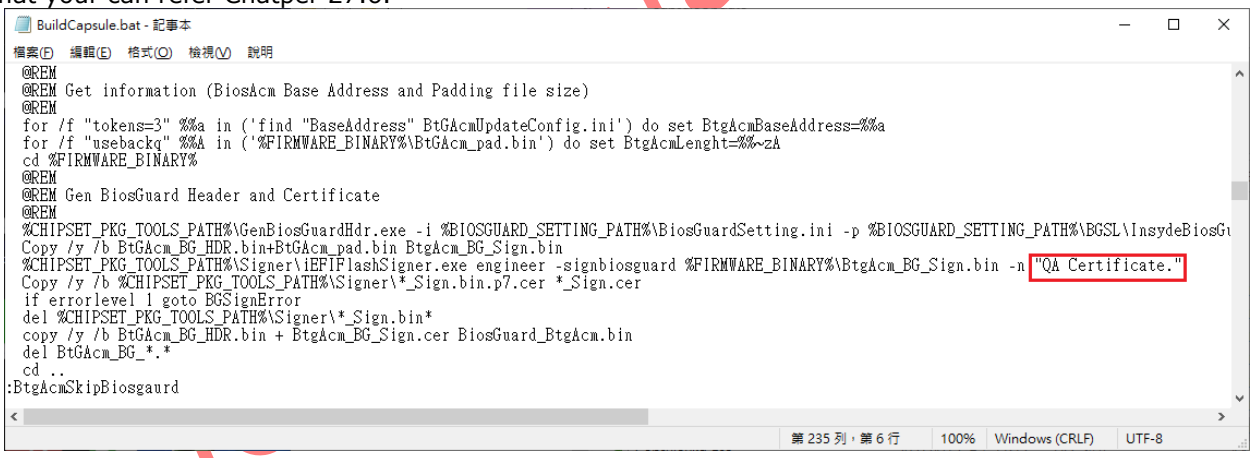
31 Press ProjectBuild.bat in "Board\Intel\AlderLakeXMultiBoardPkg".

32 Cd "Board\Intel\AlderLakeXMultiBoardPkg\CapsuleUpdate folder"

1. If the platform isn't AlderLake, below item needs to be changed in BuildCapsule.bat.

```
set PLATFORMSAMPLE_PACKAGE=AlderLakePlatSamplePkg
```

2. Make sure signtool exist, you can refer Chatper 27.6 (2)Microsoft Signtool.
3. Change your key or certificate generate tool to create Bios Guard Certificate. The certificate format your can refer Chatper 27.6.



33 Command: BuildCapsule.bat btgacmbg

BtgAcm capsule image will be created under "Board\Intel\AlderLakeXMultiBoardPkg\CapsuleUpdate\Output"
SystemFwBtGAcm.Cap

28 Thunderbolt Retimer

This section describes how to update TBT retimer firmware on Alder Lake P RVP and relate codebase.

29.1 Supported BIOS/Firmwares/Tools

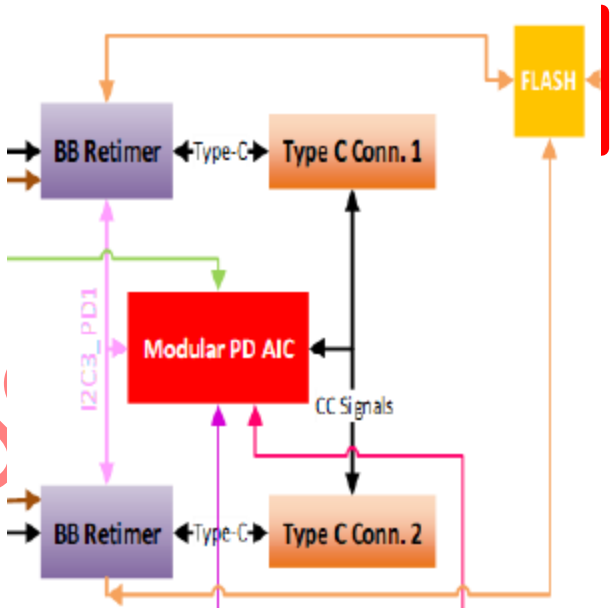
Need to check the version of related files. *The PD firmware or EC firmware can be different from RVP's, so OEM needs to verify if your firmware can be supported with Intel TBT retimer update library.*

A. ADL BIOS - Rev 05.43.10.0010 (IB17800122) and above.

B. H2O SecurityFlash - Rev. 2.00.10.00 and above.

C. Make sure your retimer is connected under ME. Please reference RVP board design.

Example:



29.2 How to generate retimer capsule image

H2OFFT command:

iEFIFlashSigner.exe signbin -n "QA Certificate." -retimer xxxxxx.bin -ver a -funcnum b

Parameter:
(1) -retimer : To sign retimer firmware.

(2) -ver : Retimer version. a = Decimal number.

(3) -funcnum : b= Retimer function. Retimer device 1 = 0x2, Retimer device 2 = 0x3

Hexadecimal number.

For example,

Retimer_BBR_CDR_A1_ADL_P_LP4_PORTS_0_1_rev3_4_SEC1.bin is for retimer device 1 .

Retimer_BBR_CDR_A1_ADL_P_LP4_PORTS_2_3_rev3_4_SEC1.bin is for retimer device 2.

A. If retimer firmware is Retimer_BBR_CDR_A1_ADL_P_LP4_PORTS_0_1_rev3_4_SEC1.bin :

iEFIFlashSigner.exe signbin -n "QA Certificate." -retimer

Retimer_BBR_CDR_A1_ADL_P_LP4_PORTS_0_1_rev3_4_SEC1.bin -ver 304 -funcnum 2

B. If retimer firmware is Retimer_BBR_CDR_A1_ADL_P_LP4_PORTS_2_3_rev3_4_SEC1.bin :

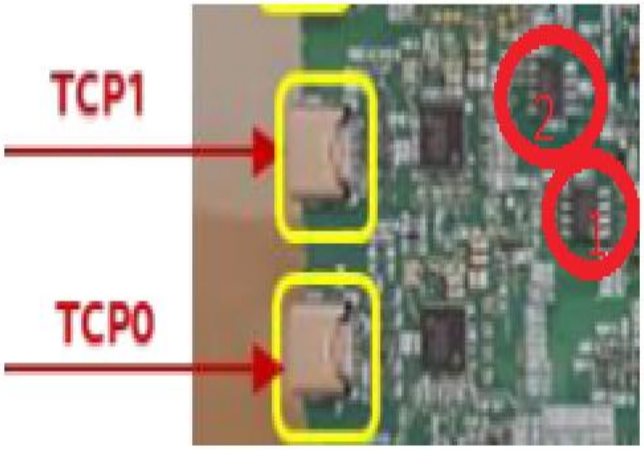
iEFIFlashSigner.exe signbin -n "QA Certificate." -retimer

Retimer_BBR_CDR_A1_ADL_P_LP4_PORTS_2_3_rev3_4_SEC1.bin -ver 304 -funcnum 3

29.3 How to update TBT retimer

1. Use Dediprog ISP test clip (SO8) to flash retimer ROM part.

(ex:ADL-P (1) is retimer device 1 for TCP0 and TCP1.The Flash Rom shared between TCP0/1)



2. Use capsule service to update TBT retimer.

Steps:

A. Make sure the device firmware that you want to be updated has no yellow bang.

B. Follow chapter 28.2 to sign capsule image.

C. The test procedures for retimer capsule update is same as any other firmware or BIOS capsule update. You can refer to the test procedure of QA.

D. Below Picture is a sample of inf file in windows SDK sign package.

As shown in the below figure, there are two items need to fill in.

Red box (1): This is the version of retimer that ready to be updated. This value is **hexadecimal**. The sample show the update version is 304(0x130).

Red box (2): This is a GUID to recognize capsule firmware. If user needs to update retimer device 1, this value will be `PcdWindowsTbtRetimer1FirmwareCapsuleGuid`. If user needs to update retimer device 2, the value will be `PcdWindowsTbtRetimer2FirmwareCapsuleGuid`. This sample is use `PcdWindowsTbtRetimer1FirmwareCapsuleGuid`.

```
CatalogFile = default.cat
PnpLockdown = 1

[Manufacturer]
%MfgName% = Firmware,NTx86,NTamd64,NTarm

[Firmware.NTx86]
%FirmwareDesc% = Firmware_Install,UEFI\RES_{%RES_GUID%}

[Firmware.NTamd64]
%FirmwareDesc% = Firmware_Install,UEFI\RES_{%RES_GUID%}

[Firmware.NTarm]
%FirmwareDesc% = Firmware_Install,UEFI\RES_{%RES_GUID%}

[Firmware_Install.NT]
CopyFiles = Firmware_CopyFiles

[Firmware_CopyFiles]
InsydeSystemFirmware.bin

[Firmware_Install.NT.Hw]
AddReg = Firmware_AddReg

[Firmware_AddReg]
HKR,,FirmwareId,,"{"%RES_GUID%"}"
HKR,,FirmwareVersion,%REG_DWORD%,0x130
HKR,,FirmwareFilename,,InsydeSystemFirmware.bin

[SourceDisksNames]
1 = %DiskName%

[SourceDisksFiles]
InsydeSystemFirmware.bin = 1

[DestinationDirs]
DefaultDestDir = %DIRID_WINDOWS%,Firmware ; %SystemRoot%\Firmware

[Strings]
; localizable
Provider     = "Insyde Software"
MfgName      = "Insyde"
FirmwareDesc = "System Firmware"
DiskName     = "Firmware Update"

; non-localizable
DIRID_WINDOWS = 10
REG_DWORD     = 0x00010001
RES_GUID      = 832af090-2ef9-7c47-8f6d-b405c8c7f156
```

1

2

29.4 Related files

Below list the related files for Insyde retimer capsule service.

1. Retimer device 1~2 main capsule driver.

   $(CHIPSET_PKG)\CapsuleIFWU\CapsuleTbtRetimer1Dxe

   $(CHIPSET_PKG)\CapsuleIFWU\CapsuleTbtRetimer2Dxe

2. A library to initialize and update Retimer by using TBT port.

   $(PLATFORM_SAMPLE_CODE_DEC_NAME)\Features\CapsuleUpdate\Library\TbtRetimerNvmUpdateLib

3. A protocol to set PD firmware controller mode. This protocol will use EcTcssLib to send EC command. ***The instance of EcTcssLib library should change by project.***

   $(PLATFORM_SAMPLE_CODE_DEC_NAME)\Features\TcssRetimerCapsuleUpdate\RetimerCapsuleSupportDriver

29.5 Related Pcds/Guids

Below list the related pcds and guids for Insyde retimer capsule service.

Pcd:

1. **PcdTbtRetimerGetVerDuringBoot**

   To get retimer version in first boot. Default value is FALSE to reduce post time.

2. **PcdTbtRetimerDefaultVersion**

   To set default version if **PcdTbtRetimerGetVerDuringBoot** is FALSE. Default value is 0xCF.

3. **PcdRetimerCapsuleUpdateSupported**

   To enable or disable retimer capsule service. Default value is TRUE.

4. **PcdLowestSupportedRetimerVersion**

   The lowest support version for ESRT table. Default value is 0.

Guid:

1. **gTbtRetimer1VerGuid, gTbtRetimer2VerGuid**

   Variable guid for saving current retimer version.

2. **PcdWindowsTbtRetimer1FirmwareCapsuleGuid,**
   PcdWindowsTbtRetimer2FirmwareCapsuleGuid,

   Capsule firmware guid for windows capsule update.

29 Hardware Security Test Interface

This section describes JasperLake Hardware Security Test Interface with Boot/Bios Guard.

29.1 Related files and parameters

1. HstiIhvDxe.c
   1.1 SecurityFeatureImplement **HSTI_BYTE0_HARDWARE_ROOTED_BOOT_INTEGRITY** bit is controlled by **PcdBootGuardEnable** after IB17800134.
   1.2 SecurityFeatureImplement **HSTI_BYTE0_SIGNED_FIRMWARE_UPDATE** bit is controlled by **PcdBiosGuardEnable** after IB17800134.

2. $(PROJECT_PKG)/Project.env
   2.1 **BOOT_GUARD_SUPPORT –** enable/disable BOOT GUARD BIOS related code
   2.2 **BIOS_GUARD_SUPPORT** – enable/disable BIOS GUARD BIOS related code

3. $(PROJECT_PKG)/Project.dsc
   3.1 PcdBootGuardEnable – PCD to enable/disable BOOT GUARD BIOS related code, controlled by BOOT_GUARD_SUPPORT
   3.2 PcdBiosGuardEnable – PCD to enable/disable BIOS GUARD BIOS related code, controlled by BIOS_GUARD_SUPPORT

29.2 How to use

To pass HSTI boot guard test, please setting **BOOT_GUARD_SUPPORT and BIOS_GUARD_SUPPORT** to **YES**

If OEM doesn't enable boot guard but still want to pass boot/bios guard test, please follow below step to pass the test.

1. OEM need to have a security feature that equivalent to boot/bios guard.

2. Create HSTI OEM driver.

3. Set **HSTI_BYTE0_HARDWARE_ROOTED_BOOT_INTEGRITY** for Boot Guard and **HSTI_BYTE0_SIGNED_FIRMWARE_UPDATE** for Bios Guard bit in byte0 in OEM FeatureImplemented table.

4. Check OEM security feature is work or not. If work, set **HSTI_BYTE0_HARDWARE_ROOTED_BOOT_INTEGRITY** for Boot Guard and **HSTI_BYTE0_SIGNED_FIRMWARE_UPDATE** for Bios Guard bit in byte0 in OEM FeaturesVerified table.

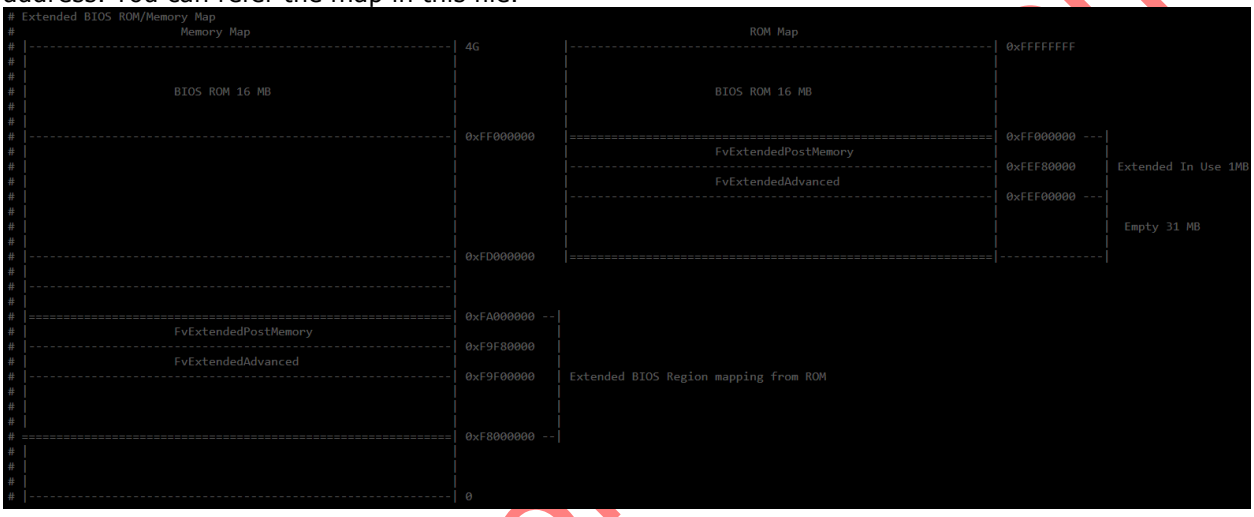
5. Then HSTI can pass without setting BOOT_GUARD_SUPPORT and **BIOS_GUARD_SUPPORT** to YES

30 Extended BIOS Region Customization

This section describes the feature about Extended BIOS Region and how to use/customize on Alder Lake .

30.1 Related files and parameters

1. Project.fdf : There are a lot of definition in FDF file about Extended BIOS Region base and size address. You can refer the map in this file.



2. PeiBootModeLib.c : Notify list for FV installation to the memory for Extended BIOS Region. Each entry of the notify list may need to be registered based on a boot path.
Make sure what FV is installed by each callback and notify them per needs.

3. DxeSmmSpiAccessInitLib.c : Set All FVs with RT attribute in MMIO map, included FvExtendedAdvanced and FvExtendedPostMemory.

4. PlatformInitAdvancedPreMem.c : Report all FVs in MMIO resource hob with RT attribute.

30.2 How to use

1. Set PCD of gPlatformModuleTokenSpaceGuid.PcdExtendedBiosRegionSupport to TRUE in BoardPkg\Package.dsc

2. Adjust the FvExtendedAdvanced and FvExtendedPostMemory size that you want in FDF file.

3. Build nmake uefi64

30.3 How to add new FV in Extended Region

1. Add new FV offset and size in FDF file.

```
DEFINE EXTENDED_TEST_OFFSET      = 0x01E80000
DEFINE EXTENDED_TEST_SIZE        = 0x00080000
```

2. Add new FD Region layout in FDF file.

```
!if gPlatformModuleTokenSpaceGuid.PcdExtendedBiosRegionSupport == TRUE
# ---- TEST
$(EXTENDED_TEST_OFFSET)|$(EXTENDED_TEST_SIZE)
gBoardModuleTokenSpaceGuid.PcdFlashFvExtendedTESTOffset|gBoardModuleTokenSpaceGuid.PcdFlashFvExtendedTESTSize
RegionType = gH2OFlashMapRegionUnknownGuid
FV = FvExtendedTEST
SET gBoardModuleTokenSpaceGuid.PcdFlashFvExtendedTESTBase = $(EXTENDED_REGION_MEMMAP_ADDRESS) + $(EXTENDED_TEST_OFFSET)
# ---- TEST
```

3. Add FV declaration and you driver file path in FDF file.

```
[FV.FvExtendedTESTUncompact]
BlockSize          = 0x10000
FvForceRebase      = FALSE
FvAlignment        = 16
ERASE_POLARITY     = 1
MEMORY_MAPPED      = TRUE
STICKY_WRITE       = TRUE
LOCK_CAP           = TRUE
LOCK_STATUS        = TRUE
WRITE_DISABLED_CAP = TRUE
WRITE_ENABLED_CAP  = TRUE
WRITE_STATUS       = TRUE
WRITE_LOCK_CAP     = TRUE
WRITE_LOCK_STATUS  = TRUE
READ_DISABLED_CAP  = TRUE
READ_ENABLED_CAP   = TRUE
READ_STATUS        = TRUE
READ_LOCK_CAP      = TRUE
READ_LOCK_STATUS   = TRUE
FvNameGuid         = 96E3DE07-3529-48F8-9B9D-4F8139C087AC

!if gPlatformModuleTokenSpaceGuid.PcdExtendedBiosRegionSupport == TRUE
  INF $(PLATFORM_FULL_PACKAGE)/Platform/ValidateExtendedBiosRegion/Dxe/ValidateExtendedBiosTestDxe.inf
!endif
```

```
[FV.FvExtendedTEST]
BlockSize          = 0x10000
FvForceRebase      = FALSE
FvAlignment        = 16
ERASE_POLARITY     = 1
MEMORY_MAPPED      = TRUE
STICKY_WRITE       = TRUE
LOCK_CAP           = TRUE
LOCK_STATUS        = TRUE
WRITE_DISABLED_CAP = TRUE
WRITE_ENABLED_CAP  = TRUE
WRITE_STATUS       = TRUE
WRITE_LOCK_CAP     = TRUE
WRITE_LOCK_STATUS  = TRUE
READ_DISABLED_CAP  = TRUE
READ_ENABLED_CAP   = TRUE
READ_STATUS        = TRUE
READ_LOCK_CAP      = TRUE
READ_LOCK_STATUS   = TRUE
FvNameGuid         = 6B4BED0B-67B9-4C8F-A1EA-04699856574B

!if gPlatformModuleTokenSpaceGuid.PcdExtendedBiosRegionSupport == TRUE
# gFvExtendedTESTFileGuid (9E1CE0CF...) is defined by DEC file
FILE FV_IMAGE = 9E1CE0CF-CCE9-4D4E-82DF-E4E2BA186F00 {
!if gPlatformModuleTokenSpaceGuid.PcdDxeCompressEnable == TRUE
  !if gMinPlatformPkgTokenSpaceGuid.PcdUefiSecureBootEnable == TRUE
    SECTION GUIDED A7717414-C616-4977-9420-844712A735BF PROCESSING_REQUIRED = TRUE AUTH_STATUS_VALID = TRUE {
      SECTION GUIDED EE4E5898-3914-4259-9D6E-DC7BD79403CF PROCESSING_REQUIRED = TRUE { # LzmaCompress/LZMA_CUSTOM_DECOMPRESS_GUID
        SECTION FV_IMAGE = FvExtendedTESTUncompact
      }
    }
  !else
    SECTION GUIDED EE4E5898-3914-4259-9D6E-DC7BD79403CF PROCESSING_REQUIRED = TRUE { # LzmaCompress/LZMA_CUSTOM_DECOMPRESS_GUID
      SECTION FV_IMAGE = FvExtendedTESTUncompact
    }
  !endif
!endif
}
!endif
```

4. Add new FV PCDs of Offset, Size and Base in BoardPkg.dec

```
gBoardModuleTokenSpaceGuid.PcdFlashFvExtendedTESTOffset|0x00000000|UINT32|0x20000A63
gBoardModuleTokenSpaceGuid.PcdFlashFvExtendedTESTSize|0x00000000|UINT32|0x20000A64
gBoardModuleTokenSpaceGuid.PcdFlashFvExtendedTESTBase|0x00000000|UINT32|0x20000A65
```

5. Add new FV Name guid and file guild in PlatformPkg.dec

```
# Used to identify FvExtendedPostMemory on ROM or within BIOS image.
# 21D78FA8-B8E5-4574-9A67-C05B53296936
gFvExtendedPostMemoryFvNameGuid       =  {0x21D78FA8, 0xB8E5, 0x4574, {0x9A, 0x67, 0xC0, 0x5B, 0x53, 0x29, 0x69, 0x36}}

# Used to identify FvExtendedAdvanced on ROM or within BIOS image.
# D18A7412-E2A8-4A45-93B8-AF974DFC7599
gFvExtendedAdvancedFvNameGuid         =  {0xD18A7412, 0xE2A8, 0x4A45, {0x93, 0xB8, 0xAF, 0x97, 0x4D, 0xFC, 0x75, 0x99}}

# ---- TEST
# Used to identify FvExtendedAdvanced on ROM or within BIOS image.
# 6B4BED0B-67B9-4C8F-A1EA-04699856574B
gFvExtendedTESTFvNameGuid             =  {0x6B4BED0B, 0x67B9, 0x4C8F, {0xA1, 0xEA, 0x04, 0x69, 0x98, 0x56, 0x57, 0x4B}}
# ---- TEST
gFvExtendedPostMemoryFileGuid         =  {0xAEE04B33, 0xE2A9, 0x4BD2, {0x90, 0x9F, 0x1C, 0xD2, 0xF3, 0x37, 0xED, 0xAF}}
gFvExtendedAdvancedFileGuid           =  {0xBB43B1EA, 0xDAEA, 0x4F31, {0xB6, 0x98, 0xCE, 0xFF, 0x42, 0x3E, 0x61, 0xE8}}
# ---- TEST
#9E1CE0CF-CCE9-4D4E-82DF-E4E2BA186F00
gFvExtendedTESTFileGuid             =  {0x9E1CE0CF, 0xCCE9, 0x4D4E, {0x82, 0xDF, 0xE4, 0xE2, 0xBA, 0x18, 0x6F, 0x00}}
# ---- TEST
```

6. Add new FV notify list for FV installation to the memory in PeiBootModeLib.c

```
// ---- TEST
EFI_STATUS
EFIAPI
InstallFvExtendedTESTCallback (
  IN EFI_PEI_SERVICES           **PeiServices,
  IN EFI_PEI_NOTIFY_DESCRIPTOR  *NotifyDescriptor,
  IN VOID                       *Ppi
  );
// ---- TEST
/**
  Notify list for FV installation to the memory for Extended BIOS Region.
  Each entry of the notify list may need to be registered based on a boot path.
  Make sure what FV is installed by each callback and notify them per needs.
**/
static EFI_PEI_NOTIFY_DESCRIPTOR mExtendedBiosDecodeReadyNotifyList [] = {
  {
    (EFI_PEI_PPI_DESCRIPTOR_NOTIFY_CALLBACK | EFI_PEI_PPI_DESCRIPTOR_TERMINATE_LIST),
    &gExtendedBiosDecodeReadyPpiGuid,
    InstallFvExtendedPostMemoryCallback
  },
  {
    (EFI_PEI_PPI_DESCRIPTOR_NOTIFY_CALLBACK | EFI_PEI_PPI_DESCRIPTOR_TERMINATE_LIST),
    &gExtendedBiosDecodeReadyPpiGuid,
    InstallFvExtendedAdvancedCallback
  },
// ---- TEST
  {
    (EFI_PEI_PPI_DESCRIPTOR_NOTIFY_CALLBACK | EFI_PEI_PPI_DESCRIPTOR_TERMINATE_LIST),
    &gExtendedBiosDecodeReadyPpiGuid,
    InstallFvExtendedTESTCallback
  },
// ---- TEST
};
```

```
// ---- TEST
EFI_STATUS
EFIAPI
InstallFvExtendedTESTCallback (
  IN EFI_PEI_SERVICES           **PeiServices,
  IN EFI_PEI_NOTIFY_DESCRIPTOR  *NotifyDescriptor,
  IN VOID                       *Ppi
  )
{
#if FixedPcdGetBool(PcdExtendedBiosRegionSupport) == 1
  EFI_FIRMWARE_VOLUME_HEADER  *FvHeader;

  DEBUG ((DEBUG_INFO, "Install FlashFvExtendedTEST - 0x%x, 0x%x\n", PcdGet32 (PcdFlashFvExtendedTESTBase), PcdGet32 (PcdFlashFvExtendedTESTSize)));
  FvHeader = (EFI_FIRMWARE_VOLUME_HEADER *) (UINTN) (UINT32) PcdGet32 (PcdFlashFvExtendedTESTBase);
  PeiServicesInstallFvInfo2Ppi (
    &(FvHeader->FileSystemGuid),
    (VOID *) (UINTN) PcdGet32 (PcdFlashFvExtendedTESTBase),
    PcdGet32 (PcdFlashFvExtendedTESTSize),
    NULL,
    NULL,
    0
    );
  PrintFvHeaderInfo (FvHeader);

#else
  DEBUG ((DEBUG_INFO, "Extended BIOS Region is not supported by the image. No FV installed here\n"));
#endif
  return EFI_SUCCESS;
}
// ---- TEST
```

```
  //[-start-210415-IB19010024-add]//
      Status = PeiServicesNotifyPpi (&mExtendedBiosDecodeReadyNotifyList[1]);
      ASSERT_EFI_ERROR (Status);
  //[-end-210415-IB19010024-add]//
  // ---- TEST
      Status = PeiServicesNotifyPpi (&mExtendedBiosDecodeReadyNotifyList[2]);
      ASSERT_EFI_ERROR (Status);
  // ---- TEST
    }
  //[-start-210415-IB19010024-add]//
    Status = PeiServicesNotifyPpi (&mExtendedBiosDecodeReadyNotifyList[0]);
    ASSERT_EFI_ERROR (Status);
```

7. Add your driver path in DSC file.

```
#
# Extended BIOS Region validation driver
#
!if gPlatformModuleTokenSpaceGuid.PcdExtendedBiosRegionSupport == TRUE
  $(PLATFORM_FULL_PACKAGE)/Platform/ValidateExtendedBiosRegion/Dxe/ValidateExtendedBiosRegionDxe.inf
# ---- TEST
  $(PLATFORM_FULL_PACKAGE)/Platform/ValidateExtendedBiosRegion/Dxe/ValidateExtendedBiosTestDxe.inf
# ---- TEST
!endif
```

8. If everything is done. The driver you add will be dispatched in log.

```
PROGRESS CODE: V03040003 I0
Loading driver ABE48851-6CD3-4381-A80A-E94DB795FCBD
DxeTpm2MeasureBootHandler - Success
InstallProtocolInterface: 5B1B31A1-9562-11D2-8E3F-00A0C969723B 6AA6DA40
Loading driver at 0x00070B1B000 EntryPoint=0x00070B1B374 ValidateExtendedBiosTestDxe.efi
InstallProtocolInterface: BC62157E-3E33-4FEC-9920-2D3B36D750DF 6AA6D818
ProtectUefiImageCommon - 0x6AA6DA40
  - 0x0000000070B1B000 - 0x0000000000001FA0
PROGRESS CODE: V03040002 I0
InstallProtocolInterface: 4C8A2451-C207-405B-9694-99EA13251341 70B1CD60
 ========================================================
 This driver resided in the FV for TEST Extended BIOS Region
 Phase                :  DXE
 This FV was extracted and saved in HOB
 FV HOB Base 6C9F9010
 Actual   FV GUID     :  6B4BED0B-67B9-4C8F-A1EA-04699856574B
 Expected FV GUID     :  6B4BED0B-67B9-4C8F-A1EA-04699856574B
 ========================================================
```

9. If you add a raw data region in FV, not a driver, you should check the FDM address is correct or not.

Appendix A - Dual and Quad Output Read

This appendix describes how to enable Dual and Quad Output Read.

A.1 Prerequisites

To enable Dual and Quad Output Read, INTEL Flash Image Tool (FITC) is required.

A.2 Enabling Dual Output Read

There are 2 ways to enable Dual Output Read.

1. If your Serial Flash Parts support 3Bh command, please follow the steps provided below:
   A. Execute FITC.
   B. Set "Flash Settings"->"Flash Configuration"->"Fast read supported" to "true".
2. If your Serial Flash Parts support SFDP, please follow the steps provided below:
   A. Execute FITC.
   B. Set "Flash Settings"->"Flash Configuration"->"Dual Output Read Enabled" to "true".

Note: this soft strap only has an effect if Dual Output read is discovered as supported via SFDP.

If parameter table is not detected via SFDP, this bit has no effect and Dual Output Read is controlled via the Flash Descriptor Component Section.

A.3 Enabling Quad Output Fast Read

If your Serial Flash Parts support SFDP, please follow the steps provided below:

1. Execute FITC.
2. Set "Flash Settings"->"Flash Configuration"->"Quad Output Read Enabled" to "true".

Note: this soft strap only has an effect if Quad Output read is discovered as supported via SFDP.

Appendix B - Dual and Quad IO Read Support

This appendix describes how to enable Dual and Quad IO Support.

B.1 Prerequisites

To enable Dual and Quad IO Read Support, the SFDP feature of Serial Flash Parts is required.

B.2 Enabling Dual IO Read Support

1. Execute FITC.
2. Set "Flash Settings"->"Flash Configuration"->"Dual I/O Read Enabled" to "true".

Note: This soft strap only has effect if Dual IO Read is discovered as supported via SFDP..

B.3. Enabling Quad IO Read Support

1. Execute FITC.
2. Set "Flash Settings"->"Flash Configuration"->"Quad I/O Read Enabled" to "true".

Note: This soft strap only has effect if Quad IO Read is discovered as supported via SFDP.

B.4. Set "Quad Enable Bit" for Serial Flash Part

For "Quad IO Read" projects, ODMs will prepare the SPI component with "QE Bit" (Quad Enable Bit) enabled by SPI manufacture. For an engineering sample, you can also follow the SPI specification to set QE bit by DediProg in Engineering Mode.

B.5. Trouble Shooting

System hangs on POSTCODE 0x00 after enabling "Quad IO Read" in FITC.

This symptom is always caused by the "QE Bit" or "VSCC Table" not being set properly.

Appendix C - H2ODDT USB3.0 Setup

This appendix describes how to enable H2ODDT by USB3.0 cable.

C.1. Prerequisites

New kernel, debug application, silicon, and USB3.0 debug cable are required to support USB3.0 debug:

1. New version of H2O Kernel 5.4 is verified.
2. New version of H2ODDT application: 100.00.02.01 is verified.
3. CRB and Chipset: xxxxx DDR4/xxxxx LPDDR4 CRBs are verified to support USB3.0 debug.
4. USB ports: All USB3.0 ports are available both on Alder Lake CRB.
5. USB3.0 Debug Cable: USB 3.0 'A' M/M DEBUG CABLE
6. Insyde USB2.0 Debug Cable

C.2. Setup Procedure

1. Modify H2O_DDT_DEBUG_IO from "Usb" to "Xhc" in Project.env to build BIOS for H2ODDT USB3.0 debug interface.
2. Download the latest H2ODDT application which supports USB3.0.

   H2ODDT can be downloaded from X:/Tool/Common/H2ODDT (Developer Debug Tool)
3. Power up CRB and you can see POST code shows D0h/0Dh.
4. Connect USB3.0 debug cable on target platform and host machine.

   (Make sure you connect to CRB & host machine USB 3.0 ports)
5. Connect Insyde USB 2.0 debug cable on host machine. Insyde H2ODDT driver will check Insyde Debug Cable before using USB 3.0 debug interface.
6. After you finish step 4&5, you can see a new USB 3.0 cable in Window Device Manager.
7. Install USB 3.0 driver from H2ODDT application.

You can see POST code changes to D1h and start debugging.

Appendix D - H2ODDT COM PORT Setup

This appendix describes how to enable H2ODDT by COM port cable.

D.1. Prerequisites

Null modem cable is required to support COM port H2ODDT debug:

1. New version of H2O Kernel 5.4 is verified.
2. New version of H2ODDT application: 100.00.02.01 is verified.
3. CRB and Chipset: xxxxx DDR4/xxxxx LPDDR4/xxxxx LPDDR4 CRBs are verified to support COM port debug.
4. Intel CRB support 2 kinds of COM port.
   A. COM port from PCH UART2
   B. COM port from H8 EC

D.2. Com port from PCH UART2 Setup Procedure

1. Modify H2O_DDT_DEBUG_IO from "Xhc" to "Com" and DEBUG_USE_PCH_COMPORT to “Yes” in Project.env to build BIOS for H2ODDT com port debug interface.
2. Run $(CHIPSET_REL_PATH)/$(CHIPSET_PKG)\Override\InsydeModulePkg\H2ODebug\DebugConfig.exe

   Change COM port type = MMIO8, Address = FE042000. This address is PCH UART2 address.

   Build BIOS with “nmake uefi64ddt”

   (Check xxxxxFspBinPkg to see if Fsp_Ddt.fd exists; if not exist, need to build FSP ddt first: BuildFsp.cmd /ddt)
3. Connect Null modem cable to target platform and host machine.
4. Power on.
5. Start H2ODDT application and change project setting from USB to Serial
6. You can see POST code changes to D1h and start debugging.

D.3. Com port from H8 EC Setup Procedure

1. Modify H2O_DDT_DEBUG_IO from "Xhc" to "Com" in Project.env to build BIOS for H2ODDT com port debug interface.
2. Run

$(CHIPSET_REL_PATH)/$(CHIPSET_PKG)\Override\InsydeModulePkg\H2ODebug\DebugConfig.exe

Change com port type = IO, Address = 3F8. This address is H8 EC com port address.

Build BIOS with "nmake uefi64ddt"

(Check xxxxxFspBinPkg to see if Fsp_Ddt.fd exists; if not exist, need to build FSP ddt first: BuildFsp.cmd /ddt)

3. Connect Null modem cable to target platform and host machine.
4. Power on.
5. Start H2ODDT application and change project setting from USB to Serial

You can see POST code changes to D1h and start debugging.