InsydeH2O® 5.4

Alder Lake Hybrid Graphics Implementation Guide

Revision 0.1
June 20, 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 History

Revision Number	Description	Author	Release Date
0.1	Initial release.	Alex Yeh	July 20, 2021

1 Introduction

Targeted Audience and Reading Suggestions:

Target Audience – Insyde H2O 5.3 User.

Reading Suggestion – Readers are recommended to understand H2O 5.3 code base, Secondary System Description Table (SSDT) and Operation Region installation in advance.

1.1 Overview

The purpose of this document is to create a step-by-step proposal that can help InsydeH2O 5.3 BIOS programmers integrate Hybrid Graphics Technology.

The graphics card video driver can provide an applet to switch display or operation to different graphics controller within the system. The graphics controllers supporting Hybrid Graphics Technology have one chipset embedded graphics controller plus one external graphics controller, and the external graphics controller is supposed to populate on the PEG or PCIe bus.

1.2 Related Information

Title	Rev.	Data	Notes
MXM Version 3.0 SBIOS Porting Guide	1.0	February 17, 2009	
MXM Version 3.0 System Design Guide	0.3	August 15, 2008	
MXM Graphics Module Software Specification Version 3.0	1.1	July 20, 2009	
Windows® 8.1 Hybrid Graphics System BIOS Guidance	1.02	July 30, 2013	53640_1.02

1.3 Terminology

Term	Description
ACPI	Advanced Configuration Power Interface
ASL	ACPI Source Language
BACO	Bus alive chip off
CCC	ATI Catalyst™ Control Center
DEC	EDK II meta-data package declaration file. This file declares all public elements of a package containing similar content.
dGPU	Discrete Graphics Processing Unit

DSC	EDK II meta-data platform description file. This file describes what gets built and makes statements that affect how it is built.
EDID	Extended Display Identification Data
FDF	EDK II Flash definition file. This file is used to define the content and binary image layouts for firmware images, update capsules and PCI option ROMs.
GPS	GPU Performance Scaling
HG	Hybrid Graphics
IGD	Intel Integrated Graphics Device
iGPU	Intel® Integrated Graphics Processing Unit
LFP	Local Flat Panel
MUX	Multiplexer
MXM	Mobile PCI Express Module
PCD	Platform Configuration Database
PCH	Platform Controller Hub
PEG	PCI Express* Graphics
SBIOS	System BIOS
SA	System Agent
SG	Switchable Graphics
SIS	System Information Structure
SSDT	Secondary System Description Table
SSID	Sub-System ID
SVID	Sub-System Vendor ID
TPV	Third-party Vendor
VBIOS	Video BIOS
VGA	Video Graphics Adapter
VR	Voltage Regulator

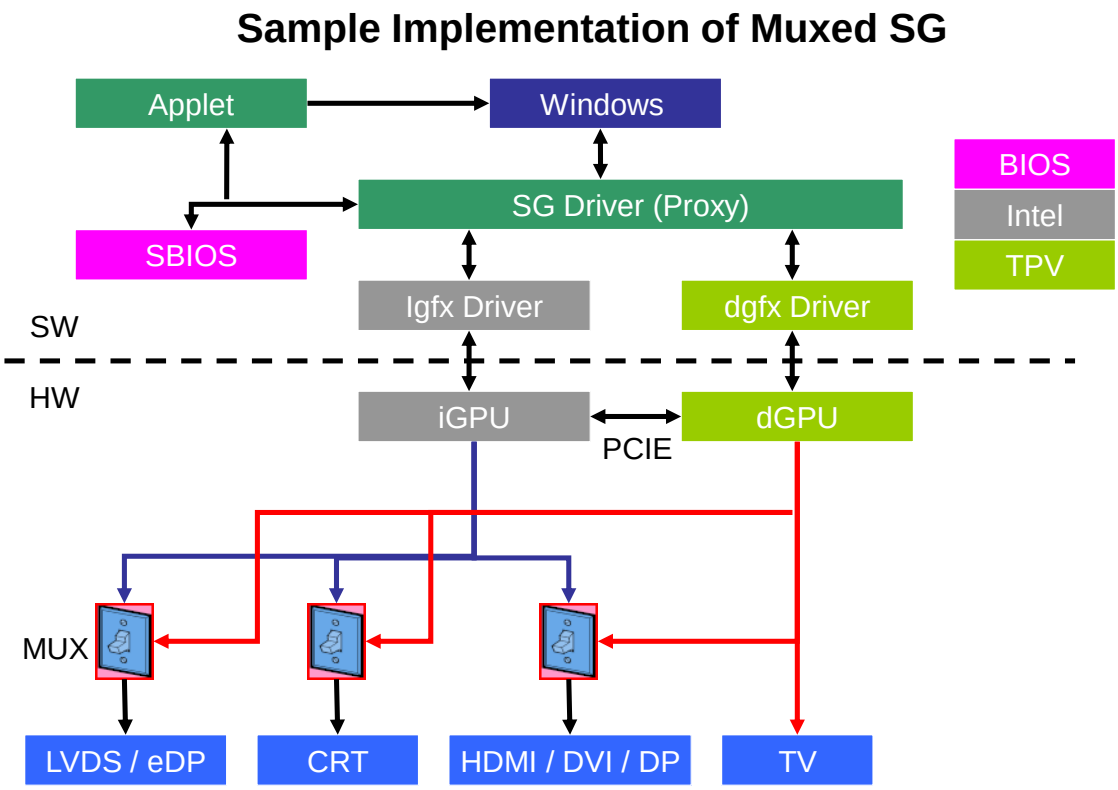
2 Switchable Graphics

What is the Switchable Graphics feature?

The Switchable Graphics feature enables dynamic switching between integrated and discrete graphics device. Integrated graphics devices are typically used when better battery life is required, and discrete graphics devices are typically used when high-end graphics performance is required for applications such as gaming. This is a Microsoft Windows 7 and 8 Operating System-based solution. The Switchable Graphics feature for Intel-based platforms runs on mobile PC systems designed with both an Intel integrated graphics processing unit and a third-party discrete graphics processing unit.

2.1Muxed

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.

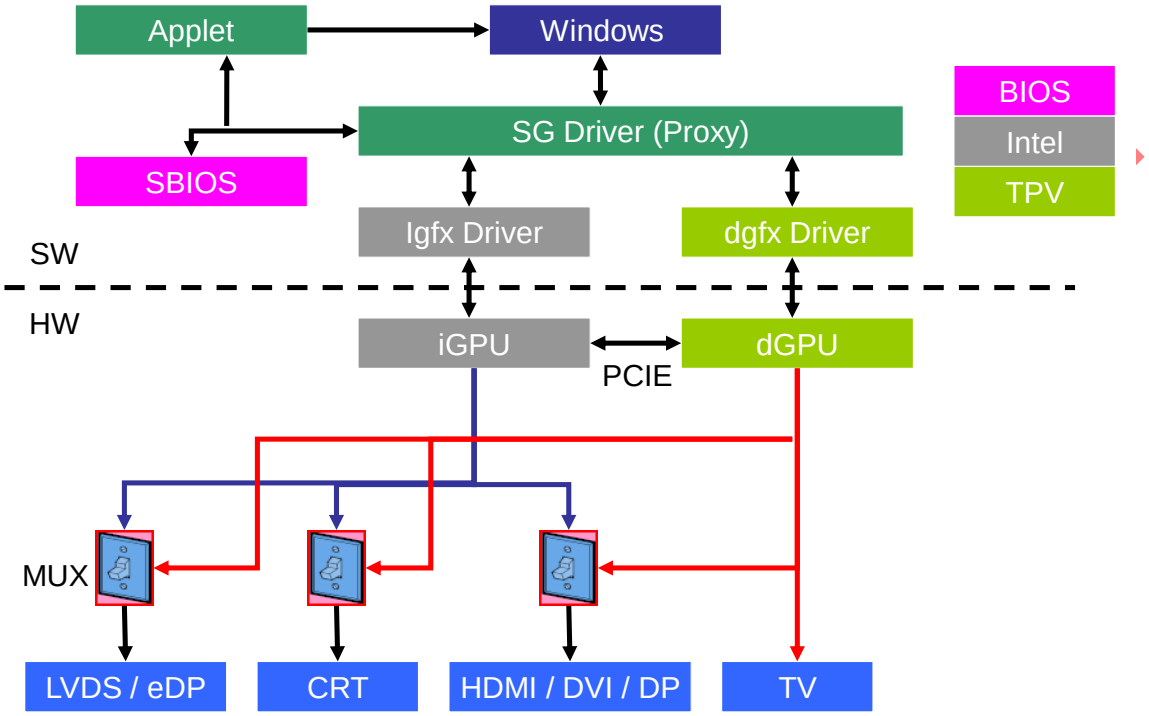


Component	Description
MUX	The output display control can be shared between the iGPU and dGPU through the use of the hardware MUX (Multiplexer).
iGPU	Integrated Graphics Processor Unit, Intel graphics device
dGPU	Discrete Graphics Processor Unit, AMD or Nvidia graphics device
SG Driver	A third-party vendor (TPV) supplied SG proxy driver and the associated applet software control how and when rendering and display responsibilities are switched from one GPU to the other.
Applet	User interface to switch graphics controller.
SBIOS	The system BIOS provides method that is called by SG driver or OS to control the programming of the multiplexers as well as the powering on/off of the discrete GPU.
Note: OS can also intervene to switch the graphics controller for power management schemes	

2.2Muxless

Muxless Switchable Graphics is an initiative which enables dGPU to render a workload then displays through iGPU. No physical Multiplexers required (Cost down).

Sample Implementation of Muxed SG



3 MXM

MXM is a standard interface defined between PC systems and graphics subsystems. MXM covers the mechanical, electrical, thermal and software interface including the connector.

The MXM software specification separates all platform information out of the graphics software into a System Information Structure (SIS) which is stored in the system BIOS (SBIOS). This allows any MXM graphics adapter to work on any MXM compliant system without the need for separate customization. A platform's SBIOS can support any MXM graphics module. A MXM graphics module and a single video BIOS and Graphics Processing Unit (GPU) driver can work on any MXM compliant system. No further customization is needed.

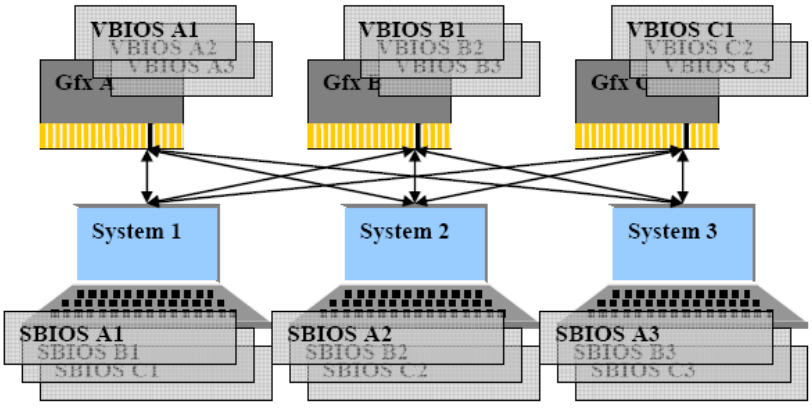
The required steps when you use MXM interface are listed below:

- Create the MXM System Information Structure (Binary file).
- Power-On sequence implementation:

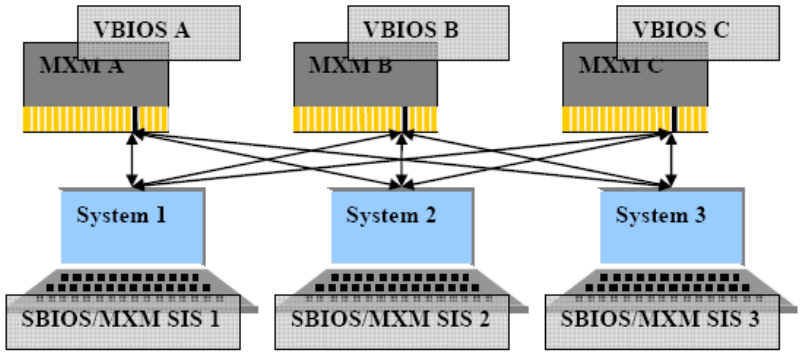
  GPIO initialization phase

  Power-On sequence phase
- SSID/SVID initialization
- SBIOS interfaces implementation
- MXM Int15 callback function (5F80h)
- MXM ASL code (_DOD and _DSM)

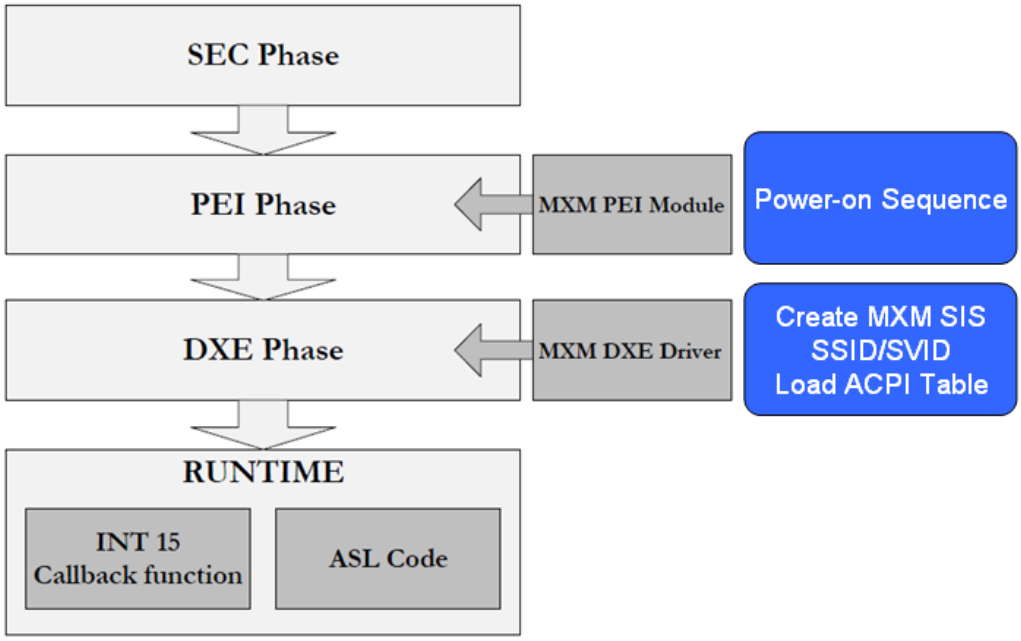
AMD doesn't follow MXM software interface completely, so it doesn't need to implement MXM SIS bin file and MXM Int15 callback function. However, it needs the GPIO power-on sequence and MXM ASL method too. Nvidia follows all MXM software interface by design, so it needs to do all steps in above list.



Non-MXM Software Combination Matrix



MXM Software



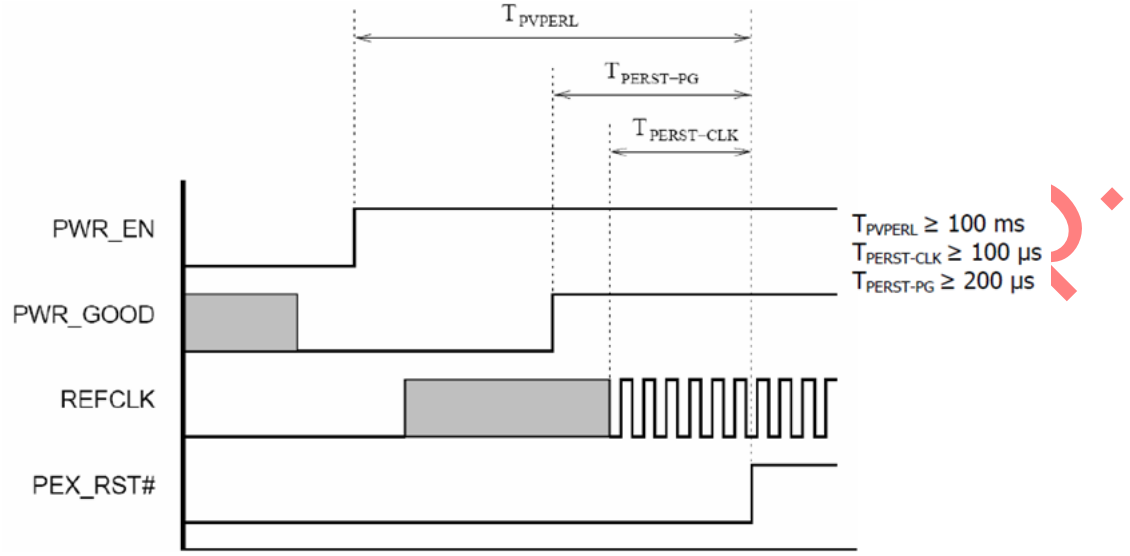
MXM Support in EFI BIOS Architecture

3.1Power-On Sequence

Power-on sequence is project HW dependent.

GPIO assert delay time is GPU device dependent.

GPIO pin high or low active is project HW dependent.



1. PEX_RST# set to low lets it become active
2. Delay some time (100ms)
3. PWR_EN set to high lets it become active
4. Delay some time (300ms)
5. PEX_RST# set to high lets it become inactive
6. Delay some time (100ms)

4 Hybrid Graphics

What is the Hybrid Graphics feature?

The Hybrid Graphics feature is a Microsoft Windows 8.1 Operating System based solution. It shares the same goal as Switchable Graphics Technology. The main difference is that "SG" is mainly controlled by the video driver and "HG" is controlled by the OS.

Hybrid Graphics system definition:

Windows will detect a hybrid graphics configuration at POST time. Any system that matches the following configuration will be considered to be a hybrid system:

1. The system contains a single integrated GPU and a single discrete GPU.
2. It is a design assumption that the discrete GPU has significantly higher performance than the integrated GPU.
3. The discrete GPU is a render-only device. **No display outputs are connected to the discrete GPU.**
4. Both GPUs should be physically enclosed as part of the system. Hot plugging of GPU should not be allowed.

Hybrid systems will have new OS behavior designed to robustly and transparently run applications on either the integrated or discrete GPU, depending on the needs of the application. The OS and graphics driver together determine which GPU a given application uses by default.

5 HG Code Process

This chapter will introduce the code process including build time, PEI, DXE, BDS and SMM phases.

Phase	Behavior	Related File Path
Build Time	Build these files into firmware volume: HybridGraphicsPei.inf HybridGraphicsDxe.inf HybridGraphicsSmm.inf AmdDiscreteSsdt.inf AmdPowerXpressSsdt.inf AmdUltPowerXpressSsdt.inf NvidiaDiscreteSsdt.inf NvidiaUltDiscreteSsdt.inf NvidiaOptimusSsdt.inf NvidiaUltOptimusSsdt.inf MasterMxm30.bin	AlderLakeChipsetPkg\Package.dsc AlderLakeMultiBoardPkg\Project.dsc AlderLakeChipsetPkg\Package.fdf AlderLakeMultiBoardPkg\Project.fdf
	Build Fixed PCDs	AlderLakeChipsetPkg\AlderLakeChipsetPkg.dec
PEI	• Register notify callback function after GPIO table initial for HG initialization(H/S SKU is in Pre-Mem phase, U/Y SKU is in Post-Mem phase; please refer Appendix A) • Install PPI (gH2OHybridGraphicsPpiGuid) for Chipset OEM service (OemSvcMxmDgpuPowerSequence) • Get HG related PCDs • Create HG information data HOB • Set HG mode • MXM dGPU power enable sequence	AlderLakeChipsetPkg\HybridGraphicsPei\HybridGraphicsPei.c AlderLakeChipsetPkg\PlatformInit\PlatformInitPei\PlatformInit.c AlderLakeChipsetPkg\PlatformInit\PlatformInitPei\PlatformInitPreMem.c
	Chipset OEM service for hook HG dGPU power enable sequence and HG information HOB data	AlderLakeChipsetPkg\Library\PeiOemSvcChipsetLib\OemSvcMxmDgpuPowerSequence.c
DXE	• Register HG event notify callback function (HybridGraphicsBdsCallback) • Install HG information Protocol for SMM driver usage.	AlderLakeChipsetPkg\HybridGraphicsDxe\HybridGraphicsDxe.c
BDS	• After connect root bridge, install HG Event Protocol • Trigger HG DXE driver registered callback function	AlderLakeChipsetPkg\Library\PlatformBdsLib\BdsPlatform.c

	• Get HG Information Data HOB • Initial MXM SIS binary, set HG variable and create event to control dGPU HD audio (Nvidia only) • Load and execute dGPU VBIOS (AMD only) • Install SSDT for HG or discrete • Create and initial HG Operation Region • Update HG information Protocol for SMM driver usage • Disable dGPU bridge hot-plug SMI/SCI • Create event to clean secondary GPU command register	AlderLakeChipsetPkg\HybridGraphicsDxe\HybridGraphicsDxe.c
SMM	• Locate HG information Protocol for callback usage • Install INT 15 callback functions • Register NvidiaOptimusHandler callback function	AlderLakeChipsetPkg\HybridGraphicsSmm\HybridGraphicsSmm.c

5.1Build Time

File Path: AlderLakeChipsetPkg/Package.dsc

Description: Build HG PEIM, DXE and SMM drivers, Nvidia/AMD HG mode SSDT and Nvidia/AMD discrete mode SSDT into EFI Firmware File System (FFS) sections.

```
[Components.IA32]
  AlderLakeChipsetPkg/HybridGraphicsPei/HybridGraphicsPei.inf
[Components.X64]
  AlderLakeChipsetPkg/HybridGraphicsDxe/HybridGraphicsDxe.inf
  AlderLakeChipsetPkg/HybridGraphicsSmm/HybridGraphicsSmm.inf
  AlderLakeChipsetPkg/HybridGraphicsAcpi/AmdDiscreteSsdt.inf
  AlderLakeChipsetPkg/HybridGraphicsAcpi/AmdPowerXpressSsdt.inf
  AlderLakeChipsetPkg/HybridGraphicsAcpi/AmdUltPowerXpressSsdt.inf
  AlderLakeChipsetPkg/HybridGraphicsAcpi/N17/NvidiaOptimusSsdt.inf
  AlderLakeChipsetPkg/HybridGraphicsAcpi/N17/NvidiaUltOptimusSsdt.inf
  AlderLakeChipsetPkg/HybridGraphicsAcpi/N17/NvidiaDiscreteSsdt.inf
  AlderLakeChipsetPkg/HybridGraphicsAcpi/N17/NvidiaUltDiscreteSsdt.inf
  AlderLakeChipsetPkg/HybridGraphicsAcpi/N18/NvidiaOptimusSsdt.inf
  AlderLakeChipsetPkg/HybridGraphicsAcpi/N18/NvidiaUltOptimusSsdt.inf
  AlderLakeChipsetPkg/HybridGraphicsAcpi/N18/NvidiaDiscreteSsdt.inf
  AlderLakeChipsetPkg/HybridGraphicsAcpi/N18/NvidiaUltDiscreteSsdt.inf
  AlderLakeChipsetPkg/HybridGraphicsAcpi/N20/NvidiaOptimusSsdt.inf
  AlderLakeChipsetPkg/HybridGraphicsAcpi/N20/NvidiaUltOptimusSsdt.inf
  AlderLakeChipsetPkg/HybridGraphicsAcpi/N20/NvidiaDiscreteSsdt.inf
  AlderLakeChipsetPkg/HybridGraphicsAcpi/N20/NvidiaUltDiscreteSsdt.inf
```

Example: How to override module or drivers?
File Path: AlderLakePMultiBoardPkg/Project.dsc
Description: 1. Add override path in Project.dsc 2. Copy file to override path 3. Modify the override file

[Components.IA32]
!disable AlderLakeChipsetPkg/HybridGraphicsPei/HybridGraphicsPei.inf
AlderLakeChipsetPkg/HybridGraphicsPei/HybridGraphicsPei.inf {
<SOURCE_OVERRIDE_PATH>
$(PROJECT_PKG)/Override/AlderLakeChipsetPkg/HybridGraphicsPei
}
 	
[Components.X64]
!disable AlderLakeChipsetPkg/HybridGraphicsDxe/HybridGraphicsDxe.inf
AlderLakeChipsetPkg/HybridGraphicsDxe/HybridGraphicsDxe.inf {
<SOURCE_OVERRIDE_PATH>
$(PROJECT_PKG)/Override/AlderLakeChipsetPkg/HybridGraphicsDxe
}
 	
!disable AlderLakeChipsetPkg/HybridGraphicsSmm/HybridGraphicsSmm.inf
AlderLakeChipsetPkg/HybridGraphicsSmm/HybridGraphicsSmm.inf {
<SOURCE_OVERRIDE_PATH>
$(PROJECT_PKG)/Override/AlderLakeChipsetPkg/HybridGraphicsSmm
}

Example: How to set HG related feature switch?
File Path: AlderLakePMultiBoardPkg/Project.env
Nvidia Optimus feature support: EDK_GLOBAL HYBRID_GRAPHICS_SUPPORT = YES EDK_GLOBAL NVIDIA_OPTIMUS_SUPPORT = YES EDK_GLOBAL AMD_POWERXPRESS_SUPPORT = NO
AMD Power Xpress feature support: EDK_GLOBAL HYBRID_GRAPHICS_SUPPORT = YES EDK_GLOBAL NVIDIA_OPTIMUS_SUPPORT = NO EDK_GLOBAL AMD_POWERXPRESS_SUPPORT = YES
Nvidia Optimus and AMD Power Xpress features support: EDK_GLOBAL HYBRID_GRAPHICS_SUPPORT = YES EDK_GLOBAL NVIDIA_OPTIMUS_SUPPORT = YES EDK_GLOBAL AMD_POWERXPRESS_SUPPORT = YES

Example: How to set HG related PCDs?
File Path: AlderLakeChipsetPkg\AlderLakeChipsetPkg .dec
Description: HG related PCDs are in declaration (.dec) file (Chipset layer), so project needs based on project design to modify these PCDs in Project.dsc

[PcdsFixedAtBuild]

gChipsetPkgTokenSpaceGuid.PcdUseCrbHgDefaultSettings|TRUE|BOOLEAN|0x2000020d
↑If your project doesn't use CRB default settings, please set this PCD to FALSE.
Note: The following PCDs (List below) will become invalid if this PCD set to TRUE.

gChipsetPkgTokenSpaceGuid.PcdHgDgpuGpioSupport|TRUE|BOOLEAN|0x2000020e
↑If your project doesn't have any GPIO pins to control discrete GPU power, you can set this PCD to FALSE, and add discrete GPU power enable sequence into OemSvcMxmDgpuPowerSequence.c

==========

gChipsetPkgTokenSpaceGuid.PcdHgDgpuWakeGpioNo|0|UINT32|0x2000020f

gChipsetPkgTokenSpaceGuid.PcdHgDgpuPwrOkGpioNo|0|UINT32|0x20000210

gChipsetPkgTokenSpaceGuid.PcdHgDgpuPwrOkActive|TRUE|BOOLEAN|0x20000211

gChipsetPkgTokenSpaceGuid.PcdHgDgpuPwrOkExist|FALSE|BOOLEAN|0x20000212

gChipsetPkgTokenSpaceGuid.PcdHgDgpuHoldRstGpioNo|0x0x090B000D|UINT32|0x20000213

gChipsetPkgTokenSpaceGuid.PcdHgDgpuHoldRstActive|FALSE|BOOLEAN|0x20000214

gChipsetPkgTokenSpaceGuid.PcdHgDgpuSelGpioNo|0|UINT32|0x20000215

gChipsetPkgTokenSpaceGuid.PcdHgDgpuSelActive|FALSE|BOOLEAN|0x20000216

gChipsetPkgTokenSpaceGuid.PcdHgDgpuSelExist|FALSE|BOOLEAN|0x20000217

gChipsetPkgTokenSpaceGuid.PcdHgDgpuPwrEnableGpioNo|0x0902000E|UINT32|0x20000218

gChipsetPkgTokenSpaceGuid.PcdHgDgpuPwrEnableActive|FALSE|BOOLEAN|0x20000219

gChipsetPkgTokenSpaceGuid.PcdHgDgpuEdidSelGpioNo|0|UINT32|0x2000021a

gChipsetPkgTokenSpaceGuid.PcdHgDgpuEdidSelActive|FALSE|BOOLEAN|0x2000021b

gChipsetPkgTokenSpaceGuid.PcdHgDgpuEdidSelExist|FALSE|BOOLEAN|0x2000021c

gChipsetPkgTokenSpaceGuid.PcdHgDgpuPwmSelGpioNo|0|UINT32|0x2000021d

gChipsetPkgTokenSpaceGuid.PcdHgDgpuPwmSelActive|FALSE|BOOLEAN|0x2000021e

gChipsetPkgTokenSpaceGuid.PcdHgDgpuPwmSelExist|FALSE|BOOLEAN|0x2000021f

gChipsetPkgTokenSpaceGuid.PcdHgDgpuPrsntGpioNo|0|UINT32|0x20000220

gChipsetPkgTokenSpaceGuid.PcdHgDgpuPrsntActive|FALSE|BOOLEAN|0x20000221

gChipsetPkgTokenSpaceGuid.PcdHgDgpuPrsntExist|FALSE|BOOLEAN|0x20000222

↑Please based on your Project HW design to set above PCDs which related with GPIO pins. GPIO pin settings include pin number, high (TRUE) or low (FALSE) active and pin exist (TRUE) or not (FALSE).

For example:
The GPIO group definition for AlderLake PCH GPIO (PCH type is ADL-LP):

```
#define GPIO_VER2_LP_GROUP_GPP_B    0x0900
#define GPIO_VER2_LP_GROUP_GPP_T    0x0901
#define GPIO_VER2_LP_GROUP_GPP_A    0x0902
#define GPIO_VER2_LP_GROUP_GPP_R    0x0903
#define GPIO_VER2_LP_GROUP_SPI      0x0904
```

```
#define GPIO_VER2_LP_GROUP_GPD      0x0905
#define GPIO_VER2_LP_GROUP_GPP_S    0x0906
#define GPIO_VER2_LP_GROUP_GPP_H    0x0907
#define GPIO_VER2_LP_GROUP_GPP_D    0x0908
#define GPIO_VER2_LP_GROUP_GPP_U    0x0909
#define GPIO_VER2_LP_GROUP_VGPIO    0x090A
#define GPIO_VER2_LP_GROUP_GPP_C    0x090B
#define GPIO_VER2_LP_GROUP_GPP_F    0x090C
#define GPIO_VER2_LP_GROUP_HVCMOS   0x090D
#define GPIO_VER2_LP_GROUP_GPP_E    0x090E
#define GPIO_VER2_LP_GROUP_JTAG     0x090F
#define GPIO_VER2_LP_GROUP_CPU      0x0910
#define GPIO_VER2_LP_GROUP_VGPIO_3  0x0911
```

Assume we set PcdHgDgpuHoldRstGpioNo to **0x090B000D**:
Red part represent Group.
If you want to know which group:
(0x090B000D & 0x0F1F0000) >> 16
If you want to know which group index:
((0x090B000D & 0x0F1F0000) >> 16) & 0x1F

Blue part represent Pad Number.
If you want to know which pad number in this group:
(0x090B000D & 0x1FF)

==========

gChipsetPkgTokenSpaceGuid.PcdHgDgpu2GpioSupport|FALSE|BOOLEAN|0x20000223

gChipsetPkgTokenSpaceGuid.PcdHgDgpu2WakeGpioNo|0|UINT32|0x20000224

gChipsetPkgTokenSpaceGuid.PcdHgDgpu2PwrOkGpioNo|0|UINT32|0x20000225

gChipsetPkgTokenSpaceGuid.PcdHgDgpu2PwrOkActive|TRUE|BOOLEAN|0x20000226

gChipsetPkgTokenSpaceGuid.PcdHgDgpu2HoldRstGpioNo|0|UINT32|0x20000227

gChipsetPkgTokenSpaceGuid.PcdHgDgpu2HoldRstActive|FALSE|BOOLEAN|0x20000228

gChipsetPkgTokenSpaceGuid.PcdHgDgpu2PwrEnableGpioNo|0|UINT32|0x20000229

gChipsetPkgTokenSpaceGuid.PcdHgDgpu2PwrEnableActive|FALSE|BOOLEAN|0x2000022a

gChipsetPkgTokenSpaceGuid.PcdHgDgpu2PrsntGpioNo|0|UINT32|0x2000022b

gChipsetPkgTokenSpaceGuid.PcdHgDgpu2PrsntActive|FALSE|BOOLEAN|0x2000022c

gChipsetPkgTokenSpaceGuid.PcdHgDgpu2PrsntExist|FALSE|BOOLEAN|0x2000022d

↑If project HW design has two dGPUs like AMD Cross Fire or Nvidia SLI, you need to set slave dGPU related GPIO pins.

Note: For the case of two dGPUs design, related C code or ASL code need to recheck or redesign because dGPU2 related code has not been updated for a long time.

gChipsetPkgTokenSpaceGuid.PcdHgPegBridgeBus|0|UINT8|0x2000022e

gChipsetPkgTokenSpaceGuid.PcdHgPegBridgeDevice|1|UINT8|0x2000022f

gChipsetPkgTokenSpaceGuid.PcdHgPegBridgeFunction|0|UINT8|0x20000230

gChipsetPkgTokenSpaceGuid.PcdHgPcieBridgeBus|0|UINT8|0x20000231

gChipsetPkgTokenSpaceGuid.PcdHgPcieBridgeDevice|28|UINT8|0x20000232

gChipsetPkgTokenSpaceGuid.PcdHgPcieBridgeFunction|4|UINT8|0x20000233

↑Based on HW design to set Bus/Device/Function of dGPU bridge.

If dGPU is on PEG slot, please set PcdHgPegBridgeBus/PcdHgPegBridgeDevice/PcdHgPegBridgeFunction

If dGPU is on PCIe slot, please set PcdHgPcieBridgeBus

Note: Why we comment out PcdHgPcieBridgeDevice and PcdHgPcieBridgeFunction?

Because we need to consider the cases which set RpFunctionSwap to enabled or disabled.

If RpFunctionSwap is enabled:

The root port function number may be swapped to 0 (Ex: 0/28/4 -> 0/28/0)

If RpFunctionSwap is disabled:

The root port function number will not be swapped.

So we think using GetPchPcieRpDevFun() to get corresponding Device/Function is the better way.

Notice:

[HybridGraphicsDxe.c -> HybridGraphicsDxeInitialize()]

The first input parameter of GetPchPcieRpDevFun() is SaConfigHob->SaRtd3.RootPortIndex

Please make sure this value is correct, otherwise this function can't return correct Device/Function.

Generally, RootPortIndex will be set by Board ID.

The code to set RootPortIndex (Use CRB as an example):

[BoardSaInitPreMemLib.c -> AdlPSaGpioConfigInit()]

```
switch (BoardId) {
  ...
  case BoardIdAdlPLp5Rvp::
  case BoardIdAdlPLp4Rvp:
  case BoardIdAdlPDdr5Rvp:
  case BoardIdAdlPDdr4Rvp:
    PcdSet8S(PcdRootPortIndex, 4);
  ...
}
```

gChipsetPkgTokenSpaceGuid.PcdHgDgpu2BridgeBus|0|UINT8|0x20000234

gChipsetPkgTokenSpaceGuid.PcdHgDgpu2BridgeDevice|1|UINT8|0x20000235

gChipsetPkgTokenSpaceGuid.PcdHgDgpu2BridgeFunction|1|UINT8|0x20000236

↑Based on HW design to set Bus/Device/Function of slave dGPU (For the case of two dGPUs design)

gChipsetPkgTokenSpaceGuid.PcdAmdSecondaryGrcphicsCommandRegister|0|UINT8|0x20000237

gChipsetPkgTokenSpaceGuid.PcdNvidiaSecondaryGrcphicsCommandRegister|6|UINT8|0x20000238

↑Secondary GPU must close I/O resource to avoid resource crash (For the case of two dGPUs design)

gChipsetPkgTokenSpaceGuid.PcdHgNvidiaOptimusDgpuHotPlugSupport|TRUE|BOOLEAN|0x20000242

gChipsetPkgTokenSpaceGuid.PcdHgNvidiaOptimusDgpuPowerControlSupport|TRUE|BOOLEAN|0x20000243

gChipsetPkgTokenSpaceGuid.PcdHgNvidiaGpsFeatureSupport|TRUE|BOOLEAN|0x20000244

gChipsetPkgTokenSpaceGuid.PcdHgNvidiaVenturaFeatureSupport|FALSE|BOOLEAN|0x20000245

gChipsetPkgTokenSpaceGuid.PcdHgNvidiaNbciFeatureSupport|FALSE|BOOLEAN|0x20000246

gChipsetPkgTokenSpaceGuid.PcdHgNvidiaOptimusGc6FeatureSupport|FALSE|BOOLEAN|0x20000247

gChipsetPkgTokenSpaceGuid.PcdHgNvidiaOptimusGc6NvsrFeatureSupport|FALSE|BOOLEAN|0x20000248

gChipsetPkgTokenSpaceGuid.PcdNvidiaGC6FBEN|0|UINT32|0x20000249

gChipsetPkgTokenSpaceGuid.PcdNvidiaGPUEvent|0|UINT32|0x2000024a

gChipsetPkgTokenSpaceGuid.PcdHgNvidiaDgpuCheckTable|{0}|VOID*|0x20000253

gChipsetPkgTokenSpaceGuid.PcdHgNvidiaDgpuCheckTable2|{0, 0}|VOID*|0x20000260

gChipsetPkgTokenSpaceGuid.PcdHgNvidiaNpcfFeatureSupport|FALSE|BOOLEAN|0x20000251

↑Nvidia related feature switch, these PCDs will through operation region pass to ASL code.

Note:

Added PcdHgNvidiaNpcfFeatureSupport for N18.

PcdHgNvidiaNpcfFeatureSupport - Indicates if the platform can support NVPCF or not.
TRUE - Support NVPCF
FALSE - Not support NVPCF
After set this PCD to TRUE, system will have one more device (NPCF) in device manager, and this device needs to install another driver otherwise it will appears yellow bang.

File Path: AlderLakeMultiBoardPkg\Project.dsc

Description: Provide PcdHgNvidiaDgpuCheckTable to switch NVIDIA DGPU N17 and N18 dynamically. Use N18 DGPU device ID as sample check method in the PcdHgNvidiaDgpuCheckTable. When table compare current DGPU device ID is match, system will support N18 chip. If you want to use other check method to switch DGPU dynamically, you can fill other values in the PcdHgNvidiaDgpuCheckTable (ex. SKUID). Also provide PcdHgNvidiaDgpuCheckTable2 to switch NVIDIA DGPU N17, N18, N20 or further generation dynamically. If you want to use PcdHgNvidiaDgpuCheckTable2, please unmark it.

[PcdsFixedAtBuild]

gChipsetPkgTokenSpaceGuid.PcdHgNvidiaDgpuCheckTable|{ \

  UINT16(0x1E90), \

  UINT16(0x1F10), \

```
  UINT16(0x1F11), \
  UINT16(0x00)    \
 }
```

↑Default values are DGPU Device ID in the PcdHgNvidiaDgpuCheckTable.

Example:

```
gChipsetPkgTokenSpaceGuid.PcdHgNvidiaDgpuCheckTable|{ \
 UINT8(SKUID1), \
 UINT8(SKUID2), \
 UINT8(SKUID3), \
 UINT8(0x00)    \
 }

#  gChipsetPkgTokenSpaceGuid.PcdHgNvidiaDgpuCheckTable2|{ \
   # DeviceID      # Generation Info
                # Ex: N17 = 0x11, N18 = 0x12, N20 = 0x14, ...
#    UINT16(0x1E90), UINT8(0x11), \
#    UINT16(0x1F10), UINT8(0x12), \
#    UINT16(0x1F11), UINT8(0x14)  \
#  }
```

5.2PEI Phase

File Path: AlderLakeChipsetPkg\HybridGraphicsPei\HybridGraphicsPei.c
Behavior: Register notify callback function for HG initialization. **Description:** Register HybridGraphicsInitNotify() callback function after GPIO table initial and depend on your dGPU is on PEG slot or PCIe slot to decide which notify PPI() should be used (Install gHybridGraphicsReadyForPowerSequenceInit in Pre-Mem phase for S or Post-Mem phase for P).
Behavior: Install HG PPI for Chipset OEM service. **Description:** Install gH2OHybridGraphicsPpiGuid for OemSvcMxmDgpuPowerSequence(). This PPI provides GPIO read/write and stall services.
Behavior: Get HG related PCDs and create HG information data HOB. **Description:** Based on PCDs to set HG related data and through HOB pass to DXE driver.
Behavior: Set HG mode. **Description:** The value of HG mode will depend on "PrimaryDisplay" setting in SaSetup variable. **PrimaryDisplay value definition:** **0 = DisplayModeIgpu** **1 = DisplayModeDgpu** **2 = DisplayModePci** **3 = DisplayModeAuto** **4 =DisplayModeHg**

SG mode value definition for Intel RC: **0 = SgModeDisabled** **1 = SgModeReserved** **2 = SgModeMuxless** **3 = SgModeDgpu** **4 = SgModeMax** If PrimaryDisplay = 4 => Set SG mode to 2. If PrimaryDisplay = 1 or 3 or 2 => Set SG mode to 3. If PrimaryDisplay = 0 => Set SG mode to 0. If InternalGraphics = 0 (IgdDisable) => Set SG mode to 3.
Behavior: MXM dGPU power enable sequence. **Description:** There has OemSvcMxmDgpuPowerSequence() before run default power enable sequence. This OEM service provides a point to let project customize power enable sequence or modify HG information data HOB.

5.3DXE and BDS Phase

File Path: AlderLakeChipsetPkg\HybridGraphicsDxe\HybridGraphicsDxe.c
Behavior: Register HG event protocol notify callback function (HybridGraphicsBdsCallback()). **Description:** After connect root bridge in BDS phase, BDS driver will install gH2OHybridGraphicsEventProtocolGuid to notify HybridGraphicsBdsCallback().
Behavior: Install HG information protocol (gH2OHybridGraphicsInfoProtocolGuid) for SMM driver usage. **Description:** This protocol includes some data for HG SMM callback usage, HG DXE driver will install protocol first, then HG SMM driver will get the location of this protocol and HG DXE driver callback will update protocol data in BDS phase.
Behavior: Get HG information data HOB **Description:** HG PEIM will pass this HOB to HG DXE driver, all data in this HOB should be updated in HG PEIM.
Behavior: Initial MXM SIS binary, set HG variable and create event to control dGPU HDA (Nvidia only) **Description:** Get MXM (Mobile PCI Express Module) SIS (System Information Structure) binary from firmware volume, legacy boot will allocate legacy region and copy MXM SIS for MXM INT 15 callback function, EFI and legacy boot will copy MXM SIS to HG own operation region. Set HG variable for S3/S4 Optimus HDA status. **Note:** <N17> Normal boot: dGPU HDA always should be disabled

S3/S4 resume: dGPU HDA should be enabled or disabled based on the value of OptimusFlag from HG variable (Nvidia Driver -> ASL code -> OpRegion -> S3/S4 callback will sync value to HG variable) <N18> Normal boot: dGPU HDA always should be enabled. S3/S4 resume: dGPU HDA always should be enabled.
Behavior: Load and execute dGPU VBIOS (AMD only) **Description:** System will search and copy VBIOS into HG operation region (Both Nvidia and AMD). Only AMD needs allocate legacy region and execute VBIOS for initialize.
Behavior: Set HG or discrete SSDT. **Description:** Get and install ACPI tables. Create and initialize HG operation region. PcdHgNvidiaDgpuCheckTable to switch SSDTs of NVIDIA DGPU N17 and N18 dynamically. Default values of PcdHgNvidiaDgpuCheckTable are N18 DGPU device ID. When table compare current DGPU device ID is match, system will support N18 chip. If you want to use other values to switch N17 and N18, you can fill these values in the PcdHgNvidiaDgpuCheckTable. Also provide PcdHgNvidiaDgpuCheckTable2 to switch NVIDIA DGPU N17, N18, N20 or further generation dynamically, When PcdDefaultBoardId isn't 0xFF, BIOS will support N18 chip for QA test public board. **Note:** We will based on your PCH type and HG mode to install corresponding table.
Behavior: Update HG information protocol for SMM driver usage. **Description:** Update all of HG data for HG SMM INT15 callback usage.
Behavior: Disable dGPU bridge hot-plug SMI/SCI. **Description:** To avoid hot-plug notify event on root port.
Behavior: Create event to clean secondary dGPU command register. **Description:** Clean the command register of secondary dGPU to avoid I/O resource crash.
Behavior: Assign the value of dGPU SSID/SVID to global NVS. **Description:** Provide a mechanism to restore corresponding dGPU SSID/SVID in _ON method instead of using hard-coded.

5.4SMM Phase

File Path: AlderLakeChipsetPkg\HybridGraphicsSmm\HybridGraphicsSmm.c
Behavior: Locate HG information protocol for callback function usage. **Description:** HG SMM driver will get the protocol location and HG DXE driver will update protocol data during BDS phase for HG SMM INT 15 callback function usage.
Behavior: Register INT 15 callback function for legacy boot.

Description: Support INT 15 function for Intel and MXM (Nvidia only).
Behavior: Register NvidiaOptimusHandler() callback function (Nvidia only). **Description:** Register I/O trap at 0x3CA for error handle.

6 Troubleshooting

This chapter will introduce some basic verify and debug methods.

6.1 Verify HG Feature

Before install driver:

1. Make sure HG related PCD settings are correct.
2. Check SSID/SVID setting are not covered by ODM common code.
3. For Windows 8.1, make sure _STA method of dGPU bridge has been removed because it will cause HG function abnormally.
4. After PEI phase graphics initialization, please check slot status and device status.
5. Boot into Shell and make sure dGPU device is existing.
6. Boot into OS and make sure HG SSDT table has been loaded.
7. Check operation region "VBOR" to make sure VBIOS is correct.

After install driver:

A. Make sure dGPU device doesn't have yellow bang.

B. Wait a moment and open RW, dGPU device should be hidden (OFF state).

C. Check HG ON/OFF function.

PEI phase initialization:

After initialization, make sure both of graphics controllers are visible on bus (You may need H2ODDT to check this). Please refer chapter 6.2 for more detailed information.

Fail to install driver:

Hybrid Graphics heavily rely on graphics driver operation under OS environment. Correct SSID/SVID must be checked to let OS can load correct driver for HG functionality. You also need to check bit 0 (I/O resource) of dGPU command register must be 0.

6.2 Check Slot Status and Device Status

Debug Point: ULT platform (AlderLake P) PCIe bridge **offset 0x58 bit 22 (Currently, AlderLake S are in the scope of this document).** This bit should be set to 1 once power enable sequence is complete in HybridGraphicsPei.c.

Table 12-1. Summary of PCI Express* Port Configuration Registers (Continue

Offset Start	Offset End	Register Name (ID)—Offset	Default Value
4Ch	4Fh	Link Capabilities (LCAP)—Offset 4Ch	710C00h
50h	53h	Link Control And Link Status (LCTL_LSTS)—Offset 50h	10000h
54h	57h	Slot Capabilities (SLCAP)—Offset 54h	40060h
58h	5Bh	Slot Control And Slot Status (SLCTL_SLSTS)—Offset 58h	0h
5Ch	5Fh	Root Control (RCTL)—Offset 5Ch	0h
60h	63h	Root Status (RSTS)—Offset 60h	0h
64h	67h	Device Capabilities 2 (DCAP2)—Offset 64h	80837h
68h	6Bh	Device Control 2 And Device Status 2 (DCTL2_DSTS2)—Offset 68h	0h
6Ch	6Fh	Link Capabilities 2 (LCAP2)—Offset 6Ch	0h
70h	73h	Link Control 2 And Link Status 2 (LCTL2_LSTS2)—Offset 70h	1h
80h	83h	Message Signaled Interrupt Identifiers And Message Signaled Interrupt Message Control (MID_MC)—Offset 80h	9005h
84h	87h	Message Signaled Interrupt Message Address (MA)—Offset 84h	0h
88h	8Bh	Message Signaled Interrupt Message Data (MD)—Offset 88h	0h
90h	93h	Subsystem Vendor Capability (SVCAP)—Offset 90h	A00Dh
94h	97h	Subsystem Vendor IDs (SVID)—Offset 94h	0h
A0h	A3h	Power Management Capability And PCI Power Management Capabilities (PMCAP_PMC)—Offset A0h	C8030001h
A4h	A7h	PCI Power Management Control And Status (PMCS)—Offset A4h	8h

Bit Range	Default and Access	Field Name (ID): Description
31:25	0h RO	Reserved.
24	0h RW/1C/V	**Data Link Layer State Changed (DLLSC):** This bit is set when the value reported in Data Link Layer Link Active field of the Link Status register is changed. In response to a Data Link Layer State Changed event, software must read Data Link Layer Link Active field of the Link Status register to determine if the link is active before initiating configuration cycles to the hot plugged device.
23	0h RO	**Electromechanical Interlock Status (EMIS):** Reserved as this port does not support and electromechanical interlock.
22	0h RO/V	**Presence Detect State (PDS):** If XCAP.SI is set (indicating that this root port spawns a slot), then this bit indicates whether a device is connected ('1') or empty ('0'). If XCAP.SI is cleared, this bit is a '1'.
21	0h RO	**MRL Sensor State (MS):** Reserved as the MRL sensor is not implemented.
20	0h RO	**Command Completed (CC):** This register is RO as this port does not implement a Hot Plug Controller.
19	0h RW/1C/V	**Presence Detect Changed (PDC):** This bit is set by the root port when the PDS bit changes state.
18	0h RO	**MRL Sensor Changed (MSC):** Reserved as the MRL sensor is not implemented.
17	0h RO	**Power Fault Detected (PFD):** Reserved as a power controller is not implemented.
16	0h RO	**Attention Button Pressed (ABP):** This register is RO as this port does not implement an attention button.
15:13	0h RO	Reserved.

Check dGPU device exists on PCIe root port or not:

Intel\AlderLake\ClientOneSiliconPkg\SystemAgent\LibraryPrivate\PeiSaInitLib\SaInitPreMem.c -> CheckOffboardPcieVga()

6.3ASL Code Debug

CMOS Debug:

You need to check which CMOS space is usable in ChipsetCmos.h and OemCmos.h

Example of CMOS debug
OperationRegion (RTCO, SystemIO, 0x72, 0x2)

```
Field (RTCO, ByteAcc, NoLock, Preserve){
   CIND, 8,
   CDAT, 8
}
IndexField (CIND, CDAT, ByteAcc, NoLock, Preserve) {
   Offset (0x00),
   DBG0, 8,
   DBG1, 8,
   DBG2, 8,
   DBG3, 8,
   DBG4, 8,
   DBG5, 8,
   DBG6, 8,
   DBG7, 8,
}
=============================================================
Store (0xAA, DBG0)
Store (0xFF, DBG7)
```

80 Port debug:

There has a method which can use 80 port to debug.

Note: When you add 80 port debug code into method, it may lead timeout and then fail.

Example of 80 port debug

```
External (P8XH, MethodObj)
=============================================================
P8XH (0, 0x30)
```

```
OperationRegion (DP80, SystemIO, 0x80, 0x01)
Field (DP80, ByteAcc, Lock, Preserve)
{
   IO80, 8,
}
```

```
Method (SH80, 1)
{
  Store (arg0, IO80)
}
================================================================
SH80 (0xAA)
SH80 (0xFF)
```

Memory debug:

File Path: Intel\AlderLake\AlderLakeBoardPkg\Acpi\AcpiTables\Dsdt\Memdbg.asl

```
OperationRegion(\MDBG,SystemMemory,0x55AA55AA, 0x55AA55AA)
Field(\MDBG,AnyAcc,Lock,Preserve)
{
  Offset(0),
  MDG0, 32768        // Set total memory for debug code = 4096 * 8
}
...
```

Example of memory debug

```
External (ADBG, MethodObj)
External (DD2H, MethodObj)
================================================================
ADBG ("HG Test")
DD2H (Arg0)
```

How to check the debug message?

1. Use RW find DSDT table and search "MDBG" to know the operation region address.
2. Use RW to read the memory address which get from step 1.

Windows Debug:

Windows debug is not in the scope of this document.

1 Appendix A - CRB code HG Flow for PEI Power Sequence