Common Platform Module

Publication #	Revision: **1.1.2**
Issue Date:	**December 12, 2013**

Revision History

Date	Revision	Description
December 11, 2013	1.1.2	Update AMD_CPM_MAIN_TABLE Update AMD_CPM_GPIO_SETTING Update AMD_CPM_GEVENT_ITEM Update AMD_CPM_GEVENT_SETTING Update AMD_CPM_PREDEFINED_SAVE_CONTEXT Update AMD_CPM_PEIM_PUBLIC_FUNCTION Update AMD_CPM_COMMON_FUNCTION Add AMD_CPM_GPIO_VDDP_VDDR_VOLTAGE_TABLE Update Sample Code
March 14, 2013	1.1.1	Update AMD_CPM_COMMON_FUNCTION
February 22, 2013	1.1.0	Update AMD_CPM_COMMON_FUNCTION Update AMD_CPM_TABLE_PPI Update AMD_CPM_TABLE_PROTOCOL
February 19, 2013	1.0.9	Update AMD_CPM_COMMON_FUNCTION
February 6, 2013	1.0.8	Remove AmdCpmOemInitDxe driver
January 23, 2013	1.0.7	Update AMD_CPM_COMMON_FUNCTION Update AMD_CPM_MAIN_TABLE Update AMD_CPM_DISPLAY_FEATURE_TABLE Add AMD_CPM_WIRELESS_BUTTON_TABLE Update Sample Code
December 5, 2012	1.0.6	Update AMD_CPM_COMMON_FUNCTION Update AMD_CPM_PEIM_PUBLIC_FUNCTION Update AMD_CPM_MAIN_TABLE Add AMD_CPM_SAVE_CONTEXT_TABLE
November 12, 2012	1.0.5	Update AMD_CPM_EXPRESS_CARD_TABLE
October 31, 2012	1.0.4	Update AMD_CPM_TABLE_PROTOCOL
September 19, 2012	1.0.3	Update the following structure AMD_CPM_DISPLAY_FEATURE_CONFIG AMD_CPM_DISPLAY_FEATURE_TABLE AMD_CPM_GPIO_DEVICE_DETECTION
August 15, 2012	1.0.2	Add Adaptive S4 support

August 01, 2012	1.0.1	1. Merge Ext ClkGen module in GPIO Init module 2. Update the following structures: AMD_CPM_MAIN_TABLE AMD_CPM_ACPI_THERMAL_FAN_TABLE AMD_CPM_DEVICE_PATH_ITEM AMD_CPM_PRE_SETTING_ITEM AMD_CPM_TABLE_PPI AMD_CPM_COMMON_FUNCTION AMD_CPM_PEIM_PUBLIC_FUNCTION 3. Update the size of GPIO pin from 8-bit to 16-bit 4. Remove Common Interface Driver 5. Remove some SMI handlers from AmdCpmDisplayFeatureSmm Driver 6. Add AmdCpmTableHobPpi in AmdCpmInitPeim Driver.
May 09, 2012	1.0.0	First Revision

TABLE OF CONTENTS

Chapter 1 Introduction

The AMD Common Platform Module (CPM) software is a BIOS procedure library designed to aid AMD customers to quickly implement AMD platform technology into their products.

This document covers the interface definition for the procedure library and provides some guidelines on how to use the library in the customer's environment.

This chapter explains the goals of the CPM software.

1.1 Goals

The Common Platform Module is designed to support UEFI code base. It is planned to support three major IBVs (Insyde, AMI and Phoenix), as well as OEM's in house UEFI code base with well-defined interfaces or wrappers. The difference UEFI code base shares the same source code. The same feature or function will only need to be implemented one time. If there is same issue, it will only need to be investigated and fixed in one code base for most cases.

The Common Platform Module defines the standard programming interfaces for different AMD platform BIOS. Different platform BIOS can use the same table format to define the board specific feature, such as GPIO pins, GEVENT pins and On-board Device Power On/Off sequence.

1.2 Proposed Modules

The following modules are included in Common Platform Module

- ACPI Thermal Fan Control
  - Use ACPI method to switch fan speed according to the CPU temperature.
- Adaptive S4
- Boot Time Record
  - Provide pre-defined macro and driver to record the time stamp in BIOS post sequence.
- Display Feature
  - This includes a set of display features which are shared by mobile and desktop platform, such as PowerXpress(PX), Hyper CrossFire(HCF), and Surround View (SView).
- EC Init
  - This module is used to initialize external KBC controller to enable S5+ in battery mode.
- GPIO Init

- Initialize GPIO and GEVENT pins according to the board design and setup option. Define on-board device initialization sequences and Initialize PCIE.

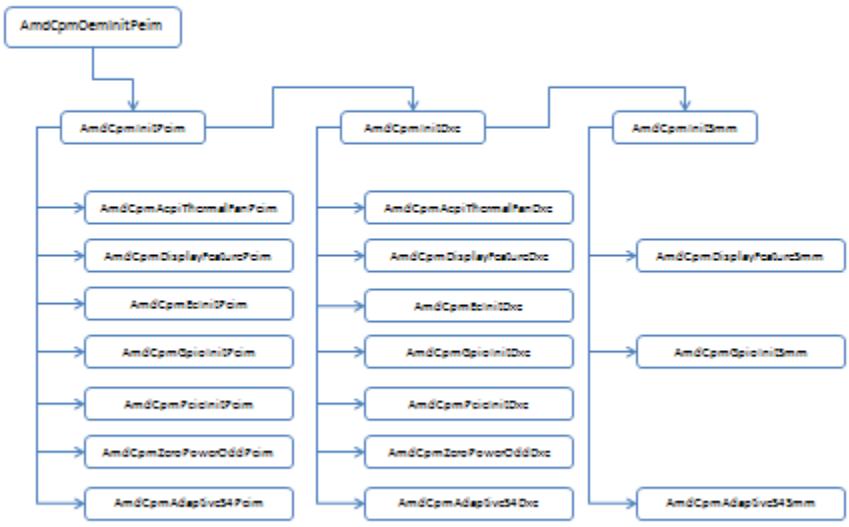
➢ PCIE Init

- Define PCIE Topology Table and PCIE Device Reset Interface. It also provides Express Card support on APU or NB PCIE slot.

➢ Zero Power ODD

- Provide ZeroPowerOdd support and ODD hot-plug support

1.3 Architecture Overview



The Common Platform Module (CPM) can be separate to three different parts.

➢ Platform OEM Driver
➢ CPM Kernel Drivers
➢ Platform Feature Drivers

The Platform OEM Driver includes AmdCpmOemInitPeim drivers. This driver is platform dependent and it will provide platform definition table for CPM in Pei and Dxe stages. If the table needs to be used in both stages, it should be defined in AmdCpmOemInitPeim and the setting will be passed to Dxe stage by using Hob data structure. AmdCpmOemInitPeim is mandatory for CPM. AmdCpmOemInitPeim can also provide call back function to override platform definition table according to the setup option.

The CPM Kernel Drivers include AmdCpmInitPeim, AmdCpmInitDxe and AmdCpmInitSmm driver. AmdCpmInitPeim gets the platform definition tables from AmdCpmOemInitPeim and copies the table, which may be overridden, to the allocated cache area. It may trigger a callback function which is defined in AmdCpmOemInitPeim to update the table according to setup menu or other condition. AmdCpmInitPeim provides a Ppi function for other Pei driver to get the pointer of the definition table and common functions. AmdCpmInitDxe will do the same task as AmdCpmInitPeim in Dxe stage except that it will duplicate the definition table in Pei stage from Hob data structure. AmdCpmInitSmm will provide an interface for Smm driver to get the pointer of definition tables and common functions.

Each platform feature may include one UEFI driver in PEI, DXE or SMM stage. These drivers are used to implement the specific feature and have to be installed after Platform OEM Driver or CPM Kernel Driver. They can get the platform specific information by accessing the definition table.

Platform BIOS needs to define a set of tables in Platform OEM Drivers for one specific platform. These tables include all platform specific information, such as GPIO pins, GEVENT pins, Power On/Off sequence, Thermal Fan Policy, etc. The CPM will use this information to initialize the board. Platform BIOS only needs to provide these tables. The detailed programming will be implemented in CPM. The function code does not need to be implemented multiple times.

One platform BIOS may be used to support several different platforms. Only one set of table will be used in BIOS post time. We may use different tables for different platforms. We can also use same table to support different platforms. In the table header, PlatformMask is used to identify which platforms this table to support.

Some CPM features may also need ACPI table to support. These ACPI tables in CPM will be packaged to a SSDT table in build time. There are three methods to communicate between CPM driver and SSDT table.

- Some ASL code will be patched in post time, such as GEVENT pin
- CPM driver allocated a buffer in ACPI memory in the post time and pass the platform information to SSDT table.
- SSDT table generates software SMI which will be handled in CPM driver.

There are the following methods to communicate between these SSDT tables and main BIOS.

- CPM SSDT table defines some methods or data which will be used by main DSDT methods or other SSDT methods.
- CPM SSDT uses some external methods or data which are requested to be defined in DSDT or other SSDT table.

Common Platform Module also needs SMI function to implement the some feature. AGESA/FCH Driver or CIM-X Driver have provided interface to register new SMI handler.

Chapter 2 Platform Definition Table Structure

Platform Definition Table defines all platform specific parameter for Common Platform Module, such as GPIO, G-Event, Fan Policy, Power Sequence, etc. These tables should be defined in AmdCpmOemInitPeim Driver.

2.1 CPM Definition Table Pointer List

It is required to define a CPM Definition Table Pointer List in AmdCpmOemInitPeim. The list pointer needs to be passed by AMD_CPM_OEM_TABLE_PPI to AmdCpmInitPeim. All definition tables have to be included in this table list.

Example:

```
void *gPcmTableList[] = {
        &gPcmMainTable,
        &gPcmDisplayFeatureTable,
        …
};
```

2.2 CPM Table Common Header Structure

This structure is used to define the header of CPM table.

Prototype

```
typedef struct {
  UINT32          TableSignature;
  UINT16          TableSize;
  UINT8           FormatRevision;
  UINT8           ContentRevision;
  UINT32          PlatformMask;
  UINT32          Attribute;
} AMD_CPM_TABLE_COMMON_HEADER;
```

Parameter

TableSignature:	This is a table signature. The same table type will have same signature. We can find the table by searching this signature in BIOS or memory image. The signature of CPM main table is “$CPM”
TableSize:	This is the table size in byte. This field should be set to 0 for CPM command table.

FormatRevision: This is the revision of table format. It needs to be updated if the definition of table structure is changed.

ContentRevision: This is the revision of table content. It needs to be updated if any data in the table is changed.

PlatformMask: This is platform revision the table supports. Each bit represents a different platform revision. If the bit current platform used is set, it means that this table should be used by CPM. If 0, it means that this table will always be used.

Attribute: This is the table attribute. It is used to define when this table will be used and whether this table will be overridden.

Bit0:	Pei
Bit1:	Dxe
Bit2:	Smm
Bit3:	Override
Bit4~31:	Reserved

2.3 AMD CPM Main Table

This structure is used to define CPM main table.

Prototype

```
typedef struct {
  AMD_CPM_TABLE_COMMON_HEADER          Header;
  UINT8                                PlatformName[32];
  UINT8                                BiosType;
  UINT16                               CurrentPlatformId;
  UINT32                               PcieMemIoBaseAddr;
  UINT32                               AcpiMemIoBaseAddr;
  AMD_CPM_POINTER                      Service;
  AMD_CPM_POINTER                      TableInRomList;
  AMD_CPM_POINTER                      TableInRamList;
  AMD_CPM_POINTER                      TableInHobList;
  AMD_CPM_POINTER                      HobTablePtr;
  AMD_CPM_DISPLAY_FEATURE_CONFIG       DisplayFeature;
  UINT8                                ZeroPowerOddEn;
  UINT8                                AcpiThermalFanEn;
  UINT8                                ExtClkGenEn;
  UINT8                                UnusedGppClkOffEn;
  UINT8                                 AdaptiveS4En;
  UINT8                                 WirelessButtonEn;
  AMD_CPM_EC_CONFIG                    Ec;
  UINT8                                TdpLimitChangeEn;
```

```
  UINT8                                   SmiCheckToolEn;
  UINT8                                   LpcUartEn;
  UINT8                                   ProchotEn;
} AMD_CPM_MAIN_TABLE;
```

Parameter

Header: Table header. The Signature should always be "$CPM".

PlatformName: This is the platform name in 32-byte.

BiosType: This is used to define BIOS type this table supports.
- Bit0: External BIOS
- Bit1: Internal BIOS
- Bit2: HST BIOS
- Bit3: Emulation BIOS
- Bit4-31: Reserved

CurrentPlatformId: This is the platform Id of current platform. It is used to identify whether one CPM definition table is used for current platform by checking PlatformMask[CurrentPlatformId]. If CurrentPlatformId >31, all tables will be included in table list and CurrentPlatformId should be overridden later by AMD_CPM_PLATFORM_ID_TABLE.

PcieMemIoBaseAddr: This is the PCIe Memory IO Base address. CPM will use this base address to access PCI register. If 0, CPM kernel will get the value from MSR register.

AcpiMemIoBaseAddr: This is the Acpi Mmeory IO Base address. CPM will use this base register to access special south bridge register, such as Gpio, IoMux, Pmio, etc. If 0, CPM kernel will get the value from PMIO register.

Service: Reserved for internal use.

TableInRomListPtr: Reserved for internal use.

TableInRamListPtr: Reserved for internal use.

TableInHobListPtr: Reserved for internal use.

HobTablePtr: Reserved for internal use.

DisplayFeature: This is the configuration setting for Display Feature.
- BIT0: PowerXpressFixedMode
- BIT1-2: PowerXpressDynamicMode
- BIT3: HyperCrossFire
- BIT4: SurroundView
- BIT5-7: Reserved

BIT8: IsDgpuPrimary
BIT9: IsBrightnessByDriver
BIT10: DisableGpuAudioInPX
BIT11: DualGraphicsNotSupported
BIT12: DgpuDisplayOutput
BIT13: SpecialPostIgpu
BIT14: PulseGeneratorSupport
BIT15-31: Reserved

ZeroPowerOddEn: This is configuration parameter to control Zero Power Odd module.
BIT0: ZeroPowerOdd Enable
BIT1: ODD Hot-plug Enable if BIT0 = 1
BIT2: Assume System Boot With PS0
BIT3: Enable _PRW method
BIT4: Enable port reset workaround

AcpiThermalFanEn: This is configuration parameter to control Acpi Thermal Fan module.
BIT0: Enable ACPI Thermal Fan Control

ExtClkGenEn: The parameter to initialize external ClkGen.
0x00 ~ 0x7F: ClkGen Init Parameter

UnusedGppClkOffEn: This is configuration parameter to control PCIE clock.
BIT0: Disable Clock on unused GPP port

AdaptiveS4En: This is configuration parameter to control Adaptive S4.
0x00: Disable.
0x01: EC mode enable
0x02: RTC mode enable

WirelessButtonEn: This is configuration parameter to control Wireless Button Config.
0x0: Disable.
0x1: Radio Off
0x2: Power Off

Ec: This is configuration parameter to control EC module.
BIT0: Reserved.
BIT1: Reserved
BIT2: S5+ Enable

TdpLimitChangeEn: Reserved

SmiCheckToolEn: Reserved

LpcUartEn: Reserved

ProchotEn: Reserved

2.4 AMD CPM ACPI Thermal Fan Table

The following table is used to define Fan Policy, FANOUT pin and GEVENT pin.

Prototype

```
typedef struct {
  AMD_CPM_TABLE_COMMON_HEADER      Header;
  AMD_CPM_FAN_HW_CONFIG            FanHwConfig;
  AMD_CPM_FAN_POLICY               FanPolicy;
} AMD_CPM_ACPI_THERMAL_FAN_TABLE;
```

Parameter

Header:	Table header.
FanHwConfig:	FanOut Pin and GEVENT Pin.
FanPolicy:	Thermal Fan Policy.

The following is the structure to define Thermal HW Config.

Prototype

```
typedef struct {
  UINT8                            EventPin;
  UINT8                            FanNum;
} AMD_CPM_FAN_HW_CONFIG;
```

Parameter

EventPin:	GEVENT Pin Number.
FanNum:	FANOUT Pin Number. 0: FANOUT0. 1: FANOUT1

The following is the structure to define Thermal Fan Policy.

Prototype

```
typedef struct {
```

```
    UINT8                   CpuCRT;
    UINT8                   CpuPSV;
    UINT8                   CpuAC0;
    UINT8                   CpuAC1;
    UINT8                   CpuAC2;
    UINT8                   CpuAC3;
    UINT8                   CpuAL0;
    UINT8                   CpuAL1;
    UINT8                   CpuAL2;
    UINT8                   CpuAL3;
    UINT8                   ThermalSensor;
    UINT8                   HysteresisInfo;
    UINT8                   HysteresisInfoPsv;
} AMD_CPM_FAN_POLICY;
```

Parameter

CpuCRT:	Critical Temperature.
CpuPSV:	Passive Temperature.
CpuAC0~3:	TemperatureThreshold 0 ~ 3.
CpuAL0~3:	Fan Speed PWM Level 0 ~ 3.
ThermalSensor:	Thermal Sensor Select.
HysteresisInfo:	Hysteresis Setting for Active Cooling.
HysteresisInfoPsv:	Hysteresis Setting for Passive Cooling.

2.5 AMD CPM Adaptive S4 Table

The following table is used to define the parameter for Adaptive S3 feature.

Prototype

```
typedef struct {
  AMD_CPM_TABLE_COMMON_HEADER     Header;
  UINT8                           BufferType;
  UINT8                           BufferOffset;
  UINT8                           BufferSize;
  UINT8                           EcRamOffset;
} AMD_CPM_ADAPTIVE_S4_TABLE;
```

Parameter

Header:	Table header.
BufferType:	The type of data buffer. 5: BIOS RAM Other: Reserved
BufferOffset:	The start address of data buffer.
BufferSize:	The size of data buffer.
EcRamOffset:	The offset of EC RAM when EC mode is enabled.

2.6 The Tables for CPM Display Feature

There are several types of table to be designed to support Display Feature. These tables have different purpose.

- AMD CPM Display Table: Define the parameters for Display Feature SSDT table and other function.
- AMD CPM Device Path Table: Define the device path for all graphic devices which may support one of display feature.
- AMD CPM Specific SSID Table: Define the vendor id and device id of the bridge or device which SSID needs to be updated. If this table does not exist, the SSID of bridges and devices to support the enabled display feature should be updated.

2.6.1 AMD CPM Display Feature Table

This table is used to define platform specific setting for SSDT table and other function in display feature module.

Prototype

```
typedef struct {
  AMD_CPM_TABLE_COMMON_HEADER      Header;
  UINT8                            FunctionDisableMask;
  UINT8                            MxmDeviceId;
  UINT8                            MxmOverTempEvent;
  UINT8                            MxmOverTempStateId;
  UINT8                            DisplayConnectEvent;
  UINT8                            DockingDeviceId;
  UINT8                            MuxFlag;
  UINT8                            DisplayMuxDeviceId;
  UINT8                            I2CMuxDeviceId;
  UINT8                            AtpxConnector8Number;
  AMD_CPM_DISPLAY_CONNECTOR_8      AtpxConnector8[20];
```

```
  UINT8                                 AtpxConnector9Number;
  AMD_CPM_DISPLAY_CONNECTOR_9           AtpxConnector9[20];
  UINT32                                AtifSupportedNotificationMask;
  UINT8                                 AtifDeviceCombinationNumber;
  UINT8                                 AtifDeviceCombinationBuffer[20];
  UINT8                                 Atif16Buffer[0x100];
} AMD_CPM_DISPLAY_FEATURE_TABLE;
```

Parameter

Header:	Table header.
FunctionDisableMask:	BIT0: Do not update the SSID of iGPU or dGPU. BIT1: Do not add the SSDT table for display feature. BIT2: Do not enable SW SMI for Display Feature.
MxmDeviceId:	The ID of MXM Module. It has to match with the ID of MXM model which is used in GPIO Device Tables.
MxmOverTempEvent:	GEVENT pin number for MXM_OVERT#.
MxmOverTempStateId:	Forced Power State Id if MXM_OVERT# is low.
DisplayConnectEvent:	GEVENT pin number for Discrete GPU display connect / disconnect event
DockingDeviceId:	The Device Id to control the detection of Docking if BIT7 = 0. If BIT7 = 1, it will be the forced status to be reported to display driver.
MuxFlag:	The flag for Mux-Based Power Xpress. If 0,
DisplayMuxDeviceId:	The device Id to control display mux pin.
I2CMuxDeviceId:	The device Id to control the switch of I2c line.
AtpxConnector8Number:	Number of reported display connectors in ATPX sub-function 8.
AtpxConnector8:	The Connector information for ATPX sub-function 8. The connector information will be generated automatically according to the PCIE topology table if AtpxConnector8Number = 0xFF.
AtpxConnector9Number:	Number of reported display connectors in ATPX sub-function 9.
AtpxConnector9:	The Connector information for ATPX sub-function 9. The connector information will be generated automatically according to the PCIE topology table if AtpxConnector9Number = 0xFF.
AtifSupportedNotificationMask:	Supported Notifications Mask in ATIF sub-function 0.

Bit 0: Display switch request is supported.
Bit 1: Expansion mode change request is supported.
Bit 2: Thermal state change request is supported.
Bit 3: Forced power state change request is supported.
Bit 4: System power source change request is supported.
Bit 5: Display configuration change request is supported.
Bit 6: PowerXpress graphics switch toggle request is supported.
Bit 7: Panel brightness change request is supported.
Bit 8: Discrete GPU display connect/disconnect event is supported
Bits 31-9: Reserved (must be zero).

AtifDeviceCombinationNumber: The number of Display Device Combination.

AtifDeviceCombinationBuffer: The data of Display Device Combination.

Atif16Buffer: The data for Query Brightness Transfer Characteristics

The following table is used to define connector information for ATPX function 8.

Prototype

```
typedef struct {
  UINT8   Flags;
  UINT8   AtifId;
  UINT8   AdaptorId;
  UINT16  AcpiId;
} AMD_CPM_DISPLAY_CONNECTOR_8;
```

Parameter

Flags:
Bit 0: display output supported by the graphics device identified by Adapter ID.
Bit 1: display detectable through HPD by the graphics device identified by Adapter ID.
Bit 2: display I2C/Aux lines available to the graphics device identified by Adapter ID.
Bits 7-3: Reserved (must be zero).

AtifId:
ATIF display vector is defined as:
Bit 0: LCD1
Bit 1: CRT1
Bit 2: TV
Bit 3: DFP1
Bit 4: CRT2
Bit 5: LCD2
Bit 6: Reserved (must be zero)
Bit 7: DFP2
Bit 8: CV

Bit 9: DFP3
Bit 10: DFP4
Bit 11: DFP5
Bit 12: DFP6
Bits 15-13: Reserved (must be zero).

AdaptorId: Adapter ID: 0 = integrated graphics device, 1 = discrete graphics device on the lowest numbered PCIe bus, increments per PCIe bus number.

AcpiId: Connector ACPI ID (local, per adapter), can be different for the same connector where output is multiplexed between two adapters.

The following table is used to define connector information for ATPX function 9.

Prototype

```
typedef struct {
  UINT8                                 AtifId;
  UINT8                                 HpdPortId;
  UINT8                                 DdcPortId;
} AMD_CPM_DISPLAY_CONNECTOR_9;
```

Parameter

AtifId: Same as AtifId in AMD_CPM_DISPLAY_CONNECTOR_8..

HpdPortId: HPD Port ID:
0 = not available
1 = HPD1
2 = HPD2
3 = HPD3
4 = HPD4
5 = HPD5
6 = HPD6

DdcPortId: DDC Port ID:
0 = not available
1 = DDC1
2 = DDC2
3 = DDC3
4 = DDC4
5 = DDC5
6 = DDC6
7 = DDC7
8 = DDC8.

2.6.2 AMD CPM Display Device Path Table

The Device Path Table is used to define display devices which may be enabled for display feature.

Prototype

```
typedef struct {
  AMD_CPM_TABLE_COMMON_HEADER    Header;
  AMD_CPM_DEVICE_PATH_ITEM       Path[AMD_DISPLAY_DEVICE_PATH_SIZE];

} AMD_CPM_DEVICE_PATH_TABLE;
```

Parameter

Header:	Table header.
Path:	The array of display device path. If Feature of the path = 0, it means the end of the array.

The following structure is used to define one display device path.

Prototype

```
typedef struct {
  AMD_CPM_DISPLAY_FEATURE_SUPPORT    Feature;
  UINT8                              IsDgpu;
  AMD_CPM_PCI_DEVICE_FUNCTION        Bridge;
  AMD_CPM_PCI_DEVICE_FUNCTION        Device;
  UINT8                              DeviceId;
  UINT8                              Mode;
} AMD_CPM_DEVICE_PATH_ITEM;
```

Parameter

Feature:	Display features this display device to support.
IsDgpu:	0: Integrated GPU. 1: Discrete GPU.
Bridge:	Device and function number of the bridge. If 0, there is no bridge.
Device:	Device and function number of the device.
DeviceId:	Device Id to control GPIO pin.
Mode:	Power Mode. 0: Power Off. 1: Power On.

The following structure is used to define which feature this display device supports.

Prototype

```
typedef union {
  UINT32                                Raw;
  struct {
        UINT32                          PowerXpress:1;
        UINT32                          HyperCrossFire:1;
        UINT32                          SurroundView:1;
        UINT32                          Reserved1:13;
        UINT32                          Bus:8;
        UINT32                          Reserved2:4;
        UINT32                          Removable:1;
        UINT32                          Vga:1;
        UINT32                          Exist:1;
        UINT32                          Valid:1;
  }                                     Mask;
} AMD_CPM_DISPLAY_FEATURE_SUPPORT;
```

Parameter

PowerXpress:	0: Not Support PowerXpress. 1: Support PowerXpress.
HyperCrossFire:	0: Not Support HyperCrossFire. 1: Support HyperCrossFire.
SurroundView:	0: Not Support SurroundView. 1: Support SurroundView.
Bus:	The bus number of device. It is internal use only.
Removable:	The flag of the device to be power on or off dynamically. It is internal use only.
Vga:	The flag of primary display device. It is internal use only.
Exist:	The flag of display device attached. It is internal use only.
Valid:	0: Invalid Item. 1: Valid Item.

The following structure is used to define PCI device and function of the bridge or device.

Prototype

```
typedef struct {
  UINT8                                 Device:5;
```

```
  UINT8                                   Function:3;
} AMD_CPM_PCI_DEVICE_FUNCTION;
```

Parameter

Device:	PCI device number.
Function:	PCI function number.

2.6.3 AMD CPM Specific SSID Table

Specific SSID Table is used to set SSID of display device which needs to be set to different value according to the feature to be enabled. Display Driver will enable different features according to this SSID. For one platform, one special display feature may only be enabled for some special display devices. By default, all SSID of display device in device path table will be updated. If this specific SSID table exists, the CPM will compare the device id & vendor id of the display device with this table. If it is matched, the SSID of the device will be overridden. Otherwise, it will be kept as the original value

Prototype

```
typedef struct {
  AMD_CPM_TABLE_COMMON_HEADER     Header;
  AMD_CPM_SPECIFIC_SSID_ITEM      Item[AMD_SPECIFIC_SSID_DEVICE_SIZE];
} AMD_CPM_SPECIFIC_SSID_TABLE;
```

Parameter

Header:	Table header.
Path:	The array of Specific SSID Item. If VendorId and DeviceId of the item = 0xFFFF, it means the end of the array.

The following structure is used to define vendor id and device id of PCI device which needs to update SSID.

Prototype

```
typedef struct {
  UINT16                                  VendorId;
  UINT16                                  DeviceId;
} AMD_CPM_SPECIFIC_SSID_ITEM;
```

Parameter

VendorId:	Vendor Id of PCI Device
DeviceId:	Device Id of PCI Device.

2.7 AMD CPM EXT ClkGen Table

This table is used to define the initialize sequence of external ClkGen and the method to set PCIE clock.

Prototype

```
typedef struct {
  AMD_CPM_TABLE_COMMON_HEADER  Header;
  UINT8                        SmbusSelect;
  UINT8                        SmbusAddress;
  AMD_CPM_EXT_CLKGEN_ITEM      Item[AMD_PCIE_CLKGEN_SIZE];
} AMD_CPM_EXT_CLKGEN_TABLE;
```

Parameter

Header:	Table header.
SmbusSelect:	Smbus Select of External ClkGen 0: Smbus 0 1: Smbus 1
SmbusAddress:	Smbus Address of External ClkGen
Item:	The list of External Clock Item

The following structure is used to define External Clock Item.

Prototype

```
typedef struct  {
  UINT8                        Function;
  UINT8                        Offset;
  UINT8                        AndMask;
```

```
  UINT8                                 OrMask;
} AMD_CPM_EXT_CLKGEN_ITEM;
```

Parameter

Function:	The ID of the external ClkGen Item. 0x00~0x7F is for initial sequence. 0x80 ~ 0x8F is used to disable PCIE clock. 0x90~0x9F is used to enable ClkReq. 0xA0~0xFE is reserved. 0xFF is the end item of list.
Offset:	The offset of external ClkGen register.
AndMask:	The register bits which will be kept.
OrMask:	The register bits which will be set.

2.8 The Tables for CPM GPIO Init

There are several tables to be used to define the setting of PCIE device.

- AMD CPM Pre Init Table: Define the register setting before GPIO module to be initialized.
- AMD CPM GPIO Init Table: Define the initial setting for GPIO pins.
- AMD CPM GEVENT Init Table: Define the initial setting for GEVENT pins.
- AMD CPM GPIO Device Config Table: Define the method to initialize the device.
- AMD CPM GPIO Device Detection Table: Define the GPIO setting to detect the device.
- AMD CPM GPIO Device Reset Table: Define GPIO reset sequence of the device.
- AMD CPM GPIO Device Power Table: Define GPIO power on or off sequence of the device
- AMD CPM GPIO Mem Voltage Table: Define GPIO setting to set memory voltage.
- AMD CPM PCIE Clock Table: Define the setting of PCIE clock and ClkReq.
- AMD CPM Ext ClkGen Table: Define the initialize sequence of external clock generator and how to program PCIE clock and ClkReq.

2.8.1 AMD CPM Pre Init Table

This table is used to define special register settings which should be set before GPIO initialize.

Prototype

```
typedef struct {
  AMD_CPM_TABLE_COMMON_HEADER       Header;
  AMD_CPM_PRE_SETTING_ITEM          Item[AMD_PRE_INIT_SIZE];
 } AMD_CPM_PRE_INIT_TABLE;
```

Parameter

Header: Table Header.

Item: The list of register setting which needs to be set before GPIO initializarion.

This Structure is used to define the setting of one special register.

Prototype

```
typedef struct {
  UINT8                                 Type;
  UINT8                                 Select;
  UINT8                                 Offset;
  UINT8                                 AndMask;
  UINT8                                 OrMask;
  UINT8                                 Stage;
 } AMD_CPM_PRE_SETTING_ITEM;
```

Parameter

Type Register type. 0: FCH MMIO. 1: PCI

Select: The register sub-type.

Offset: The offset of register.

AndMask: The AND mask of the register value to set.

OrMask: The OR mask of the register value to set.

Stage: The stage number to load this register.

2.8.2 AMD CPM GPIO Init Table

This table is used to define GPIO initial setting.

Prototype

```
typedef struct {
  AMD_CPM_TABLE_COMMON_HEADER       Header;
  AMD_CPM_GPIO_ITEM                 GpioList[AMD_GPIO_ITEM_SIZE];
 } AMD_CPM_GPIO_INIT_TABLE;
```

Parameter

Header: Table Header.

GpioList: The setting list of GPIO pins

The following structure is used to define the setting of one GPIO pin.

Prototype

```
typedef struct {
  UINT16                    Pin;
  AMD_CPM_GPIO_SETTING      Setting;
 } AMD_CPM_GPIO_ITEM;
```

Parameter

Pin: GPIO Pin Number.

Setting: The setting for this GPIO pin.

The following structure is the definition of GPIO setting.

Prototype

```
typedef union {
  UINT16                    Raw
  struct {
        UINT8               Out:1;
        UINT8               OutEnB:1;
        UINT8               PullUpSel:1;
        UINT8               SetEnB:1;
        UINT8               Sticky:1;
        UINT8               PullUp:1;
        UINT8               PullDown:1;
        UINT8               PresetEn:1;
        UINT8               IoMux:3;
        UINT8               IoMuxEn:1;
        UINT8               DrvStrengthSel:2;
        UINT8               Reserved:2;
  }                         Gpio;
} AMD_CPM_GPIO_SETTING;
```

Parameter

Raw: It is used to access this structure by 16-bit mode.

Out: 0: Set GPIO to low 1: Set GPIO to high if OutputEnB = 0.

OutEnB:	0: GPIO Output. 1: GPIO input
PullUpSel:	0: 4K. 1: 8K
SetEnB:	if 1, both of Out and OutEnB will be ignored and these fields will not be updated.
Sticky:	If 1, GPIO setting will be kept after reset
PullUp:	0: Pull up Disable. 1: Pull up Enable
PullDown:	0: Pull down Disable. 1: Pull down Enable
PresetEn:	If 1, the value of Sticky, Sticky, PullUp and Pull Down will be set to GPIO register.
IoMux:	Multi-function IO pin function select for this GPIO pin
IoMuxEn:	If 1, the value of IOMux will be set to IOMux register.
DrvStrengthSel:	0: 4mA. 1: 8mA. 2: 12mA. 3: 16mA

2.8.3 AMD CPM GEVENT Init Table

This table is used to define initial setting of GVENT pins.

Prototype

```
typedef struct {
  AMD_CPM_TABLE_COMMON_HEADER      Header;
  AMD_CPM_GEVENT_ITEM              GeventList[AMD_GEVENT_ITEM_SIZE];
} AMD_CPM_GEVENT_INIT_TABLE;
```

Parameter

Header:	Table Header.
GeventList:	The setting list of GEVENT pins

The following structure is used to define the setting of one GEVENT pin.

Prototype

```
typedef struct {
  UINT16                                Pin;
```

```
    AMD_CPM_GEVENT_SETTING          Setting;
} AMD_CPM_GEVENT_ITEM;
```

Parameter

Pin:	GPIO Pin Number.	
	0x00 ~ 0x3F:	GEVENT
	0x100 ~ 0x1FF:	GPIO Interrupt
Setting:	The setting for this GEVENT or GPIO interrupt pin.	

The following structure is the definition of GPIO setting.

Prototype

```
typedef union {
  UINT16                        Raw;
  struct {
    UINT16                      EventEnable:1;
    UINT16                      SciTrig:1;
    UINT16                      SciLevl:1;
    UINT16                      SmiSciEn:1;
    UINT16                      SciS0En:1;
    UINT16                      SciMap:6;
    UINT16                      SmiTrig:1;
    UINT16                      SmiControl:4:
  }                             Gevent;
  struct {
    UINT16                      DebounceTmrOut:4;
    UINT16                      DebounceTmrOutUnit:1;
    UINT16                      DebounceCntrl:2;
    UINT16                      Reserved:1;
    UINT16                      LevelTrig:1;
    UINT16                      ActiveLevel:2;
    UINT16                      InterruptEnable:2;
    UINT16                      WakeCntrl:3:
  }                             Gpio;
} AMD_CPM_GEVENT_SETTING;
```

Parameter

Raw:	It is used to access this structure by 16-bit mode.
EventEnable:	0: Disable, 1: Enable.
SciTrig:	0: Falling Edge. 1: Rising Edge

SciLevl:	0: Edge Trigger. 1: Level Trigger.
SmiSciEn:	0: Not send SMI. 1: Send SMI
SciS0En:	0: Disable. 1: Enable
SciMap:	0000b ~ 1111b. SCI interrupt mapping for this GEVENT pin
SmiTrig:	0: Active Low. 1: Active High.
SmiControl:	0: Disable. 1: SMI. 2: NMI. 3: IRQ13
DebounceTmrOut:	Specifies the debounce timer out number
DebounceTmrOutUnit:	0: 30.5us (One RtcClk period), 1: 122us (four RtcClk periods)
DebounceContrl:	00b: No debounce, 01b: Preserve low glitch 10b: Preserve high glitch, 11b: Remove glitch
LevelTrig:	0: Edge trigger, 1: Level trigger
ActiveLevel:	00b: Active High. 01b: Active Low. 10b: Active on both edges if LevelTrig=0
InterruptEnable:	BIT0: Enable interrupt status, BIT1: Enable interrupt delivery
WakeCntrl:	BIT0: Enable wake in S0I3 state, BIT1: Enable wake in S3 state, BIT2: Enable wake in S4/S5 state

2.8.4 AMD CPM GPIO Device Config Table

This table is used to define the initialize sequence of the on-board devices.

Prototype

```
typedef struct {
  AMD_CPM_TABLE_COMMON_HEADER  Header;
  AMD_CPM_GPIO_DEVICE_CONFIG   DeviceList[AMD_GPIO_DEVICE_SIZE];
} AMD_CPM_GPIO_DEVICE_CONFIG_TABLE;
```

Parameter

Header:	Table Header.
DeviceList:	The config list of all onboard devices

The following structure is used to define the setting of each device.

Prototype

```
typedef struct {
  UINT8                       DeviceId;
  union {
    UINT8                 Raw;
    struct {
      UINT8         Enable:2;
      UINT8         ResetAssert:1;
      UINT8         ResetDeassert:1;
      UINT8         Reserved:4;
    }                     Setting;
  }                           Config;
} AMD_CPM_GPIO_DEVICE_CONFIG;
```

Parameter

DeviceId:	The Device Id of the device which needs GPIO pin to control, such as Reset, Detection, Power On or Off. The DeviceId should always be the same for one device in different tables..
Enable:	The default setting in post. 0: Power Off 1: Power On 2: Auto Detection
ResetAssert:	Reset pin needs to be asserted before the device is powered on if ResetAssert = 1.
ResetDeassert:	Reset pin needs to be de-asserted after the device is powered on if ResetDeassert = 1.

2.8.5 AMD CPM GPIO Device Detection Table

This table is used to define how to detect the device.

Prototype

```
typedef struct {
  AMD_CPM_TABLE_COMMON_HEADER   Header;
  AMD_CPM_GPIO_DEVICE_DETECTION
                          DeviceDetectionList[AMD_GPIO_DEVICE_DETECT_SIZE];
} AMD_CPM_GPIO_DEVICE_DETECTION_TABLE;
```

Parameter

Header:	Table Header.
DeviceDetectionList:	The config list of all onboard devices

The following structure is used to define the detection sequence of one device.

Prototype

```
typedef struct {
  UINT8                 DeviceId;
  UINT8                 Type;
  UINT16                PinNum1;
  UINT8                 Value1;
  UINT16                PinNum2;
  UINT8                 Value2;
  UINT16                PinNum3;
  UINT8                 Value3;
} AMD_CPM_GPIO_DEVICE_DETECTION;
```

Parameter

DeviceId:	The Device Id of the device
Type:	The default setting in post. 0: One GPIO pin 1: Two GPIO pin AND 2: Two GPIO pin OR 3: Three GPIO pin AND 4: Three GPIO pin OR
PinNum1:	GPIO Pin One number.
Value1:	GPIO Pin One value when the device is attached.
PinNum2:	GPIO Pin Two number.
Value2:	GPIO Pin Two value when the device is attached.
PinNum3:	GPIO Pin Three number.
Value3:	GPIO Pin Three value when the device is attached.

2.8.6 AMD CPM GPIO Device Reset Table

This table is used to define how to reset the device.

Prototype

```
typedef struct {
  AMD_CPM_TABLE_COMMON_HEADER   Header;
  AMD_CPM_GPIO_DEVICE_RESET
                          DeviceResetList[AMD_GPIO_DEVICE_RESET_SIZE];
} AMD_CPM_GPIO_DEVICE_RESET_TABLE;
```

Parameter

Header: Table Header.

DeviceResetList: The item list of the reset

The following table is used to define GPIO pin.

Prototype

```
typedef struct {
  UINT16                                Pin;
  UINT8                                 Value;
} AMD_CPM_GPIO_PIN;
```

Parameter

Pin: The pin number of GPIO.
0x0000 ~ 0x00FF is for FCH GPIO.
0x0100 ~ 0x01FF is for EC GPIO on CRB.

Value: The value of GPIO pin. It should be 0 or 1 only.

The following structure is used to define one step of one device reset

Prototype

```
typedef struct {
  UINT8                 DeviceId;
  UINT8                 Mode;
  UINT8                 Type;
  union {
    UINT32                  Stall;
```

```
    AMD_CPM_GPIO_PIN      Gpio;
  }                       Config;
  UINT8                   InitFlag;
} AMD_CPM_GPIO_DEVICE_RESET;
```

Parameter

DeviceId: The Device Id of the device

Mode:
- 0: Reset Assert
- 1: Reset De-assert
- 2: Delay between reset assert and de-assert

Type: Type of the register if Mode = 0 or 1.
- 0: GPIO
- 1: Special Pin

Stall: Delay in 1μs/unit between reset assert and de-assert if Mode = 2.

Gpio: The GPIO setting if Type = 0

InitFlag: The flag to init this item in the post time. It will be overridden if the DeviceId is also used in AMD CPM GPIO Device Config Table.
- 0: Disable
- 1: Set in stage one
- 2: Set in stage two
- 3: Not set in BIOS post. The value will be passed to ASL code

2.8.7 AMD CPM GPIO Device Power Table

This table is used to define how to power on or off the device.

Prototype

```
typedef struct {
  AMD_CPM_TABLE_COMMON_HEADER   Header;
  AMD_CPM_GPIO_DEVICE_POWER
                    DevicePowerList[AMD_GPIO_DEVICE_POWER_SIZE];
} AMD_CPM_GPIO_DEVICE_POWER_TABLE;
```

Parameter

Header: Table Header.

DevicePowerList: The item list to power on or off the device.

The following structure is used to define one step to power on or off the device.

Prototype

```
typedef struct {
  UINT8                   DeviceId;
  UINT8                   Mode;
  UINT8                   Type;
  union {
    UINT32                    Stall;
    AMD_CPM_GPIO_PIN          SetGpio;
    AMD_CPM_GPIO_PIN          WaitGpio;
  }                       Config;
  UINT8                   InitFlag;
} AMD_CPM_GPIO_DEVICE_POWER;
```

Parameter

DeviceId: The Device Id of the device

Mode:
- 0: Power Off
- 1: Power On

Type:
- 0: Set GPIO
- 1: Wait GPIO
- 2: Stall

Stall: Delay in 1μs/unit if Type = 2.

SetGpio: Set GPIO pin if Type = 0

WaitGpio: Wait for the value of GPIO pin if Type = 1

InitFlag: The flag to init this item in the post time. It will be overridden if the DeviceId is also used in AMD CPM GPIO Device Config Table.
- 0: Disable
- 1: Set in stage one
- 2: Set in stage two
- 3: Not set in BIOS post. The value will be passed to ASL code

2.8.8 AMD CPM GPIO Mem Voltage Table

This table is used to set the memory voltage according to the memory speed.

Prototype

```
typedef struct {
  AMD_CPM_TABLE_COMMON_HEADER   Header;
  AMD_CPM_GPIO_MEM_VOLTAGE_ITEM
                                Item[AMD_GPIO_MEM_VOLTAGE_SIZE];
} AMD_CPM_GPIO_MEM_VOLTAGE_TABLE;
```

Parameter

Header:	Table Header.
Item:	The item list to set memory voltage.

The following structure is used to define the GPIO pins to set the memory voltage.

Prototype

```
typedef struct {
  UINT8                 Voltage;
  UINT16                GpioPin1;
  UINT8                 Value1;
  UINT16                GpioPin2;
  UINT8                 Value2;
} AMD_CPM_GPIO_MEM_VOLTAGE_ITEM;
```

Parameter

Voltage:	The index of memory voltage which has to match with the setting of AGESA. It is the end of list if Voltage = 0xFF. 0: Initial value for VDDIO 1: 1.5V 2: 1.35V 3: 1.25V
GpioPin1:	The pin number of GPIO one
Value1:	The value to set for GPIO one
GpioPin2:	The pin number of GPIO two.
Value2:	The value to set for GPIO two

2.8.9 AMD CPM GPIO VDDP/VDDR Voltage Table

This table is used to set the VDDP/VDDR voltage according to the memory setting.

Prototype

```
typedef struct {
  AMD_CPM_TABLE_COMMON_HEADER   Header;
  AMD_CPM_GPIO_VDDP_VDDR_VOLTAGE_ITEM
                                Item[AMD_GPIO_VDDP_VDDR_VOLTAGE_SIZE];
} AMD_CPM_GPIO_VDDP_VDDR_VOLTAGE_TABLE;
```

Parameter

Header:	Table Header.
Item:	The item list to set VDDP/VDDR voltage.

The following structure is used to define the GPIO pins to set the memory voltage.

Prototype

```
typedef struct {
  UINT8                         Voltage;
  UINT16                        GpioPin1;
  UINT8                         Value1;
} AMD_CPM_GPIO_VDDP_VDDR_VOLTAGE_ITEM;
```

Parameter

Voltage:	The index of VDDP/VDDR voltage which has to match with the setting of AGESA. It is the end of list if Voltage = 0xFF. 0: 0.95 Volt 1: 1.05 Volt
GpioPin1:	The pin number of GPIO one
Value1:	The value to set for GPIO one

2.8.10 AMD CPM PCIE Clock Table

This table is used to define the setting of PCIE Clock.

Prototype

```
typedef struct {
  AMD_CPM_TABLE_COMMON_HEADER    Header;
  AMD_CPM_PCIE_CLOCK_ITEM        Item[AMD_PCIE_CLOCK_SIZE];
} AMD_CPM_PCIE_CLOCK_TABLE;
```

Parameter

Header:	Table header.
Item	The definition of PCIE Clock

The following structure is used to define each PCIE Clock.

Prototype

```
typedef struct  {
  UINT8          ClkId;
  UINT8          ClkReq;
  UINT8          ClkIdExt;
  UINT8          ClkReqExt;
  UINT8          DeviceId;
  UINT8          Device;
  UINT8          Function;
  UINT8          SlotCheck;
  UINT32         SpecialFunctionId;
} AMD_CPM_PCIE_CLOCK_ITEM;
```

Parameter

ClkId:	The ID of PCIE Clock for Internal ClkGen.
ClkReq:	PCIE Clock Setting for Internal ClkGen. 0x00: Clock Disable 0xFF: Clock Always On 0x01~0x0A: CLK_REQ0 ~ CLK_REQ10
ClkId:	The ID of PCIE Clock for External ClkGen.
ClkReq:	PCIE Clock Setting for External ClkGen. 0x00: Clock Disable 0xFF: Clock Always On 0x01~0x0A: CLK_REQ0 ~ CLK_REQ10
DeviceId:	The Device Id in GPIO Device Detection Table. If it is not 0xFF, GPIO Device Detection Table will be used to check

	the status of the device. If the device does exist, the clock will be disabled.
Device:	The device number of PCIE Bridge this clock is connected.
Function:	The function number of PCIE Bridge this clock connected.
SlotCheck:	The device behind the bridge will be check. The clock will be disabled if there is no device is found. BIT0: Check whether the PCI space exists BIT1: Check whether the device exists according to GPIO pins. BIT2: Check whether clock power management is enabled in PCI space
SpecialFunctionId:	This Id is used to do some special sequence for this device while setting the clock.

2.9 The Tables for CPM PCIE Init

There are several tables to be used to define the setting of PCIE device.

- AMD CPM PCIE Topology Table: Define the PCIE topology structure of the platform.
- AMD CPM PCIE Topology Override Table: Define the items in AMD CPM PCIE Topology Table to be overridden and how to override.
- AMD CPM Express Card Table: Define the parameter of Express Card on APU PCIE port.

2.9.1 AMD CPM PCIE Topology Table

The following table is used to define PCIE Topology Table which will be passed to AGESA. The definition of PCIe_PORT_DESCRIPTOR and PCIe_DDI_DESCRIPTOR is same as that in AGESA.

Prototype

```
typedef struct {
  AMD_CPM_TABLE_COMMON_HEADER      Header;
  UINT32                           SocketId;
  PCIe_PORT_DESCRIPTOR             Port[AMD_PCIE_PORT_DESCRIPTOR_SIZE];
  PCIe_DDI_DESCRIPTOR              Ddi[AMD_PCIE_DDI_DESCRIPTOR_SIZE];
} AMD_CPM_PCIE_TOPOLOGY_TABLE;
```

Parameter

Header:	Table header.

SocketId:	The Socket Id of this PCIe Topology table.
Port:	PCIe Port Descriptor List
Ddi:	PCIe DDI Descriptor List.

2.9.2 AMD CPM PCIE Topology Override Table

PCIE Topology Table will be changed according to different board configuration or setup option. CPM will update the topology table dynamically according to this override table. This override table can be hardcoded in the build time. It can be also updated in the early post time.

Prototype

```
typedef struct {
  AMD_CPM_TABLE_COMMON_HEADER        Header;
  AMD_CPM_PCIE_TOPOLOGY_OVERRIDE_ITEM
                                     Item[AMD_PCIE_TOPOLOGY_OVERRIDE_SIZE];
} AMD_CPM_PCIE_TOPOLOGY_OVERRIDE_TABLE;
```

Parameter

Header:	Table header.
Item:	The array of override item. If the flag of item = 0xFF, it means the end of the array. CPM will update one of PCIe Port Descriptor or PCIe DDI Descriptor according to each override item.

The following structure is used to define one of PCIE Topology Override Item.

Prototype

```
typedef struct {
  union {
    UINT8                          Raw;
    struct {
      UINT8                              EnableOverride:1;
      UINT8                              DdiTypeOverride:1;
      UINT8                              LaneOverride:1;
      UINT8                              PortPresentOverride:1;
      UINT8                              IsDdi:1;
      UINT8                              Reserved:2;
      UINT8                              Valid:1;
```

```
    }                                   Config;
  }                                   Flag;
  UINT8                               Offset;
  UINT8                               Enable;
  UINT8                               DdiType;
  UINT8                               PortPresent;
  UINT8                               StartLane;
  UINT8                               EndLane;
} AMD_CPM_PCIE_TOPOLOGY_OVERRIDE_ITEM;
```

Parameter

Flag: This is the flag of this override item. It will be last and invalid item if Flag = 0xFF.
BIT0: 0: Enable Override Disable. 1: Enable Override Enable
BIT1: 0: DDI Type Override Disable. 1: DDI Type Override Enable
BIT2: 0: Lane Override Disable. 1: Lane Override Enable
BIT3: 0: Port Present Override Disable. 1: Port Present Override Enable
BIT4: 0: Override Port Descriptor. 1: Override DDI Descriptor
BIT5-6: Reserved
BIT7: 0: Invalid Override Item. 1: Valid Override Item.

Offset: The offset of descriptor to override in Port Descriptor List or DDI Descriptor List according to the value of IsDdi in Flag.

Enable: The EngineType of the descriptor to be set, if EnableOverride = 1.

DdiType: The ConnectorType of the DDI descriptor to be set, if DdiTypeOverride = 1 and IsDdi = 1.

PortPresent: The PortPresent of the Port descriptor to be set, if PortPresentOverride = 1 and IsDdi = 0.

StartLane: The StartLane of the Port descriptor to be set, if PortPresentOverride = 1 and IsDdi = 0.

EndLane: The EndLane of the Port descriptor to be set, if PortPresentOverride = 1 and IsDdi = 0.

2.9.3 AMD CPM Express Card Table

This table is used to define the setting of Express Card.

Prototype

```
typedef struct _AMD_CPM_PCIE_CLOCK_TABLE {
  AMD_CPM_TABLE_COMMON_HEADER    Header;
  UINT8                          Device;
  UINT8                          Function;
  UINT8                          EventPin;
  UINT8                          DeviceId;
} AMD_CPM_EXPRESS_CARD_TABLE;
```

Parameter

Header:	Table header.
Device:	The device number of PCIE Bridge.
Function:	The function number of PCIE Bridge.
EventPin:	GEVENT Pin Number.
DeviceId:	Device Id of Express Card

2.9.4 AMD CPM Wireless Button Table

This table is used to define the setting of Wireless Button.

Prototype

```
typedef struct _AMD_CPM_WIRELESS_BUTTON_TABLE {
   AMD_CPM_TABLE_COMMON_HEADER    Header;
   AMD_CPM_PCI_DEVICE_FUNCTION    Bridge[4];
   UINT8                          EventPin;
   UINT8                          DeviceIdRadio;
   UINT8                          DeviceIdPower;
   UINT8                          DeviceIdOther;
} AMD_CPM_WIRELESS_BUTTON_TABLE;
```

Parameter

Header:	Table header.
Bridge:	The device and function number of PCIE Bridge.
EventPin:	GEVENT Pin Number.
DeviceIdRadio:	Device Id to control the radio of wireless device

DeviceIdPower: Device Id to control the power of wireless device

DeviceIdOther: Device Id to control other device, such as BlueTooth.

2.10 AMD CPM Zero Power Odd Table

This table is used to define GPIO pin to control ODD power, GEVENT pins to trigger interrupt , SATA port and SATA mode to support.

Prototype

```
typedef struct {
  AMD_CPM_TABLE_COMMON_HEADER     Header;
  UINT8                           DeviceId;
  UINT8                           EventPin1;
  UINT8                           EventPin2;
  UINT8                           EventPin3;
  UINT8                           SataModeSupportMask;
  UINT8                           SataPortId;
} AMD_CPM_ZERO_POWER_ODD_TABLE;
```

Parameter

Header: Table header.

DeviceId: Device Id of ODD. It has to match with the ID of MXM model which is used in GPIO Device Tables.

EventPin1: GEVENT pin for FCH_ODD_DA.

EventPin2: GEVENT pin for ODD_PLUGIN#.

EventPin3: Dummy GEVENT pin to workaround hang issue in old OS when _PRW is defined even if _STA return 0. This pin should not be used for other purpose.

SataModeSupportMask SATA mode Zero Power Odd is supported.
- BIT0: IDE Mode
- BIT1: AHCI Mode
- BIT2: RAID or RAID-5 Mode
- BIT3: AMD AHCI Mode

SataPortId SATA port number Odd is connected.

2.11 AMD CPM Save Context Table

This table is used to define the area to save CPM context. It could be defined to BIOS Ram or CMOS.

Prototype

```
typedef struct {
  AMD_CPM_TABLE_COMMON_HEADER    Header;
  UINT8                          BufferType;
  UINT8                          BufferOffset;
  UINT8                          BufferSize;
} AMD_CPM_SAVE_CONTEXT_TABLE;
```

Parameter

Header:	Table header.
BufferType:	Buffer Type. 5: BIOS RAM. 6: CMOS RAM. Other: Reserved
BufferOffset:	Offset of Buffer.
BufferSize:	Size of Buffer

Prototype

```
typedef struct {
  UINT32                         PcieDeviceStatus;
  UINT32                         PcieClockSlotStatus;
  UINT32                         WirelessButtonStatus;
  UINT32                         BootMode;
} AMD_CPM_PREDEFINED_SAVE_CONTEXT;
```

Parameter

PcieDeviceStatus:	The status of PCIe device on APU
PcieClockSlotStatus:	The status of PCIe device slot.
WirelessButtonStatus:	The status of Wireless button
BootMode:	The boot mode: 0: S0. 3: S3. 4: S4

Chapter 3 CPM PEI/DXE/SMM Driver for Kernel

3.1 AmdCpmInitPeim

This PEIM will perform CPM initialization in PEI early stage, and then publish the AMD_CPM_TABLE_PPI. This allows any component depending upon CPM initialization an opportunity to access CPM table and invoke common functions in PEI stage. AMD_CPM_TABLE_HOB_PPI will only be installed to store CPM tables temporary in S3 resume.

This PEIM will get the CPM tables by AMD_CPM_OEM_TABLE_PPI. These tables will be re-organized by the usage. If the table will be read only, it will be kept in ROM area. If it will be modified and only used in PEI stage, the table will be moved to cache. Otherwise, the table will be stored in Hob. If AMD_CPM_PRE_INIT_TABLE is defined, the register in this table will be initialized.

This PEIM consumes the following events:

- AMD_CPM_OEM_TABLE_PPI
- PEI_SMBUS_PPI
- PEI_PERMANENT_MEMORY_INSTALLED_PPI

This PEIM produces the following events (PPIs):

- AMD_CPM_TABLE_PPI
- AMD_CPM_TABLE_HOB_PPI

This PEIM depends on the following events (PPIs):

- AMD_CPM_OEM_TABLE_PPI
- PEI_SMBUS_PPI
- PEI_PERMANENT_MEMORY_INSTALLED_PPI

AMD_CPM_TABLE_PPI (Public)

GUID

```
#define AMD_CPM_TABLE_PPI_GUID \
```

```
  { 0xd71cf893, 0xa8b5, 0x49d3, 0xa2, 0x1b, 0x31, 0xe2, 0xf5, 0xc4, 0xa7,
0x47 }
```

PPI Interface Structure

```
typedef struct _AMD_CPM_TABLE_PPI {
  UINTN                                 Revision;
  AMD_CPM_MAIN_TABLE                    *MainTablePtr;
  AMD_CPM_CHIP_ID                       ChipId;
  AMD_CPM_COMMON_FUNCTION               CommonFunction;
  AMD_CPM_PEIM_PUBLIC_FUNCTION          PeimPublicFunction;
} AMD_CPM_TABLE_PPI;
```

Parameters

Revision	Revision number for this PEIM driver.
MainTablePtr	The pointer of CPM Main Table.
ChipId	The Chip ID.
CcommonFunction	The private function in PEI stage.
PeimPublicFunction	The public function in PEI stage.

The following structure is used to define public functions in this PPI.

Prototype

```
typedef struct _AMD_CPM_PEIM_PUBLIC_FUNCTION {
  AMD_CPM_SETMEMVOLTAGE_FN        SetMemVoltage;
  AMD_CPM_SETVDDPVDDRVOLTAGE_FN   SetVddpVddrVoltage;
  AMD_CPM_PCIERESET_FN            PcieReset;
  PCIe_COMPLEX_DESCRIPTOR         *PcieComplexDescriporPtr;
} AMD_CPM_PEIM_PUBLIC_FUNCTION;
```

Parameter

SetMemVoltage:	The function to set memory voltage.
SetVddpVddrVoltage:	The function to set VDDP/VDDR voltage.
PcieReset:	The function to reset PCIe device.
PcieComplexDescriporPtr:	The pointer of PCIE Complex Descriptor.

The following structure is used to define private functions in this PPI. These functions will be used in different CPM PEI Drivers.

Prototype

```
typedef struct _AMD_CPM_COMMON_FUNCTION {
  AMD_CPM_IOREAD8_FN              IoRead8
  AMD_CPM_IOREAD16_FN             IoRead16;
  AMD_CPM_IOREAD32_FN             IoRead32;
  AMD_CPM_IOWRITE8_FN             IoWrite8;
  AMD_CPM_IOWRITE16_FN            IoWrite16;
  AMD_CPM_IOWRITE32_FN            IoWrite32;
  AMD_CPM_MMIOREAD8_FN            MmioRead8;
  AMD_CPM_MMIOREAD16_FN           MmioRead16;
  AMD_CPM_MMIOREAD32_FN           MmioRead32;
  AMD_CPM_MMIOWRITE8_FN           MmioWrite8;
  AMD_CPM_MMIOWRITE16_FN          MmioWrite16;
  AMD_CPM_MMIOWRITE32_FN          MmioWrite32;
  AMD_CPM_MMIOAND8_FN             MmioAnd8;
  AMD_CPM_MMIOAND16_FN            MmioAnd16;
  AMD_CPM_MMIOAND32_FN            MmioAnd32;
  AMD_CPM_MMIOOR8_FN              MmioOr8;
  AMD_CPM_MMIOOR16_FN             MmioOr16;
```

```
AMD_CPM_MMIOOR32_FN             MmioOr32;
AMD_CPM_MMIOANDTHENOR8_FN       MmioAndThenOr8;
AMD_CPM_MMIOANDTHENOR16_FN      MmioAndThenOr16;
AMD_CPM_MMIOANDTHENOR32_FN      MmioAndThenOr32;
AMD_CPM_MSRREAD_FN              MsrRead;
AMD_CPM_MSRWRITE_FN             MsrWrite;
AMD_CPM_PCIREAD8_FN             PciRead8;
AMD_CPM_PCIREAD16_FN            PciRead16;
AMD_CPM_PCIREAD32_FN            PciRead32;
AMD_CPM_PCIWRITE8_FN            PciWrite8;
AMD_CPM_PCIWRITE16_FN           PciWrite16;
AMD_CPM_PCIWRITE32_FN           PciWrite32;
AMD_CPM_PCIWRITE8_FN            PciAnd8;
AMD_CPM_PCIWRITE16_FN           PciAnd16;
AMD_CPM_PCIWRITE32_FN           PciAnd32;
AMD_CPM_PCIWRITE8_FN            PciOr8;
AMD_CPM_PCIWRITE16_FN           PciOr16;
AMD_CPM_PCIWRITE32_FN           PciOr32;
AMD_CPM_PCIANDTHENOR8_FN        PciAndThenOr8;
AMD_CPM_PCIANDTHENOR16_FN       PciAndThenOr16;
AMD_CPM_PCIANDTHENOR32_FN       PciAndThenOr32;
AMD_CPM_READTSC_FN              ReadTsc;
AMD_CPM_CPUIDREAD_FN            CpuidRead;
AMD_CPM_POSTCODE_FN             PostCode;
AMD_CPM_CHECKPCIEDEVICE_FN      CheckPcieDevice;
AMD_CPM_DETECTDEVICE_FN         DetectDevice;
AMD_CPM_POWERONDEVICE_FN        PowerOnDevice;
```

```
AMD_CPM_GETDEVICECONFIG_FN      GetDeviceConfig;
AMD_CPM_KBCREAD_FN              KbcRead;
AMD_CPM_KBCWRITE_FN             KbcWrite;
AMD_CPM_GETRTC_FN               GetRtc;
AMD_CPM_SETRTC_FN               SetRtc;
AMD_CPM_GETACPI_FN              GetAcpi;
AMD_CPM_SETACPI_FN              SetAcpi;
AMD_CPM_GETGPIO_FN              GetGpio
AMD_CPM_SETGPIO_FN              SetGpio;
AMD_CPM_GETGEVENT_FN            GetGevent;
AMD_CPM_SETGEVENT_FN            SetGevent;
AMD_CPM_SETSMICONTROL_FN        SetSmiControl;
AMD_CPM_SETGEVENTSCITRIG_FN     SetGeventSciTrig;
AMD_CPM_SETGEVENTSCI_FN         SetGeventSci;
AMD_CPM_GETSTRAP_FN             GetStrap;
AMD_CPM_SETCLKREQ_FN            SetClkReq;
AMD_CPM_STALL_FN                Stall;
AMD_CPM_SETFANON_FN             SetFanOn;
AMD_CPM_SETPROCHOT_FN           SetProchot;
AMD_CPM_GETSATAMODE_FN          GetSataMode;
AMD_CPM_ISFCHDEVICE_FN          IsFchDevice;
AMD_CPM_GETSCIMAP_FN            GetSciMap;
AMD_CPM_GETCPUREVISIONID_FN     GetCpuRevisionId;
AMD_CPM_GETSBTSIADDR_FN         GetSbTsiAddr
AMD_CPM_ISTHERMALSUPPORT_FN     IsThermalSupport;
AMD_CPM_GETPCIEASLNAME_FN       GetPcieAslName;
AMD_CPM_GETPCIEASLNAME_FN       GetFchPcieAslName;
```

```
  AMD_CPM_GETBOOTMODE_FN          GetBootMode;
  AMD_CPM_ISRTCWAKEUP_FN          IsRtcWakeup;
  AMD_CPM_ISUMI_FN                IsUmi;
  AMD_CPM_GETTABLEPTR_FN          GetTablePtr;
  AMD_CPM_GETTABLEPTR_FN          GetTablePtr2;
  AMD_CPM_ADDTABLE_FN             AddTable;
  AMD_CPM_REMOVETABLE_FN          RemoveTable;
  AMD_CPM_SMBUSREAD_FN            ReadSmbus;
  AMD_CPM_SMBUSWRITE_FN           WriteSmbus;
  AMD_CPM_SMBUSREAD_FN            ReadSmbusBlock;
  AMD_CPM_SMBUSWRITE_FN           WriteSmbusBlock;
  AMD_CPM_RESETDEVICE_FN          ResetDevice;
  AMD_CPM_RELOCATETABLE_FN        RelocateTable;
  AMD_CPM_COPYMEM_FN              CopyMem;
  AMD_CPM_LOADPREINITTABLE_FN     LoadPreInitTable;
  AMD_CPM_ADDSSDTTABLE_FN         AddSsdtTable;
  AMD_CPM_ISAMLOPREGIONOBJECT_FN  IsAmlOpRegionObject;
  AMD_CPM_SETSAVECONTEXT_FN       SetSaveContext;
  AMD_CPM_GETSAVECONTEXT_FN       GetSaveContext;
} AMD_CPM_COMMON_FUNCTION;
```

AMD_CPM_TABLE_HOB_PPI (Private)

GUID

```
#define AMD_CPM_TABLE_HOB_PPI_GUID \
  { 0xc02c596b, 0xcd04, 0x486e, 0x86, 0x66, 0x30, 0x3e, 0x55, 0x67, 0xc0,
0x48 }
```

3.2 AmdCpmInitDxe

This DXE module will perform CPM initialization in DXE early stage, and then publish the AMD_CPM_TABLE_PROTOCOL. This allows any component depending upon CPM initialization an opportunity to access CPM table, decode the command table and run some other common function in DXE stage. It will merge the data which is passed from Hob and the new table which is obtained by AMD_CPM_OEM_TABLE_PROTOCOL. This module will also allocate a common buffer in ACPI memory and provide a SSDT table to access the buffer in ASL code. It also provides some common ACPI methods in this SSDT table.

This DXE module consumes the following events:

- AMD_CPM_OEM_TABLE_PROTOCOL
- EFI_ACPI_SUPPORT_PROTOCOL
- EFI_SMBUS_HC_PROTOCOL
- EFI_FIRMWARE_VOLUME_PROTOCOL

This DXE module produces the following events:

- AMD_CPM_TABLE_PROTOCOL
- AMD_CPM_NV_DATA_PROTOCOL

AMD_CPM_TABLE_PROTOCOL (Public)

GUID

```
#define AMD_CPM_TABLE_PROTOCOL_GUID \

  { 0x3724cf01, 0x00c2, 0x9762, 0x11, 0xb3, 0x0e, 0xa8, 0xaa, 0x89, 0x72,
0x00 }
```

Protocol Prototype

```
typedef struct _AMD_CPM_DXE_PUBLIC_FUNCTION {

  AMD_CPM_GETPOSTEDVBIOSIMAGE_FN      GetPostedVbiosImage;

} AMD_CPM_DXE_PUBLIC_FUNCTION;
```

Parameters

GetPostedVbiosImage The method to get posted VBIOS image.

```
typedef struct _AMD_CPM_TABLE_PROTOCOL {
  UINTN                                 Revision;
  AMD_CPM_MAIN_TABLE                    *MainTablePtr;
  AMD_CPM_CHIP_ID                       ChipId;
  AMD_CPM_COMMON_FUNCTION               CommonFunction;
  AMD_CPM_DXE_PUBLIC_FUNCTION           DxePublicFunction;
} AMD_CPM_TABLE_PROTOCOL;
```

Parameters

Revision Revision number for this DXE driver.

MainTablePtr The pointer of CPM Main Table

ChipId The Chip ID

CommonFunction The private function in DXE stage.

DxePublicFunction Public function of Protocol

AMD_CPM_NV_DATA_PROTOCOL (Private)

GUID

```
#define AMD_CPM_NV_DATA_PROTOCOL_GUID \
  { 0xdb4a79ac, 0x87ba, 0x4625, 0x6a, 0x9e, 0xfe, 0xbf, 0x9d, 0x6d, 0x95,
    0xeb }
```

Protocol Prototype

```
typedef struct _AMD_CPM_TABLE_PROTOCOL {
    UINTN                   Revision;
    AMD_CPM_NV_DATA_STRUCT  *NvDataPtr;
  } AMD_CPM_TABLE_PROTOCOL;
```

Parameters

Revision	Revision number for this DXE driver.
NvDataPtr	The Pointer of NV Data Buffer.

3.3 AmdCpmInitSmm

This module will duplicate AMD_CPM_TABLE_PROTOCOL from AmdCpmInitDxe and install in for other SMM driver to use. It also registers the command function which can be called in other SMM driver.

This SMM module consumes the following events:

- AMD_CPM_TABLE_PROTOCOL
- AMD_CPM_NV_DATA_PROTOCOL

This SMM module produces the following events:

- AMD_CPM_TABLE_SMM_PROTOCOL

AMD_CPM_TABLE_SMM_PROTOCOL (Public)

GUID

```
#define AMD_CPM_TABLE_SMM_PROTOCOL_GUID \
```

```
  { 0xaf6efacf, 0x7a13, 0x45a3, 0xb1, 0xa5, 0xaa, 0xfc, 0x06, 0x1c, 0x4b,
0x79 }


typedef struct _AMD_CPM_TABLE_PROTOCOL {

  UINTN                                 Revision;

  AMD_CPM_MAIN_TABLE                    *MainTablePtr;

  AMD_CPM_CHIP_ID                       ChipId;

  AMD_CPM_COMMON_FUNCTION               CommonFunction;

  AMD_CPM_DXE_PUBLIC_FUNCTION           DxePublicFunction;

} AMD_CPM_TABLE_PROTOCOL;
```

Parameters

Revision	Revision number for this DXE driver.
MainTablePtr	The pointer of CPM Main Table
ChipId	The Chip ID
CommonFunction	The private function in DXE stage.
DxePublicFunction	Public function of Protocol

Chapter 4 CPM OEM PEI Driver Overview

4.1 AmdCpmOemInitPeim

This PEIM is responsible to define platform specific table for CPM in PEI stage and it is mandatory. It publishes the AMD_CPM_OEM_TABLE_PPI and registers for a callback upon publication of AMD_CPM_TABLE_PPI.

This PEIM consumes the following events:

- AMD_CPM_TABLE_PPI

This PEIM produces the following events (PPIs):

- AMD_CPM_OEM_TABLE_PPI

AMD_CPM_OEM_TABLE_PPI (Public)

GUID

```
#define AMD_CPM_OEM_TABLE_PPI_GUID \

  { 0xfd1fe103, 0x40f1, 0x459c, 0x98, 0x3e, 0x11, 0x0b, 0x69, 0x5e, 0xd1,
0x1a }
```

PPI Interface Structure

```
typedef struct _AMD_CPM_OEM_TABLE_PPI {

  UINTN                                 Revision;

  UINT16                                PlatformId;

  void                                  *TablePtr;

} AMD_CPM_OEM_TABLE_PPI;
```

Parameters

Revision	Revision number for this PEIM driver.
PlatformId	The Default Platform ID.
TablePtr	The Pointer of CPM Table List.

Chapter 5 CPM PEI/DXE/SMM Drivers for Feature

5.1 The Drivers for ACPI Thermal Fan

The following tables need to be defined to support ACPI Thermal Fan.

- AMD_CPM_MAIN_TABLE
  - The fields in main table need to be defined: CurrentPlatformId, PcieMemIoBaseAddr, AcpiMemIoBaseAddr, AcpiThermalFanEn
- AMD_CPM_ACPI_THERMAL_FAN_TABLE

The following drivers are used to implement the feature of ACPI Thermal Fan.

- AmdCpmAcpiThermalFanPeim
  - This driver is responsible to disable Thermal Fan Control in BIOS early post and force the fan to run in full speed. Thermal Fan Control will be enabled when the system boots up to ACPI OS.
- AmdCpmAcpiThermalFanDxe
  - This DXE driver is responsible for initializing ACPI Thermal Fan Feature. The SSDT for ACPI Thermal Fan will be patched according to the platform design. Thermal Fan Policy and other HW setting will be passed to NV Data Table which will be referred by ACPI method in SSDT

The following drivers also need to be initialized before the feature is implemented.

- AmdCpmOemInitPeim
- AmdCpmInitPeim
- AmdCpmInitDxe

5.2 The Drivers for Adaptive S4

The following tables need to be defined to support ACPI Thermal Fan.

- AMD_CPM_MAIN_TABLE
  - The fields in main table need to be defined: CurrentPlatformId, PcieMemIoBaseAddr, AcpiMemIoBaseAddr, AdaptiveS4En
- AMD_CPM_ADAPTIVE_S4_TABLE
- AMD_CPM_SAVE_CONTEXT_TABLE

The following drivers are used to implement the feature of Adaptive S4.

- AmdCpmAdaptiveS4Peim
  - This driver is used to check whether the system is waking up from Adaptive S4 RTC mode.
- AmdCpmAdaptiveS4Dxe
  - This driver is responsible for installing Adaptive S4 SSDT table.
- AmdCpmAdaptiveS4Smm
  - This driver is used to set RTC alarm if the system goes to Adaptive S4 and RTC mode is enabled.

The following drivers also need to be initialized before the feature is implemented.

- AmdCpmOemInitPeim
- AmdCpmInitPeim
- AmdCpmInitDxe
- AmdCpmInitSmm

5.3 The Driver for Boot Time Record

Boot Time Record Module is used to record time stamp in BIOS post time and S3/Resume.

5.4 The Drivers for Display Feature

The following tables need to be defined to support Display Feature.

- AMD_CPM_MAIN_TABLE
  - The fields in main table need to be defined: CurrentPlatformId, PcieMemIoBaseAddr, AcpiMemIoBaseAddr, DisplayFeature
- AMD_CPM_DISPLAY_FEATURE_TABLE
- AMD_CPM_DEVICE_PATH_TABLE
- AMD_CPM_SPECIFIC_SSID_TABLE (Optional)
- AMD_CPM_GPIO_DEVICE_CONFIG_TABLE
  - Define the initialize sequence of MXM module in BIOS post.
- AMD_CPM_GPIO_DEVICE_DETECTION_TABLE
  - Define the detection sequence of MXM module
- AMD_CPM_GPIO_DEVICE_RESET_TABLE
  - Define the reset sequence of MXM module
- AMD_CPM_GPIO_DEVICE_POWER_TABLE
  - Define the power on/off sequence of MXM module
- AMD_CPM_PCIE_TOPOLOGY_TABLE
  - Display Feature Module will override the LinkHotplug of MXM module in PCIE Topology Table.

The following drivers are used to implement the feature of Display Feature.

- AmdCpmDisplayFeaturePeim
  - This driver is responsible to set LinkHotplug in PCIE Port Descriptor if PowerXpress is enabled.
- AmdCpmDisplayFeatureDxe
  - This DXE driver is responsible for initializing Display Feature. The sequence is divided two different stages. The first one is to register an event handling function which will be launched after AllPciIoPrtclsInstlFinished Protocol is installed. In this event function, special VBIOS post will be done and the image will be stored in frame buffer. The second one is to register an event which is linked with gEfiEventReadyToBootGuid. In this event, Special SSID will be set according to the display feature to be enabled. The SSDT tables for display feature will be registered and the special-posted VBIOS image and other parameters will be uploaded to ACPI area

- AmdCpmDisplayFeatureSmm

This SMM driver disables Audio Device in dGPU if PowerXpress is enabled.

The following drivers also need to be initialized before the feature is implemented.

- AmdCpmOemInitPeim
- AmdCpmInitPeim
- AmdCpmGpioInitPeim
- AmdCpmPcieInitPeim
- AmdCpmInitDxe
- AmdCpmInitSmm

5.5 The Driver for CPM EC Init

The following tables need to be defined to support EC Init.

- AMD_CPM_MAIN_TABLE
  - The fields in main table need to be defined: CurrentPlatformId, PcieMemIoBaseAddr, AcpiMemIoBaseAddr, Ec

The driver AmdCpmEcInitPeim is used to initialize external EC controller. It will send the command sequence to EC controller and enable/disable S5+ on Battery mode.

The following drivers also need to be initialized before the feature is implemented.

- AmdCpmOemInitPeim
- AmdCpmInitPeim
- AmdCpmInitDxe

5.6 The Driver for GPIO Init

The following tables need to be defined to support GPIO Init.

- AMD_CPM_MAIN_TABLE

  - The fields in main table need to be defined: CurrentPlatformId, PcieMemIoBaseAddr, AcpiMemIoBaseAddr, ExtClkGen, UnusedGppClkOffEn.

- AMD_CPM_GPIO_INIT_TABLE
- AMD_CPM_GEVENT_INIT_TABLE
- AMD_CPM_GPIO_DEVICE_CONFIG_TABLE
- AMD_CPM_GPIO_DEVICE_DETECTION_TABLE
- AMD_CPM_GPIO_DEVICE_RESET_TABLE
- AMD_CPM_GPIO_DEVICE_POWER_TABLE
- AMD_CPM_GPIO_MEM_VOLTAGE_TABLE
- AMD_CPM_PCIE_CLOCK_TABLE
- AMD_CPM_EXT_CLKGEN_TABLE
- AMD_CPM_SAVE_CONTEXT_TABLE

The following drivers are used to implement the feature of Display Feature.

- AmdCpmGpioInitPeim
  - This driver is responsible to set GPIO and GEVENT registers and initialize on-board devices. It will also register the functions to set memory voltage and reset PCIE devices, which will be called by AGESA callback functions. Set PCIE Clock and ClkReq in GPIO Initialization Stage Two if External ClkGen is used.
- AmdCpmGpioInitDxe
  - Set PCIE Clock and ClkReq in BIOS post.
- AmdCpmGpioInitSmm
  - Set PCIE Clock and ClkReq in S3/Resume.

The following drivers also need to be initialized before the feature is implemented.

- AmdCpmOemInitPeim
- AmdCpmInitPeim
- AmdCpmInitDxe
- AmdCpmInitSmm

5.7 The Drivers for PCIE Init

The following tables need to be defined to support GPIO Init.

- AMD_CPM_MAIN_TABLE
  - The fields in main table need to be defined: CurrentPlatformId, PcieMemIoBaseAddr, AcpiMemIoBaseAddr
- AMD_CPM_PCIE_TOPOLOGY_TABLE
- AMD_CPM_PCIE_TOPOLOGY_OVERRIDE_TABLE (Optional)
- AMD_CPM_GPIO_DEVICE_RESET_TABLE
- AMD_CPM_EXPRESS_CARD_TABLE
- AMD_CPM_SAVE_CONTEXT_TABLE

The following drivers are used to implement the feature of PCIE Init.

- AmdCpmPcieInitPeim
  - This driver is responsible to generate PCIE Complex Descriptor Table, which will be an input parameter of AGESA.
- AmdCpmPcieInitDxe
  - This driver is used to setup the SSDT table to support Express Card.

The following drivers also need to be initialized before the feature is implemented.

- AmdCpmOemInitPeim
- AmdCpmInitPeim
- AmdCpmInitDxe

5.8 The Driver for Zero Power Odd

The following tables need to be defined to support Zero Power Odd.

- AMD_CPM_MAIN_TABLE
  - The fields in main table need to be defined: CurrentPlatformId, PcieMemIoBaseAddr, AcpiMemIoBaseAddr, ZeroPowerOddEn.
- AMD_CPM_ZERO_POWER_ODD_TABLE

The following drivers are used to implement the feature of Zero Power Odd.

- AmdCpmZeroPowerOddPeim
  - This driver is responsible to set the trigger status of the GEVENT pins which is used for Zero Power Odd.
- AmdCpmZeroPowerOddDxe
  - This DXE driver is responsible for initializing Zero Power ODD Feature. The SSDT for Zero Power ODD will be patched according to the platform design. Some other parameters will be passed to NV Data Table which will be referred by ACPI method in SSDT.

The following drivers also need to be initialized before the feature is implemented.

- AmdCpmOemInitPeim
- AmdCpmInitPeim
- AmdCpmInitDxe

Chapter 6 Sample code

6.1 AMD CPM Build Options

The following options needs to be defined in the build file.

- Define CPM root folder

  ```
  CPM_ROOT    = $(AMD_COMMON_PLATFORM_DIR)\Cpm
  ```

- Define the folder of CPM OEM driver

  ```
  CPM_OEMDIR = $(AMD_COMMON_PLATFORM_DIR)\Cpm\Addendum\Oem\Rathmore
  ```

- Define common build option of CPM PEIM driver

  ```
  CPM_PEIM_BUILD_OPTION         = BUILD_TYPE=$(NO_DPX)TE_PEIM
  ```

- Define common build option of CPM DXE & SMM driver

  ```
  CPM_DXE_BUILD_OPTION          =
  ```

- Define the folder of CPM OEM option

  ```
  CPM_OPTSDIR                   = $(PROJECT_OEM_TIP)\CpmPlatformLib
  ```

- Define ASL component type in ASL .INF file

  ```
  CPM_ASL_COMPONENT_TYPE        = ACPI_COMMON_ASL
  ```

- Define whether “ACPI_SECTIONS” needs to be defined in ASL .INF file

  ```
  CPM_ACPI_SECTIONS_SUPPORT     = YES
  ```

6.2 AMD CPM OEM Table Sample

```
//
// OEM CPM Table Definition
//

//
// Platform Id Table: Get Board Id from GPIO pins
//
AMD_CPM_PLATFORM_ID_TABLE       gCpmPlatformIdTable = {
  {CPM_SIGNATURE_GET_PLATFORM_ID, sizeof(gCpmPlatformIdTable)/sizeof(UINT8), 0, 0, 0, 1},
  {
    14,                         // BOARD_ID0: GPIO14
    15,                         // BOARD_ID1: GPIO15
    16,                         // BOARD_ID2: GPIO16
    17,                         // BOARD_ID3: GPIO17
    18,                         // BOARD_ID4: GPIO18
    19,                         // BOARD_ID5: GPIO19
    0xFF
  }
};
```

```
//
// Convert Table from Board Id to Platform Id
//
AMD_CPM_PLATFORM_ID_CONVERT_TABLE  gCpmPlatformIdConvertTable = {
  {CPM_SIGNATURE_GET_PLATFORM_ID_CONVERT, sizeof(gCpmPlatformIdConvertTable)/sizeof(UINT8), 0, 0, 0, 1},
  {
    // CpuRevisionId, OriginalIdMask, OriginalId, ConvertedId
    {0x00,   0x0000,   0x0000,   0x0000},                  // Board Id -> Platform Id
    0xFFFF,
  }
};

//
// Pre-Init Table
//
AMD_CPM_PRE_INIT_TABLE          gCpmPreInitTable = {
  {CPM_SIGNATURE_PRE_INIT, sizeof(gCpmPreInitTable)/sizeof(UINT8), 0, 0, 0, 0x00000001},
  {
    {  0x00,  0x03,  0xEA,  0xFE,  0x01  },       // PM_RegEA[0]: PCIDisable = 1
    {  0x00,  0x03,  0x2E,  0xF9,  0x00  },       // PM_Reg2E[2:1]: Smbus0Sel = 0
    {  0x00,  0x03,  0xBE,  0xED,  0x12  },       // PM_RegBE[1,4]: Enable KbRst
    {  0x01,  0xA3,  0x78,  0xF7,  0x00  },       // LPC Reg78[3]: Disable LDRQ1#
    {  0x02,  0xC3,  0xE4,  0x8F,  0x00  },       // APU_MISC Reg1E4[6:4] = 0
    {  0x01,  0xC3,  0xA4,  0x00,  0xEF  },       // APU_MISC RegA4 = 0xEF
    {  0x01,  0xC3,  0xA5,  0x00,  0x0F  },       // APU_MISC RegA5 = 0x0F
    0xFF,
  }
};

//
// GPIO Init Table
//
AMD_CPM_GPIO_INIT_TABLE          gCpmGpioInitTable = {
  {CPM_SIGNATURE_GPIO_INIT, sizeof(gCpmGpioInitTable)/sizeof(UINT8), 0, 0, 0, 0x00000001},
  {
    GPIO_DEFINITION(0,    GPIO_FUNCTION_0,   GPIO_OUTPUT_HIGH,  GPIO_PU_EN,    GPIO_STICKY_EN),   // MPCIE_RST1#
    GPIO_DEFINITION(1,    GPIO_FUNCTION_0,   GPIO_OUTPUT_HIGH,  GPIO_PU_EN,    GPIO_STICKY_EN),   // MPCIE_RST2#
    GPIO_DEFINITION(2,    GPIO_FUNCTION_0,   GPIO_OUTPUT_HIGH,  GPIO_PU_EN,    GPIO_STICKY_EN),   // DMC_RST0#
    GPIO_DEFINITION(4,    GPIO_FUNCTION_0,   GPIO_OUTPUT_HIGH,  GPIO_PU_EN,    GPIO_STICKY_EN),   // MPCIE_RST_DT#
    GPIO_DEFINITION(7,    GPIO_FUNCTION_0,   GPIO_NA,           GPIO_PU_EN,    GPIO_STICKY_EN),   // BT_ON
    GPIO_DEFINITION(8,    GPIO_FUNCTION_0,   GPIO_NA,           GPIO_PD_EN,    GPIO_STICKY_DIS),  // PEX_STD_SW#
    GPIO_DEFINITION(12,   GPIO_FUNCTION_0,   GPIO_NA,           GPIO_PU_EN,    GPIO_STICKY_EN),   // WL_DISABLE#
    GPIO_DEFINITION(13,   GPIO_FUNCTION_0,   GPIO_NA,           GPIO_PU_EN,    GPIO_STICKY_EN),   // WU_DISABLE#
    GPIO_DEFINITION(22,   GPIO_FUNCTION_0,   GPIO_NA,           GPIO_PU_EN,    GPIO_STICKY_EN),   // FCH_PWR_LV
    GPIO_DEFINITION(25,   GPIO_FUNCTION_0,   GPIO_OUTPUT_HIGH,  GPIO_PU_EN,    GPIO_STICKY_EN),   // PCIE_RST#_LAN
    GPIO_DEFINITION(27,   GPIO_FUNCTION_0,   GPIO_OUTPUT_HIGH,  GPIO_PU_EN,    GPIO_STICKY_EN),   // DDI3_PCIE_RST#
    GPIO_DEFINITION(41,   GPIO_FUNCTION_1,   GPIO_NA,           GPIO_PU_EN,    GPIO_STICKY_DIS),  // FCH_PCIE_PE2_CLKREQ#
    GPIO_DEFINITION(42,   GPIO_FUNCTION_1,   GPIO_NA,           GPIO_PU_EN,    GPIO_STICKY_DIS),  // FCH_PCIE_DT_CLKREQ#
    GPIO_DEFINITION(53,   GPIO_FUNCTION_1,   GPIO_OUTPUT_HIGH,  GPIO_PU_EN,    GPIO_STICKY_EN),   // MPCIE_RST_XPRESS#
    GPIO_DEFINITION(54,   GPIO_FUNCTION_1,   GPIO_OUTPUT_HIGH,  GPIO_PU_EN,    GPIO_STICKY_DIS),  // FCH_PROCHOT#_C
    GPIO_DEFINITION(55,   GPIO_FUNCTION_2,   GPIO_NA,           GPIO_PD_EN,    GPIO_STICKY_DIS),  // MXM_PWR_EN
    GPIO_DEFINITION(57,   GPIO_FUNCTION_1,   GPIO_INPUT,        GPIO_PU_EN,    GPIO_STICKY_DIS),  // MLDIR
    GPIO_DEFINITION(59,   GPIO_FUNCTION_2,   GPIO_INPUT,        GPIO_PD_EN,    GPIO_STICKY_DIS),  // FFS_INT1
    GPIO_DEFINITION(171,  GPIO_FUNCTION_1,   GPIO_NA,           GPIO_PU_EN,    GPIO_STICKY_EN),   // ODD_PWR
    GPIO_DEFINITION(172,  GPIO_FUNCTION_1,   GPIO_INPUT,        GPIO_PU_EN,    GPIO_STICKY_DIS),  // DMC_PRESENT#
    GPIO_DEFINITION(175,  GPIO_FUNCTION_1,   GPIO_NA,           GPIO_PD_EN,    GPIO_STICKY_EN),   // DMC_PD
    GPIO_DEFINITION(176,  GPIO_FUNCTION_1,   GPIO_NA,           GPIO_PD_EN,    GPIO_STICKY_EN),   // MPCIE_PD1
    GPIO_DEFINITION(177,  GPIO_FUNCTION_1,   GPIO_NA,           GPIO_PD_EN,    GPIO_STICKY_EN),   // MPCIE_PD2
    GPIO_DEFINITION(189,  GPIO_FUNCTION_0,   GPIO_NA,           GPIO_PU_EN,    GPIO_STICKY_EN),   // MEM_1V25#
    GPIO_DEFINITION(190,  GPIO_FUNCTION_0,   GPIO_NA,           GPIO_PU_EN,    GPIO_STICKY_EN),   // MEM_1V5#
    GPIO_DEFINITION(191,  GPIO_FUNCTION_0,   GPIO_NA,           GPIO_PU_EN,    GPIO_STICKY_EN),   // PE_GPIO0
    GPIO_DEFINITION(192,  GPIO_FUNCTION_0,   GPIO_NA,           GPIO_PD_EN,    GPIO_STICKY_EN),   // PE_GPIO1
    GPIO_DEFINITION(198,  GPIO_FUNCTION_0,   GPIO_OUTPUT_HIGH,  GPIO_PU_EN,    GPIO_STICKY_EN),   // HDD2_PWR
    GPIO_DEFINITION(199,  GPIO_FUNCTION_0,   GPIO_NA,           GPIO_PU_EN,    GPIO_STICKY_EN),   // WP_DISABLE#
```

```
    GPIO_DEFINITION(200,  GPIO_FUNCTION_0,  GPIO_OUTPUT_HIGH,  GPIO_PU_EN,    GPIO_STICKY_EN),  // HDD0_PWR

    GPIO_DEFINITION(0x64,  GPIO_FUNCTION_2,  GPIO_NA,        GPIO_NA,      GPIO_STICKY_DIS),  // FCH_PCIE_RST#
    GPIO_DEFINITION(0x66,  GPIO_FUNCTION_1,  GPIO_NA,        GPIO_PU_EN,   GPIO_STICKY_DIS),  // FCH_ODD_DA
    GPIO_DEFINITION(0x67,  GPIO_FUNCTION_1,  GPIO_OUTPUT_LOW,  GPIO_PD_EN,   GPIO_STICKY_EN),  // VGA_PD
    GPIO_DEFINITION(0x6B,  GPIO_FUNCTION_1,  GPIO_NA,        GPIO_PU_PD_DIS, GPIO_STICKY_DIS),  // DP_HPD_DIG
    GPIO_DEFINITION(0x6C,  GPIO_FUNCTION_1,  GPIO_NA,        GPIO_NA,      GPIO_STICKY_DIS),  // WF_RADIO
    GPIO_DEFINITION(0x6D,  GPIO_FUNCTION_1,  GPIO_NA,        GPIO_PU_EN,   GPIO_STICKY_DIS),  // LID_CLOSED#
    GPIO_DEFINITION(0x6E,  GPIO_FUNCTION_1,  GPIO_NA,        GPIO_NA,      GPIO_STICKY_DIS),  // TALERT#_FCH
    GPIO_DEFINITION(0x6F,  GPIO_FUNCTION_1,  GPIO_NA,        GPIO_NA,      GPIO_STICKY_DIS),  // AC_PRES_OK#
    GPIO_DEFINITION(0x70,  GPIO_FUNCTION_1,  GPIO_NA,        GPIO_PU_EN,   GPIO_STICKY_DIS),  // ODD_PLUGIN#
    GPIO_DEFINITION(0x77,  GPIO_FUNCTION_0,  GPIO_NA,        GPIO_PD_EN,   GPIO_STICKY_DIS),  // FFS_INT2

    0xFF,
  }
};

//
// GEVENT Init Table
//
AMD_CPM_GEVENT_INIT_TABLE        gCpmGeventInitTable = {
  {CPM_SIGNATURE_GEVENT_INIT, sizeof(gCpmGeventInitTable)/sizeof(UINT8), 0, 0, 0xFFFFFFFF, 0x00000001},
  {  //                    PinNum     EventEnable   SciTrigE          SciLevl          SmiSciEn          SciS0En          SciMap
SmiTrig       SmiControl
    GEVENT_DEFINITION( 0x00,  EVENT_DISABLE,  SCITRIG_LOW,  SCILEVEL_EDGE,  SMISCI_DISABLE, SCIS0_DISABLE,  SCIMAP_00,
SMITRIG_HI, SMICONTROL_DISABLE ),
    GEVENT_DEFINITION( 0x01,  EVENT_DISABLE,  SCITRIG_LOW,  SCILEVEL_EDGE,  SMISCI_DISABLE, SCIS0_DISABLE,  SCIMAP_01,
SMITRIG_HI, SMICONTROL_DISABLE ),
    GEVENT_DEFINITION( 0x02,  EVENT_DISABLE,  SCITRIG_LOW,  SCILEVEL_EDGE,  SMISCI_DISABLE, SCIS0_DISABLE,  SCIMAP_02,
SMITRIG_HI, SMICONTROL_DISABLE ),
    GEVENT_DEFINITION( 0x03,  EVENT_ENABLE,  SCITRIG_LOW,  SCILEVEL_EDGE,  SMISCI_DISABLE, SCIS0_DISABLE,  SCIMAP_03,
SMITRIG_HI, SMICONTROL_DISABLE ),
    GEVENT_DEFINITION( 0x04,  EVENT_DISABLE,  SCITRIG_LOW,  SCILEVEL_EDGE,  SMISCI_DISABLE, SCIS0_DISABLE,  SCIMAP_04,
SMITRIG_HI, SMICONTROL_DISABLE ),
    GEVENT_DEFINITION( 0x05,  EVENT_ENABLE,  SCITRIG_LOW,  SCILEVEL_EDGE,  SMISCI_DISABLE, SCIS0_DISABLE,  SCIMAP_05,
SMITRIG_HI, SMICONTROL_DISABLE ),
    GEVENT_DEFINITION( 0x06,  EVENT_ENABLE,  SCITRIG_LOW,  SCILEVEL_EDGE,  SMISCI_DISABLE, SCIS0_DISABLE,  SCIMAP_06,
SMITRIG_HI, SMICONTROL_DISABLE ),
    GEVENT_DEFINITION( 0x07,  EVENT_DISABLE,  SCITRIG_LOW,  SCILEVEL_EDGE,  SMISCI_DISABLE, SCIS0_DISABLE,  SCIMAP_07,
SMITRIG_HI, SMICONTROL_DISABLE ),
    GEVENT_DEFINITION( 0x08,  EVENT_DISABLE,  SCITRIG_LOW,  SCILEVEL_EDGE,  SMISCI_DISABLE, SCIS0_DISABLE,  SCIMAP_08,
SMITRIG_HI, SMICONTROL_DISABLE ),
    GEVENT_DEFINITION( 0x09,  EVENT_DISABLE,  SCITRIG_LOW,  SCILEVEL_EDGE,  SMISCI_DISABLE, SCIS0_DISABLE,  SCIMAP_09,
SMITRIG_HI, SMICONTROL_DISABLE ),
    GEVENT_DEFINITION( 0x0A,  EVENT_ENABLE,  SCITRIG_LOW,  SCILEVEL_EDGE,  SMISCI_DISABLE, SCIS0_DISABLE,  SCIMAP_10,
SMITRIG_HI, SMICONTROL_DISABLE ),
    GEVENT_DEFINITION( 0x0B,  EVENT_ENABLE,  SCITRIG_LOW,  SCILEVEL_EDGE,  SMISCI_DISABLE, SCIS0_DISABLE,  SCIMAP_11,
SMITRIG_HI, SMICONTROL_DISABLE ),
    GEVENT_DEFINITION( 0x0C,  EVENT_ENABLE,  SCITRIG_LOW,  SCILEVEL_EDGE,  SMISCI_DISABLE, SCIS0_DISABLE,  SCIMAP_12,
SMITRIG_HI, SMICONTROL_DISABLE ),
    GEVENT_DEFINITION( 0x0D,  EVENT_ENABLE,  SCITRIG_LOW,  SCILEVEL_EDGE,  SMISCI_DISABLE, SCIS0_DISABLE,  SCIMAP_13,
SMITRIG_HI, SMICONTROL_DISABLE ),
    GEVENT_DEFINITION( 0x0E,  EVENT_ENABLE,  SCITRIG_LOW,  SCILEVEL_EDGE,  SMISCI_DISABLE, SCIS0_DISABLE,  SCIMAP_14,
SMITRIG_HI, SMICONTROL_DISABLE ),
    GEVENT_DEFINITION( 0x0F,  EVENT_DISABLE,  SCITRIG_LOW,  SCILEVEL_EDGE,  SMISCI_DISABLE, SCIS0_DISABLE,  SCIMAP_15,
SMITRIG_HI, SMICONTROL_DISABLE ),
    GEVENT_DEFINITION( 0x10,  EVENT_ENABLE,  SCITRIG_LOW,  SCILEVEL_EDGE,  SMISCI_DISABLE, SCIS0_DISABLE,  SCIMAP_16,
SMITRIG_HI, SMICONTROL_DISABLE ),
    GEVENT_DEFINITION( 0x11,  EVENT_DISABLE,  SCITRIG_LOW,  SCILEVEL_EDGE,  SMISCI_DISABLE, SCIS0_DISABLE,  SCIMAP_17,
SMITRIG_HI, SMICONTROL_DISABLE ),
    GEVENT_DEFINITION( 0x12,  EVENT_DISABLE,  SCITRIG_LOW,  SCILEVEL_EDGE,  SMISCI_DISABLE, SCIS0_DISABLE,  SCIMAP_18,
SMITRIG_HI, SMICONTROL_DISABLE ),
    GEVENT_DEFINITION( 0x13,  EVENT_DISABLE,  SCITRIG_LOW,  SCILEVEL_EDGE,  SMISCI_DISABLE, SCIS0_DISABLE,  SCIMAP_19,
SMITRIG_HI, SMICONTROL_DISABLE ),
```

```
    GEVENT_DEFINITION( 0x14,  EVENT_DISABLE, SCITRIG_LOW,  SCILEVEL_EDGE, SMISCI_DISABLE, SCIS0_DISABLE, SCIMAP_20,
SMITRIG_HI, SMICONTROL_DISABLE ),
    GEVENT_DEFINITION( 0x15,  EVENT_DISABLE, SCITRIG_LOW,  SCILEVEL_EDGE, SMISCI_DISABLE, SCIS0_DISABLE, SCIMAP_21,
SMITRIG_HI, SMICONTROL_DISABLE ),
    GEVENT_DEFINITION( 0x16,  EVENT_ENABLE, SCITRIG_LOW,  SCILEVEL_EDGE, SMISCI_DISABLE, SCIS0_DISABLE, SCIMAP_22,
SMITRIG_HI, SMICONTROL_DISABLE ),
    GEVENT_DEFINITION( 0x17,  EVENT_ENABLE, SCITRIG_LOW,  SCILEVEL_EDGE, SMISCI_DISABLE, SCIS0_DISABLE, SCIMAP_23,
SMITRIG_HI, SMICONTROL_DISABLE ),
    0xFF,
  }
};

//
// Set Mem Voltage
//
AMD_CPM_GPIO_MEM_VOLTAGE_TABLE  gCpmSetMemVoltage = {
  {CPM_SIGNATURE_SET_MEM_VOLTAGE, sizeof(gCpmSetMemVoltage)/sizeof(UINT8), 0, 0 ,0, 1 },
  {
    {1, 190, 0, 189, 0},      // 1.5V     GPIO190 = 0          GPIO189 = 0
    {2, 190, 1, 189, 1},      // 1.35V    GPIO190 = 1          GPIO189 = 1
    {3, 190, 1, 189, 0},      // 1.25V    GPIO190 = 1          GPIO189 = 0
    0xFF,
  }
};

//
// Set Vddp/Vddr Voltage
//
AMD_CPM_GPIO_VDDP_VDDR_VOLTAGE_TABLE  gCpmSetVddpVddrVoltage = {
 {CPM_SIGNATURE_SET_VDDP_VDDR_VOLTAGE, sizeof (gCpmSetVddpVddrVoltage) / sizeof (UINT8), 0, 0 ,0, 0x01 },
 {
  {0, 197, 1},            // 0.95V
  {1, 197, 0},            // 1.05V
  0xFF,
 }
};

//
// Device Config Table
//

AMD_CPM_GPIO_DEVICE_CONFIG_TABLE   gCpmGpioDeviceConfigTable = {
  {CPM_SIGNATURE_GPIO_DEVICE_CONFIG, sizeof(gCpmGpioDeviceConfigTable)/sizeof(UINT8), 0, 0, 0, 0x0000000F},
  {
    //                          DeviceId                    Enable              Assert    Deassert  Hotplug
    GPIO_DEVICE_DEFINITION( DEVICE_ID_ODD,              CPM_DEVICE_AUTO,    0,        0,        0  ),
    GPIO_DEVICE_DEFINITION( DEVICE_ID_DMC,              CPM_DEVICE_ON,      0,        0,        0  ),
    GPIO_DEVICE_DEFINITION( DEVICE_ID_MPCIE1,           CPM_DEVICE_ON,      0,        0,        0  ),
    GPIO_DEVICE_DEFINITION( DEVICE_ID_MPCIE2,           CPM_DEVICE_ON,      0,        0,        0  ),
    GPIO_DEVICE_DEFINITION( DEVICE_ID_MXM,              CPM_DEVICE_AUTO,    1,        0,        0  ),
    GPIO_DEVICE_DEFINITION( DEVICE_ID_BT,               CPM_DEVICE_ON,      0,        0,        0  ),
    GPIO_DEVICE_DEFINITION( DEVICE_ID_SWINGMODE,        CPM_DEVICE_ON,      0,        0,        0  ),
    GPIO_DEVICE_DEFINITION( DEVICE_ID_POWERLEVEL,       CPM_DEVICE_ON,      0,        0,        0  ),
    GPIO_DEVICE_DEFINITION( DEVICE_ID_VGAMUXSEL,        CPM_DEVICE_ON,      0,        0,        0  ),
    0xFF,
  }
};

//
// Device Detection Table
//
AMD_CPM_GPIO_DEVICE_DETECTION_TABLE gCpmGpioDeviceDetectionTable = {
  {CPM_SIGNATURE_GPIO_DEVICE_DETECTION, sizeof(gCpmGpioDeviceDetectionTable)/sizeof(UINT8), 0, 0, 0, 0x0000000F},
  {
    {  DEVICE_ID_ODD,           CPM_TYPE_GPIO_1,   112,   0,    0,    0  },      // ODD_PLUGIN#: GEVENT16#
```

```
    { DEVICE_ID_DMC,              CPM_TYPE_GPIO_1,  172,  0,   0,   0 },     // DMC_PRESENT#: GPIO172
    { DEVICE_ID_MXM,              CPM_TYPE_GPIO_2,  32,   0,   34,  0 },     // MXM_PRESENT1: GPIO32. MXM_PRESENT2: GPIO34
    { DEVICE_ID_EXPRESSCARD,      CPM_TYPE_GPIO_1,  101,  0,   0,   0 },     // PCIE_EXPCARD_PWREN#: GEVENT5
    0xFF,
  }
};

//
// Device Reset Table
//
AMD_CPM_GPIO_DEVICE_RESET_TABLE gCpmGpioDeviceResetTable = {
  {CPM_SIGNATURE_GPIO_DEVICE_RESET, sizeof(gCpmGpioDeviceDetectionTable)/sizeof(UINT8), 0, 0, 0, 0x0000000F},
  {
    //   DeviceId                  Mode                  Type                 Num Value          InitFlag;
    { DEVICE_ID_DMC,              CPM_RESET_ASSERT,     CPM_RESET_GPIO,      CPM_GPIO_PIN(2, 0),      0 },    // DMC_RST0#
    { DEVICE_ID_MPCIE1,           CPM_RESET_ASSERT,     CPM_RESET_GPIO,      CPM_GPIO_PIN(0, 0),      0 },    // MPCIE_RST1#
    { DEVICE_ID_MPCIE2,           CPM_RESET_ASSERT,     CPM_RESET_GPIO,      CPM_GPIO_PIN(1, 0),      0 },    // MPCIE_RST2#
    { DEVICE_ID_MXM,              CPM_RESET_ASSERT,     CPM_RESET_GPIO,      CPM_GPIO_PIN(191, 0),    0 },    // PE_GPIO0#
    { DEVICE_ID_DT,               CPM_RESET_ASSERT,     CPM_RESET_GPIO,      CPM_GPIO_PIN(4, 0),      0 },    // MPCIE_RST_DT#
    { DEVICE_ID_LAN,              CPM_RESET_ASSERT,     CPM_RESET_GPIO,      CPM_GPIO_PIN(25, 0),     0 },    // PCIE_RST_LAN#
    { DEVICE_ID_DDI3,             CPM_RESET_ASSERT,     CPM_RESET_GPIO,      CPM_GPIO_PIN(27, 0),     0 },    // DDI3_PCIE_RST#
    { DEVICE_ID_EXPRESSCARD,      CPM_RESET_ASSERT,     CPM_RESET_GPIO,      CPM_GPIO_PIN(53, 0),     0 },    // MPCIE_RST_XPRESS#

    { DEVICE_ID_DMC,              CPM_RESET_DEASSERT,   CPM_RESET_GPIO,      CPM_GPIO_PIN(2, 1),      0 },    // DMC_RST0#
    { DEVICE_ID_MPCIE1,           CPM_RESET_DEASSERT,   CPM_RESET_GPIO,      CPM_GPIO_PIN(0, 1),      0 },    // MPCIE_RST1#
    { DEVICE_ID_MPCIE2,           CPM_RESET_DEASSERT,   CPM_RESET_GPIO,      CPM_GPIO_PIN(1, 1),      0 },    // MPCIE_RST2#
    { DEVICE_ID_MXM,              CPM_RESET_DEASSERT,   CPM_RESET_GPIO,      CPM_GPIO_PIN(191, 1),    0 },    // PE_GPIO0#
    { DEVICE_ID_DT,               CPM_RESET_DEASSERT,   CPM_RESET_GPIO,      CPM_GPIO_PIN(4, 1),      0 },    // MPCIE_RST_DT#
    { DEVICE_ID_LAN,              CPM_RESET_DEASSERT,   CPM_RESET_GPIO,      CPM_GPIO_PIN(25, 1),     0 },    // PCIE_RST_LAN#
    { DEVICE_ID_DDI3,             CPM_RESET_DEASSERT,   CPM_RESET_GPIO,      CPM_GPIO_PIN(27, 1),     0 },    // DDI3_PCIE_RST#
    { DEVICE_ID_EXPRESSCARD,      CPM_RESET_DEASSERT,   CPM_RESET_GPIO,      CPM_GPIO_PIN(53, 1),     0 },    // MPCIE_RST_XPRESS#

    0xFF,
  }
};

//
// GPIO Device Power Table
//
AMD_CPM_GPIO_DEVICE_POWER_TABLE gCpmGpioDevicePowerTable = {
  {CPM_SIGNATURE_GPIO_DEVICE_POWER, sizeof(gCpmGpioDevicePowerTable)/sizeof(UINT8), 0, 0, 0, 0x0000000F},
  {
    //  DeviceId                  Mode                  Type                  Config             InitFlag;
    { DEVICE_ID_ODD,              CPM_POWER_OFF,        CPM_POWER_SET,        CPM_GPIO_PIN(171, 0),    0 },    // ODD_PWR
    { DEVICE_ID_DMC,              CPM_POWER_OFF,        CPM_POWER_SET,        CPM_GPIO_PIN(175, 1),    0 },    // DMC_PD
    { DEVICE_ID_MXM,              CPM_POWER_OFF,        CPM_POWER_SET,        CPM_GPIO_PIN(55, 0),     0 },    // MXM_PWR_EN
    { DEVICE_ID_MXM,              CPM_POWER_OFF,        CPM_POWER_DELAY,      10000,                   0 },    // MXM Delay 3ms
    { DEVICE_ID_MXM,              CPM_POWER_OFF,        CPM_POWER_SET,        CPM_GPIO_PIN(192, 0),    0 },    // PE_GPIO1
    { DEVICE_ID_SWINGMODE,        CPM_POWER_OFF,        CPM_POWER_SET,        CPM_GPIO_PIN(8, 0),      0 },    // PEX_STD_SW#: Standard
    { DEVICE_ID_POWERLEVEL,       CPM_POWER_OFF,        CPM_POWER_SET,        CPM_GPIO_PIN(22, 0),     0 },    // FCH_PWR_LV: Battery
    { DEVICE_ID_VGAMUXSEL,        CPM_POWER_OFF,        CPM_POWER_SET,        CPM_GPIO_PIN(24, 1),     0 },    // VGA_MUX_SEL: MXM
    { DEVICE_ID_BT,               CPM_POWER_OFF,        CPM_POWER_SET,        CPM_GPIO_PIN(7, 0),      3 },    // BT_ON
    { DEVICE_ID_RADIO,            CPM_POWER_OFF,        CPM_POWER_SET,        CPM_GPIO_PIN(12, 0),     3 },    // WL_DISABLE#
    { DEVICE_ID_RADIO,            CPM_POWER_OFF,        CPM_POWER_SET,        CPM_GPIO_PIN(13, 0),     3 },    // WU_DISABLE#
    { DEVICE_ID_RADIO,            CPM_POWER_OFF,        CPM_POWER_SET,        CPM_GPIO_PIN(199, 0),    3 },    // WP_DISABLE#
    { DEVICE_ID_WIRELESS,         CPM_POWER_OFF,        CPM_POWER_SET,        CPM_GPIO_PIN(176, 1),    3 },    // MPCIE_PD1
    { DEVICE_ID_WIRELESS,         CPM_POWER_OFF,        CPM_POWER_SET,        CPM_GPIO_PIN(177, 1),    3 },    // MPCIE_PD2

    { DEVICE_ID_ODD,              CPM_POWER_ON,         CPM_POWER_SET,        CPM_GPIO_PIN(171, 1),    0 },    // ODD_PWR
    { DEVICE_ID_DMC,              CPM_POWER_ON,         CPM_POWER_SET,        CPM_GPIO_PIN(175, 0),    0 },    // DMC_PD
    { DEVICE_ID_MXM,              CPM_POWER_ON,         CPM_POWER_SET,        CPM_GPIO_PIN(192, 1),    0 },    // PE_GPIO1
    { DEVICE_ID_MXM,              CPM_POWER_ON,         CPM_POWER_DELAY,      10000,                   0 },    // MXM Delay 3ms
    { DEVICE_ID_MXM,              CPM_POWER_ON,         CPM_POWER_SET,        CPM_GPIO_PIN(55, 1),     0 },    // MXM_PWR_EN
    { DEVICE_ID_MXM,              CPM_POWER_ON,         CPM_POWER_WAIT,       CPM_GPIO_PIN(51, 1),     0 },    // MXM_PWRGD
    { DEVICE_ID_SWINGMODE,        CPM_POWER_ON,         CPM_POWER_SET,        CPM_GPIO_PIN(8, 1),      0 },    // PEX_STD_SW#: Half
```

```
    { DEVICE_ID_POWERLEVEL, CPM_POWER_ON,  CPM_POWER_SET,  CPM_GPIO_PIN(22, 1),  0 },  // FCH_PWR_LV: AC
    { DEVICE_ID_VGAMUXSEL,  CPM_POWER_ON,  CPM_POWER_SET,  CPM_GPIO_PIN(24, 0),  0 },  // VGA_MUX_SEL: FCH
    { DEVICE_ID_BT,         CPM_POWER_ON,  CPM_POWER_SET,  CPM_GPIO_PIN(7, 1),   3 },  // BT_ON
    { DEVICE_ID_RADIO,      CPM_POWER_ON,  CPM_POWER_SET,  CPM_GPIO_PIN(12, 1),  3 },  // WL_DISABLE#
    { DEVICE_ID_RADIO,      CPM_POWER_ON,  CPM_POWER_SET,  CPM_GPIO_PIN(13, 1),  3 },  // WU_DISABLE#
    { DEVICE_ID_RADIO,      CPM_POWER_ON,  CPM_POWER_SET,  CPM_GPIO_PIN(199, 1), 3 },  // WP_DISABLE#
    { DEVICE_ID_WIRELESS,   CPM_POWER_ON,  CPM_POWER_SET,  CPM_GPIO_PIN(176, 0), 3 },  // MPCIE_PD1
    { DEVICE_ID_WIRELESS,   CPM_POWER_ON,  CPM_POWER_SET,  CPM_GPIO_PIN(177, 0), 3 },  // MPCIE_PD2

    0xFF,
  }
};

//
// PCIE Clock Table
//
AMD_CPM_PCIE_CLOCK_TABLE  gCpmPcieClockTable = {
  {CPM_SIGNATURE_PCIE_CLOCK, sizeof(gCpmPcieClockTable)/sizeof(UINT8), 0, 0, 0, 0x0000000F},
  {
    //  ClkId       ClkReq          ClkIdExt    ClkReqExt     DeviceId                Device  Function  SlotCheck  SpecialFunctionId;
    { GPP_CLK0,  CLK_REQ0,     SRC_CLK0,  CLK_REQ0,     DEVICE_ID_EXPRESSCARD, 7,   0, NON_SLOT_CHECK,  0 },  // EXPRESS CARD
    { GPP_CLK1,  CLK_REQ1,     SRC_CLK4,  CLK_ENABLE,   DEVICE_ID_DDI3,        0,   0, NON_SLOT_CHECK,  0 },  // DDI SLOT3
    { GPP_CLK2,  CLK_REQ2,     SRC_CLK5,  CLK_REQ5,     DEVICE_ID_DMC,         21,  1, SLOT_CHECK,      0 },  // DMC
    { GPP_CLK3,  CLK_REQ3,     SRC_CLK1,  CLK_REQ1,     DEVICE_ID_LAN,         4,   0, SLOT_CHECK,      0 },  // LAN
    { GPP_CLK4,  CLK_REQ4,     SRC_CLK2,  CLK_REQ2,     DEVICE_ID_MPCIE1,      5,   0, SLOT_CHECK,      0 },  // Mini PCIE1
    { GPP_CLK5,  CLK_ENABLE,   SRC_CLK6,  CLK_REQ6,     DEVICE_ID_DT,          21,  3, SLOT_CHECK,      0 },  // DT X1 PCIE
    { GPP_CLK6,  CLK_DISABLE,  SRC_CLK7,  CLK_DISABLE,  0xFF,                  0,   0, NON_SLOT_CHECK,  0 },  // N/A
    { GPP_CLK7,  CLK_DISABLE,  SRC_CLK9,  CLK_ENABLE,   0xFF,                  0,   0, NON_SLOT_CHECK,  0 },  // N/A
    { GPP_CLK8,  CLK_REQ8,     SRC_CLK3,  CLK_REQ3,     DEVICE_ID_MPCIE2,      6,   0, SLOT_CHECK,      0 },  // Mini PCIE2
    { GPP_CLK9,  CLK_REQGFX,   SRC_CLK8,  CLK_REQ8,     DEVICE_ID_MXM,         2,   0, SLOT_CHECK,      0 },  // MXM
    0xFF,
  }
};

//
// External ClkGen Table
//
AMD_CPM_EXT_CLKGEN_TABLE  gCpmExtClkGenTable = {
  {CPM_SIGNATURE_EXT_CLKGEN, sizeof(gCpmExtClkGenTable)/sizeof(UINT8), 0, 0, 0, 0x0000000F},
  0,                                    // Smbus Select: 0: Smbus0. 1: Smbus1
  0x69,                                 // Smbus Address: 0xD2
  {
  //  Function   Offset    AndMask    OrMask
    { 0x80,    0x02,    0xFE,    0x00 },  // Clk0 Disable
    { 0x81,    0x02,    0xFD,    0x00 },  // Clk1 Disable
    { 0x82,    0x02,    0xFB,    0x00 },  // Clk2 Disable
    { 0x83,    0x02,    0xF7,    0x00 },  // Clk3 Disable
    { 0x84,    0x02,    0xFE,    0x00 },  // Clk4 Disable
    { 0x85,    0x02,    0xEF,    0x00 },  // Clk5 Disable
    { 0x86,    0x02,    0xDF,    0x00 },  // Clk6 Disable
    { 0x87,    0x02,    0xBF,    0x00 },  // Clk7 Disable
    { 0x88,    0x02,    0x7F,    0x00 },  // Clk8 Disable
    { 0x89,    0x01,    0xFE,    0x00 },  // Clk9 Disable
    { 0x8A,    0x01,    0xFD,    0x00 },  // Clk10 Disable
    { 0x8B,    0x01,    0xFB,    0x00 },  // Clk11 Disable

    { 0x90,    0x04,    0xBF,    0x40 },  // ClkREQ0 Enable
    { 0x91,    0x04,    0x7F,    0x80 },  // ClkREQ1 Enable
    { 0x92,    0x03,    0xBF,    0x40 },  // ClkREQ2 Enable
    { 0x93,    0x03,    0x7F,    0x80 },  // ClkREQ3 Enable
    { 0x94,    0x01,    0xEF,    0x10 },  // ClkREQ4 Enable
    { 0x95,    0x01,    0xDF,    0x20 },  // ClkREQ5 Enable
    { 0x96,    0x01,    0xBF,    0x40 },  // ClkREQ6 Enable
    { 0x97,    0x01,    0x7F,    0x80 },  // ClkREQ7 Enable
    { 0x98,    0x0B,    0xFE,    0x01 },  // ClkREQ8 Enable
```

```
    { 0x99,    0x0B,    0xFD,    0x02  },    // ClkREQ9 Enable
    { 0x9A,    0x0B,    0xFB,    0x04  },    // ClkREQ10 Enable
    { 0x9B,    0x0B,    0xF7,    0x08  },    // ClkREQ11 Enable

    0xFF,
  }
};


//
// PCIe Topology Table
//
AMD_CPM_PCIE_TOPOLOGY_TABLE gCpmPcieTopologyTable = {
  {CPM_SIGNATURE_PCIE_TOPOLOGY, sizeof(gCpmPcieTopologyTable)/sizeof(UINT8), 0, 0, 0, 0x0000000F},          // Header
  0,                                                                  // SocketId
  {                                                                   // PCIe_PORT_DESCRIPTOR
    {                                                                 // Lanes 8:23, PCI Device Number 2
      0,
      PCIE_ENGINE_DATA_INITIALIZER (PciePortEngine, 8, 23),
      PCIE_PORT_DATA_INITIALIZER (PortEnabled, ChannelTypeExt6db, 2, HotplugDisabled, PcieGenMaxSupported, PcieGenMaxSupported,
AspmL0sL1, DEVICE_ID_MXM)
    },

    {                                                                 // Lanes 16:19, PCI Device Number 3
      0,
      PCIE_ENGINE_DATA_INITIALIZER (PcieUnusedEngine, 16, 19),
      PCIE_PORT_DATA_INITIALIZER (PortDisabled, ChannelTypeExt6db, 3, HotplugDisabled, PcieGenMaxSupported, PcieGenMaxSupported,
AspmL0sL1, DEVICE_ID_MXM)
    },

    {                                                                 // Lanes 4, PCI Device Number 4
      0,
      PCIE_ENGINE_DATA_INITIALIZER (PciePortEngine, 4, 4),
      PCIE_PORT_DATA_INITIALIZER (PortEnabled, ChannelTypeExt6db, 4, HotplugDisabled, PcieGenMaxSupported, PcieGenMaxSupported,
AspmL0sL1, DEVICE_ID_LAN)
    },
    {                                                                 // Lanes 5, PCI Device Number 5
    0,
      PCIE_ENGINE_DATA_INITIALIZER (PciePortEngine, 5, 5),
      PCIE_PORT_DATA_INITIALIZER (PortEnabled, ChannelTypeExt6db, 5, HotplugDisabled, PcieGenMaxSupported, PcieGenMaxSupported,
AspmL0sL1, DEVICE_ID_MPCIE1)
    },
    {                                                                 // Lanes 6, PCI Device Number 6
      0,
      PCIE_ENGINE_DATA_INITIALIZER (PciePortEngine, 6, 6),
      PCIE_PORT_DATA_INITIALIZER (PortEnabled, ChannelTypeExt6db, 6, HotplugDisabled, PcieGenMaxSupported, PcieGenMaxSupported,
AspmL0sL1, DEVICE_ID_MPCIE2)
    },
    {                                                                 // Lanes 7, PCI Device Number 7
      DESCRIPTOR_TERMINATE_LIST,
      PCIE_ENGINE_DATA_INITIALIZER (PciePortEngine, 7, 7),
      PCIE_PORT_DATA_INITIALIZER (PortEnabled, ChannelTypeExt6db, 7, HotplugDisabled, PcieGenMaxSupported, PcieGenMaxSupported,
AspmL0sL1, DEVICE_ID_EXPRESSCARD)
    },
  },
  {                                                                   // PCIe_DDI_DESCRIPTOR
    {                                                                 // Port 0. Mini DDI slot
      0,
      PCIE_ENGINE_DATA_INITIALIZER (PcieDdiEngine, 24, 27),
      PCIE_DDI_DATA_INITIALIZER (ConnectorTypeDP, Aux1, Hdp1)
    },
    {                                                                 // Port 1, DMC slot
      0,
      PCIE_ENGINE_DATA_INITIALIZER (PcieDdiEngine, 28, 31),
      PCIE_DDI_DATA_INITIALIZER (ConnectorTypeNutmegDpToVga, Aux2, Hdp2)
```

```
    },
    {                                                   // Port 2, Mini DDI slot
      0,
      PCIE_ENGINE_DATA_INITIALIZER (PcieDdiEngine, 32, 35),
      PCIE_DDI_DATA_INITIALIZER (ConnectorTypeDP, Aux3, Hdp3)
    },
    {                                                   // Via MXM slot, Lane[8,11] unused for DDI
      0,
      PCIE_ENGINE_DATA_INITIALIZER (PcieUnusedEngine, 12, 15),
      PCIE_DDI_DATA_INITIALIZER (ConnectorTypeDP, Aux4, Hdp4)
    },
    {
      0,
      PCIE_ENGINE_DATA_INITIALIZER (PcieUnusedEngine, 16, 19),
      PCIE_DDI_DATA_INITIALIZER (ConnectorTypeDP, Aux5, Hdp5)
    },
    {
      DESCRIPTOR_TERMINATE_LIST,
      PCIE_ENGINE_DATA_INITIALIZER (PcieUnusedEngine, 20, 23),
      PCIE_DDI_DATA_INITIALIZER (ConnectorTypeDP, Aux6, Hdp6)
    }
  },
};

//
// CPM PCIe Topology Override Table
//
AMD_CPM_PCIE_TOPOLOGY_OVERRIDE_TABLE   gCpmPcieTopologyOverride = {
  {CPM_SIGNATURE_PCIE_TOPOLOGY_OVERRIDE, sizeof(gCpmPcieTopologyOverride)/sizeof(UINT8), 0, 0, 0, 0x0000000F},
  {
    0xFF,
  },
};

//
// CPM Express Card Table
//
AMD_CPM_EXPRESS_CARD_TABLE       gCpmExpressCardTable ={
  {CPM_SIGNATURE_PCIE_EXPRESS_CARD, sizeof(gCpmExpressCardTable)/sizeof(UINT8), 0, 0, 0, 0x0000000F},
  7,          // Device Number of PCIE Bridge
  0,          // Function Number of PCIE Bridge
  5,          // GEVENT Pin 5
};

//
// CPM Wireless Button Table
//
AMD_CPM_WIRELESS_BUTTON_TABLE     gCpmWirelessButtonTable = {
 {CPM_SIGNATURE_WIRELESS_BUTTON, sizeof (gCpmWirelessButtonTable) / sizeof (UINT8), 0, 0, 0x01, 0x0000000F},
 {
  {3, 2},
  {3, 3},
 },
 12,                              // GEVENT Pin 12
 DEVICE_ID_RADIO,                 // Device Id ro control radio
 DEVICE_ID_WIRELESS,              // Device Id to control power
 DEVICE_ID_BT                     // Device Id to control BT
};

//
// Thermal Fan Control Table
//
AMD_CPM_ACPI_THERMAL_FAN_TABLE     gCpmAcpiThermalFanTable = {
  {CPM_SIGNATURE_ACPI_THERMAL_FAN, sizeof(gCpmAcpiThermalFan)/sizeof(UINT8), 0, 0, 0, 0x0000000F},
  {
```

```
    0x0E,                 // EventPin: GEVENT14
    0x00,                 // SbFanCtrlId: FANOUT0
  },
  {
    105,                  // CpuCRT
    98,                   // CpuPSV
    50,                   // CpuAC0
    80,                   // CpuAC1
    0,                    // CpuAC2
    0,                    // CpuAC3
    40,                   // CpuAL0
    100,                  // CpuAL1
    0,                    // CpuAL2
    0,                    // CpuAL3
    0,                    // ThermalSensor
    4,                    // HysteresisInfo
    4,                    // HysteresisInfoPsv
  },
};

//
// CPM Main Table
//
AMD_CPM_MAIN_TABLE gCpmMainTable = {
  {CPM_SIGNATURE_MAIN_TABLE, sizeof(gCpmMainTable)/sizeof(UINT8), 0, 0, 0, 0x0000000F},
  "RathmoreTV",           // PlatformName:          RathmoreTV
  0x02,                   // BiosType:              Internal BIOS
  0,                      // CurrentPlatformId:     0
  0xF8000000,             // PcieMemIoBaseAddr      0xF8000000
  0xFED80000,             // AcpiMemIoBaseAddr      0xFED80000
  NULL,                   // Reserved for Internal Used
  NULL,                   // Reserved for Internal Used
  NULL,                   // Reserved for Internal Used
  NULL,                   // Reserved for Internal Used
  NULL,                   // Reserved for Internal Used
  0x400,                  // DisplayFeature:        Disable
  0,                      // ZeroPowerOddEn:        Disable
  0,                      // AcpiThermalFanEn:      Disable
  0,                      // ExtClkGen              Config Type 0
  0,                      // UnusedGppClkOffEn:     Disable
  0,                      // AdaptiveS4En           Disable
  0,                      // WirelessButtonEn       Disable
  0,                      // Ec:                    Disable
  0,                      // Reserved
  0,                      // Reserved
  0,                      // Reserved
  0                       // Reserved
};

//
// CPM Display Feature Module
//

//
// CPM Device Path Table
//
AMD_CPM_DEVICE_PATH_TABLE gCpmDevicePathTable = {
  {CPM_SIGNATURE_DEVICE_PATH, sizeof(gCpmDevicePathTable)/sizeof(UINT8), 0, 0, 0, 0x0000000F},
  {
    {0x80000001, 0x00, {0, 0}, {1, 0}},       // PowerXpress, iGPU, (0,0,0)/(1,0)
    {0x80000001, 0x01, {2, 0}, {0, 0}},       // PowerXpress, dGPU, (0,2,0)/(0,0)
    {0x80000001, 0x01, {3, 0}, {0, 0}},       // PowerXpress, dGPU, (0,3,0)/(0,0)
    {0x00000000, 0x00, {0, 0}, {0, 0}},
  }
};
```

```
//
// CPM Specific Ssid Table
//
AMD_CPM_SPECIFIC_SSID_TABLE gCpmSpecificSsidTable = {
  {CPM_SIGNATURE_SPECIFIC_SSID, sizeof(gCpmSpecificSsidTable)/sizeof(UINT8), 0, 0, 0, 0x00000006},
  {
    {  0x1002,   0x95C4   },
    {  0x1002,   0x9553   },
    {  0x1002,   0x68B0   },
    {  0x1002,   0x9480   },
    {  0x1002,   0x68E0   },    // MXM
    {  0x1002,   0x68E5   },    // MXM-Robson-Cedar    6300M
    {  0x1002,   0x6760   },    // MXM-Caisos-Seymour 6470M
    {  0x1002,   0x6741   },    // MXM-Turks-Whistler   6600M
    {  0x1002,   0x6742   },    // MXM-Turks-Whistler   6600M
    {  0x1002,   0x6840   },
    {  0x1002,   0x6841   },
    {  0x1002,   0x6842   },

    {  0x1002,   0x9900   },
    {  0x1002,   0x9903   },
    {  0x1002,   0x9904   },
    {  0x1002,   0x990F   },
    {  0x1002,   0x9990   },
    {  0x1002,   0x9991   },
    {  0x1002,   0x9992   },
    {  0x1002,   0x9993   },
    {  0xFFFF,   0xFFFF   },    //End of Table
  }
};

//
// Display Feature Table
//
AMD_CPM_DISPLAY_FEATURE_TABLE gCpmDisplayFeatureTable = {
  {CPM_SIGNATURE_DISPLAY_FEATURE, sizeof(gCpmDisplayFeatureTable)/sizeof(UINT8), 0, 0, 0, 0x0000000F},
  0,                            // FunctionDisableMask
  DEVICE_ID_MXM,                // MXM Device Id
  0,                            // Docking Device Id
  0,                             // MuxFlag
  0,                            // No Display Mux
  0,                            // No I2c Mux
  10,                           // AtpxConnector8Number
  {                             // AtpxConnector8
    {0x05, 0x00, 0x00, 0x0110},         // Connector #0: LCD1 on iGPU
    {0x05, 0x01, 0x00, 0x0100},         // Connector #1: CRT1 on iGPU
    {0x07, 0x03, 0x00, 0x0210},         // Connector #2: DFP1 on iGPU
    {0x07, 0x07, 0x00, 0x0220},         // Connector #3: DFP2 on iGPU
    {0x00, 0x09, 0x00, 0x0230},         // Connector #4: DFP3 on iGPU
    {0x01, 0x00, 0x01, 0x0110},         // Connector #5: LCD1 on dGPU
    {0x01, 0x01, 0x01, 0x0100},         // Connector #6: CRT1 on dGPU
    {0x03, 0x03, 0x01, 0x0210},         // Connector #7: DFP1 on dGPU
    {0x03, 0x07, 0x01, 0x0220},         // Connector #8: DFP2 on dGPU
    {0x00, 0x09, 0x01, 0x0230},         // Connector #9: DFP3 on dGPU
  },
  0,                            // AtpxConnector9Number
  {                             // AtpxConnector9
    {0x00, 0x00, 0x00},
    {0x00, 0x00, 0x00},
    {0x00, 0x00, 0x00},
  },
  0x51,                         // AtifSupportedNotificationMask;
  7,                            // AtifDeviceCombinationNumber;
  {                             // AtifDeviceCombinationBuffer[20];
```

```
    0x01, 0x02, 0x08, 0x80, 0x03, 0x09, 0x81
  },
  {
    0x6C, 0x00,                 // WORD Structure Size : 0x6A
    0x00, 0x00,                 // WORD Flags          : Reserved
    0x00,                     // BYTE       Error Code     : 0x00
    0x64,                     // BYTE       AC Level       : 100%
    0x20,                     // BYTE       DC Level       : 32%
    0x0C,                     // BYTE       Minimum signal : 12
    0xFF,                     // BYTE       Maximum signal : 255
    0x31,                     // BYTE       Count          : 49
    0x02, 0x0E,                 // BYTE-BYTE    First Lumi/Signal: 2% - 14
    0x04, 0x10,                 // BYTE-BYTE
    0x06, 0x12,                 // BYTE-BYTE
    0x08, 0x15,                 // BYTE-BYTE
    0x0A, 0x17,                 // BYTE-BYTE
    0x0C, 0x1A,                 // BYTE-BYTE
    0x0E, 0x1D,                 // BYTE-BYTE
    0x10, 0x20,                 // BYTE-BYTE
    0x12, 0x23,                 // BYTE-BYTE
    0x14, 0x26,                 // BYTE-BYTE
    0x16, 0x29,                 // BYTE-BYTE
    0x18, 0x2C,                 // BYTE-BYTE
    0x1A, 0x30,                 // BYTE-BYTE
    0x1C, 0x34,                 // BYTE-BYTE
    0x1E, 0x37,                 // BYTE-BYTE
    0x20, 0x3B,                 // BYTE-BYTE
    0x22, 0x3E,                 // BYTE-BYTE
    0x24, 0x43,                 // BYTE-BYTE
    0x26, 0x47,                 // BYTE-BYTE
    0x28, 0x4B,                 // BYTE-BYTE
    0x2A, 0x50,                 // BYTE-BYTE
    0x2C, 0x54,                 // BYTE-BYTE
    0x2E, 0x58,                 // BYTE-BYTE
    0x30, 0x5D,                 // BYTE-BYTE
    0x32, 0x62,                 // BYTE-BYTE
    0x34, 0x67,                 // BYTE-BYTE
    0x36, 0x6C,                 // BYTE-BYTE
    0x38, 0x71,                 // BYTE-BYTE
    0x3A, 0x76,                 // BYTE-BYTE
    0x3C, 0x7B,                 // BYTE-BYTE
    0x3E, 0x81,                 // BYTE-BYTE
    0x40, 0x87,                 // BYTE-BYTE
    0x42, 0x8C,                 // BYTE-BYTE
    0x44, 0x92,                 // BYTE-BYTE
    0x46, 0x98,                 // BYTE-BYTE
    0x48, 0x9E,                 // BYTE-BYTE
    0x4A, 0xA4,                 // BYTE-BYTE
    0x4C, 0xAB,                 // BYTE-BYTE
    0x4E, 0xB1,                 // BYTE-BYTE
    0x50, 0xB7,                 // BYTE-BYTE
    0x52, 0xBE,                 // BYTE-BYTE
    0x54, 0xC5,                 // BYTE-BYTE
    0x56, 0xCC,                 // BYTE-BYTE
    0x58, 0xD3,                 // BYTE-BYTE
    0x5A, 0xDA,                 // BYTE-BYTE
    0x5C, 0xE1,                 // BYTE-BYTE
    0x5E, 0xE8,                 // BYTE-BYTE
    0x60, 0xF0,                 // BYTE-BYTE
    0x62, 0xF7                  // BYTE-BYTE   Last Lumi/Signal: 98% - 250
  }
};

//
// CPM Zero Power Odd Table
```

```
//
AMD_CPM_ZERO_POWER_ODD_TABLE   gCpmZeroPowerOddTable = {
  {CPM_SIGNATURE_ZERO_POWER_ODD, sizeof(gCpmZeroPowerOddTable)/sizeof(UINT8), 0, 0, 0, 0x0000000F},
  DEVICE_ID_ODD,                // Gpio pin
  0x06,                         // Gevent pin for FCH_ODD_DA
  0x10,                         // Gevent pin for ODD_PLUGIN# Detect
  0x1F,                         // Dummy Event
  0x0E,                         // SATA Mode Mask
  0x01,                         // SATA Port Number
};

//
// Adaptive S4 Table
//
AMD_CPM_ADAPTIVE_S4_TABLE       gCpmAdaptiveS4Table = {
 {CPM_SIGNATURE_ADAPTIVE_S4, sizeof (gCpmAdaptiveS4Table) / sizeof (UINT8), 0, 0, 0, 0x0F},
 0x05,                          // BufferType
 0x40,                          // BufferOffset
 0x0C,                           // BufferSize
 0xE0                           // EcRamOffset
};

//
// Save Context Table
//
AMD_CPM_SAVE_CONTEXT_TABLE      gCpmSaveContextTable = {
 {CPM_SIGNATURE_SAVE_CONTEXT, sizeof (gCpmSaveContextTable) / sizeof (UINT8), 0, 0, 0, 0x01},
 0x05,          // BufferType
 0x50,          // BufferOffset
 0x10,          // BufferSize
};


void *gCpmTableList[] = {
  &gCpmMainTable,

  &gCpmPlatformIdTable,
  &gCpmPlatformIdConvertTable,
  &gCpmPreInitTable,
  &gCpmSaveContextTable,

  &gCpmGpioInitTable,
  &gCpmGeventInitTable,
  &gCpmGpioDeviceConfigTable,
  &gCpmGpioDevicePowerTable,
  &gCpmGpioDeviceDetectionTable,
  &gCpmGpioDeviceResetTable,
  &gCpmPcieClockTable,
  &gCpmSetMemVoltage,
  &gCpmSetVddpVddrVoltage,

  &gCpmPcieTopologyTable,
  &gCpmPcieTopologyOverride,
  &gCpmExpressCardTable,
  &gCpmWirelessButtonTable,

  &gCpmAcpiThermalFanTable,

  &gCpmAdaptiveS4Table,

  &gCpmDisplayFeatureTable,
  &gCpmDevicePathTable,
  &gCpmZeroPowerOddTable,
  &gCpmSpecificSsidTable,
```

```
  NULL
};
```

6.3 AmdCpmOemInitPeim Driver Sample

```
#include "Tiano.h"
#include "Pei.h"
#include "PeiLib.h"
#include "Variable.h"
#include "SetupConfig.h"
#include "AmdCpmCommon.h"

#include EFI_PPI_CONSUMER (AmdCpmOemTablePpi)
#include EFI_PPI_CONSUMER (AmdCpmTablePpi)
#include EFI_PPI_CONSUMER (Smbus)

EFI_STATUS
EFIAPI
CpmOverrideTableNotifyCallback (
    IN EFI_PEI_SERVICES             **PeiServices,
    IN EFI_PEI_NOTIFY_DESCRIPTOR    *NotifyDescriptor,
    IN VOID                         *Ppi
    );

static EFI_PEI_NOTIFY_DESCRIPTOR mCpmOemTableOverrideNotify = {
    (EFI_PEI_PPI_DESCRIPTOR_NOTIFY_CALLBACK | EFI_PEI_PPI_DESCRIPTOR_TERMINATE_LIST),
    &gAmdCpmTablePpiGuid,
    CpmOverrideTableNotifyCallback
};


EFI_STATUS
EFIAPI
InitializeAmdCpmOemInitPeim (
    IN EFI_FFS_FILE_HEADER      *FfsHeader,
    IN EFI_PEI_SERVICES         **PeiServices
    )
{
    EFI_STATUS              Status = 0;
    AMD_CPM_OEM_TABLE_PPI   *AmdCpmOemTablePpi;
    EFI_PEI_PPI_DESCRIPTOR  *PpiListCpmOemTable;

    Status = (*PeiServices)->AllocatePool (
                             PeiServices,
                             sizeof (AMD_CPM_OEM_TABLE_PPI),
                             &AmdCpmOemTablePpi
                             );
  if (EFI_ERROR (Status)) {
    return Status;
  }

  AmdCpmOemTablePpi->Revision = AMD_CPM_OEM_TABLE_PPI_REV;
  AmdCpmOemTablePpi->PlatformId = 0xFF;
  AmdCpmOemTablePpi->TableList = &gCpmTableList[0];

  Status = (*PeiServices)->AllocatePool (
                             PeiServices,
                             sizeof (EFI_PEI_PPI_DESCRIPTOR),
                             &PpiListCpmOemTable
                             );
    if (EFI_ERROR (Status)) {
        return Status;
    }
```

```
     PpiListCpmOemTable->Flags = (EFI_PEI_PPI_DESCRIPTOR_PPI |
                                  EFI_PEI_PPI_DESCRIPTOR_TERMINATE_LIST);
    PpiListCpmOemTable->Guid = &gAmdCpmOemTablePpiGuid;
    PpiListCpmOemTable->Ppi = AmdCpmOemTablePpi;
    Status = (*PeiServices)->InstallPpi (
                               PeiServices,
                               PpiListCpmOemTable
                               );

    if (EFI_ERROR (Status)) {
        return Status;
    }

    Status = (**PeiServices).NotifyPpi (PeiServices, &mCpmOemTableOverrideNotify);

    return EFI_SUCCESS;
}


/*----------------------------------------------------------------------------------------*/
/**
 * CPM Override Function After AMD CPM Table PPI
 *
 * This function updates CPM OEM Tables according to setup options or the value to be detected
 * on run time after AMD CPM Table PPI is installed.
 *
 * @param[in]     PeiServices    Pointer to Pei Services
 *
 * @retval        EFI_SUCCESS    Function initialized successfully
 * @retval        EFI_ERROR      Function failed (see error for more details)
 *
 */
EFI_STATUS
EFIAPI
CpmTableOverride (
  IN       EFI_PEI_SERVICES       **PeiServices
  )
{
  EFI_STATUS                      Status;
  AMD_CPM_TABLE_PPI               *AmdCpmTablePpi;
  AMD_CPM_DISPLAY_FEATURE_TABLE   *DisplayFeatureTablePtr;
  AMD_CPM_MAIN_TABLE              *MainTablePtr;
  AMD_CPM_PCIE_CLOCK_TABLE        *PcieClockTablePtr;
  AMD_CPM_GPIO_DEVICE_CONFIG_TABLE      *GpioDeviceConfigTablePtr;
  AMD_CPM_PCIE_TOPOLOGY_OVERRIDE_TABLE  *PcieTopologyOverrideTablePtr;
  AMD_CPM_PCIE_TOPOLOGY_TABLE     *PcieTopologyTablePtr;
  CPM_OEM_SETUP_OPTION            OemSetupOption;

  Status = (*PeiServices)->LocatePpi (
                           PeiServices,
                           &gAmdCpmTablePpiGuid,
                           0,
                           NULL,
                           &AmdCpmTablePpi
                           );
  if (EFI_ERROR (Status)) {
    return Status;
  }

  MainTablePtr                  = AmdCpmTablePpi->MainTablePtr;
  DisplayFeatureTablePtr        = AmdCpmTablePpi->CommonFunction.GetTablePtr (AmdCpmTablePpi,
CPM_SIGNATURE_DISPLAY_FEATURE);
  PcieClockTablePtr             = AmdCpmTablePpi->CommonFunction.GetTablePtr (AmdCpmTablePpi,
CPM_SIGNATURE_PCIE_CLOCK);
  PcieTopologyTablePtr          = AmdCpmTablePpi->CommonFunction.GetTablePtr (AmdCpmTablePpi,
CPM_SIGNATURE_PCIE_TOPOLOGY);
  GpioDeviceConfigTablePtr      = AmdCpmTablePpi->CommonFunction.GetTablePtr (AmdCpmTablePpi,
CPM_SIGNATURE_GPIO_DEVICE_CONFIG);
```

```
  PcieTopologyOverrideTablePtr  = AmdCpmTablePpi->CommonFunction.GetTablePtr (AmdCpmTablePpi,
CPM_SIGNATURE_PCIE_TOPOLOGY_OVERRIDE);

  Status = CpmOemSetupOption (PeiServices, &OemSetupOption);
  if (EFI_ERROR (Status)) {
    return Status;
  }

  MainTablePtr->DisplayFeature.Raw &= 0xFFFFFC00;

  switch (OemSetupOption.SpecialVgaFeature) {
  case 3: //PX
    if (OemSetupOption.PowerExpressDynamicMode) {
      MainTablePtr->DisplayFeature.Raw |= OemSetupOption.PowerExpressDynamicMode << 1;
    } else {
      MainTablePtr->DisplayFeature.Raw |= BIT0;
    }
    break;
  }

  if ((OemSetupOption.PrimaryVideoAdaptor == 2)) {
    MainTablePtr->DisplayFeature.Raw |= 1 << 8;
  }

  if (OemSetupOption.BrightnessControlMethod == 1) {
    MainTablePtr->DisplayFeature.Raw |= 1 << 9;
  }

  if (OemSetupOption.LoopbackAdaptor == 1) {
    PcieClockTablePtr->Item[9].ClkReq = CLK_ENABLE;
    PcieClockTablePtr->Item[9].ClkReqExt = CLK_ENABLE;
  }

  if (OemSetupOption.DisplayOutput == 0) {
    SetDevice (GpioDeviceConfigTablePtr, DEVICE_ID_VGAMUXSEL, CPM_DEVICE_ON);
  } else {
    SetDevice (GpioDeviceConfigTablePtr, DEVICE_ID_VGAMUXSEL, CPM_DEVICE_OFF);
  }

  if (OemSetupOption.BlueToothEn) {
    SetDevice (GpioDeviceConfigTablePtr, DEVICE_ID_BT, CPM_DEVICE_OFF);
  } else {
    SetDevice (GpioDeviceConfigTablePtr, DEVICE_ID_BT, CPM_DEVICE_ON);
  }

  MainTablePtr->ZeroPowerOddEn = 0;
  if (OemSetupOption.ZeroPowerOddEn) {
    MainTablePtr->ZeroPowerOddEn = BIT0 | BIT1;
  }
  if (OemSetupOption.SystemBootWithPS0 == 0) {
    MainTablePtr->ZeroPowerOddEn |= BIT2;
  }

  MainTablePtr->UnusedGppClkOffEn = OemSetupOption.UnusedGppClkOff;
  MainTablePtr->AcpiThermalFanEn  = OemSetupOption.AcpiThermalFanEn;
  MainTablePtr->AdaptiveS4En      = OemSetupOption.AdaptiveS4En;
  MainTablePtr->WirelessButtonEn  = OemSetupOption.WirelessSwitch + 1;
  MainTablePtr->Ec.Config.S5PlusEn = 1;

  DetectPcieDevices (AmdCpmTablePpi, PcieTopologyOverrideTablePtr);

  return Status;
}
/*----------------------------------------------------------------------------------------*/
/**
 * Update Setup Options
 *
 * This function reads setup options from ReadOnlyVariable and fills in the data
 * structure of CPM OEM Setup Option.
 *
```

```
 * @param[in]     PeiServices    Pointer to Pei Services
 *
 * @retval        EFI_SUCCESS    Function initialized successfully
 * @retval        EFI_ERROR      Function failed (see error for more details)
 *
 */
EFI_STATUS
EFIAPI
CpmOemSetupOption (
  IN       EFI_PEI_SERVICES       **PeiServices,
  IN       CPM_OEM_SETUP_OPTION   *SetupOption
  )
{
  EFI_STATUS                          Status;
  PEI_READ_ONLY_VARIABLE_PPI          *ReadOnlyVariable;
  UINTN                               VariableSize;
  EFI_GUID                            CpmSetupOptionGuid = AMD_CPM_SETUP_GUID;
  AMD_CPM_SETUP_OPTION                CpmSetupOption;

  Status = (*PeiServices)->LocatePpi (
                              PeiServices,
                              &gPeiReadOnlyVariablePpiGuid,
                              0,
                              NULL,
                              &ReadOnlyVariable
                              );
  if (EFI_ERROR (Status)) {
    return Status;
  }

  VariableSize = sizeof (AMD_CPM_SETUP_OPTION);
  Status = ReadOnlyVariable->PeiGetVariable (
                              PeiServices,
                              AMD_CPM_SETUP_VARIABLE_NAME,
                              &CpmSetupOptionGuid,
                              NULL,
                              &VariableSize,
                              &CpmSetupOption
                              );
  if (EFI_ERROR (Status)) {
    return Status;
  }

  SetupOption->SpecialVgaFeature        = CpmSetupOption.AMD_CPM_SETUP_OPTION_SPECIAL_VGA_FEATURE;
  SetupOption->PowerExpressDynamicMode  =
CpmSetupOption.AMD_CPM_SETUP_OPTION_POWER_XPRESS_DYNAMIC_MODE;
  SetupOption->PrimaryVideoAdaptor      =
CpmSetupOption.AMD_CPM_SETUP_OPTION_PRIMARY_VIDEO_ADAPTOR;
  SetupOption->LoopbackAdaptor          = CpmSetupOption.AMD_CPM_SETUP_OPTION_LOOPBACK_ADAPTOR;
  SetupOption->DisplayOutput            = CpmSetupOption.AMD_CPM_SETUP_OPTION_DISPLAY_OUTPUT;
  SetupOption->BrightnessControlMethod  =
CpmSetupOption.AMD_CPM_SETUP_OPTION_BRIGHTNESS_CONTROL_METHOD;
  SetupOption->BlueToothEn              = CpmSetupOption.AMD_CPM_SETUP_OPTION_BLUE_TOOTH_EN;
  SetupOption->ZeroPowerOddEn           = CpmSetupOption.AMD_CPM_SETUP_OPTION_ZERO_POWER_ODD_EN;
  SetupOption->UnusedGppClkOff          =
CpmSetupOption.AMD_CPM_SETUP_OPTION_UNUSED_GPP_CLOCK_OFF;
  SetupOption->SystemBootWithPS0        =
CpmSetupOption.AMD_CPM_SETUP_OPTION_SYSTEM_BOOT_WITH_PS0;
  SetupOption->AcpiThermalFanEn         = CpmSetupOption.AMD_CPM_SETUP_OPTION_ACPI_THERMAL_FAN_EN;
  SetupOption->AdaptiveS4En             = CpmSetupOption.AMD_CPM_SETUP_OPTION_ADAPTIVE_S4_EN;
  SetupOption->WirelessSwitch           = CpmSetupOption.AMD_CPM_SETUP_OPTION_WIRELESS_SWITCH;

  return Status;
}
```

6.4 AGESA wrapper and hook function Sample

```
/*----------------------------------------------------------------------------------------*/
/**
 *  AgesaHookBeforeDramInit
 *
 *  Description:
 *    This is the stub function will call the host environment through the binary block
 *    interface (call-out port) to provide a user hook opportunity
 *
 *  Parameters:
 *    @param[in]      FcnData
 *    @param[in, out] *MemData
 *
 *    @retval         AGESA_STATUS
 *
 **/
/*----------------------------------------------------------------------------------------*/
AGESA_STATUS
AgesaHookBeforeDramInit (
  IN       UINTN           FcnData,
  IN OUT   MEM_DATA_STRUCT *MemData
  )
{
    EFI_PEI_SERVICES                **PeiServices;
    AMD_CPM_TABLE_PPI               *AmdCpmTablePpi;
    AGESA_STATUS                    Status;

    Status      = AGESA_UNSUPPORTED;
    PeiServices = (EFI_PEI_SERVICES **)MemData->StdHeader.ImageBasePtr;

    Status = (*PeiServices)->LocatePpi (
                                        PeiServices,
                                        &gAmdCpmTablePpiGuid,
                                        0,
                                        NULL,
                                        &AmdCpmTablePpi
                                       );
    if (!EFI_ERROR (Status)) {
        AmdCpmTablePpi->PeimPublicFunction.SetMemVoltage(
                                        AmdCpmTablePpi,
                                        MemData->ParameterListPtr->DDR3Voltage
                                       );
        AmdCpmTablePpi->PeimPublicFunction.SetVddpVddrVoltage(
                                        AmdCpmTablePpi,
                                        MemData->ParameterListPtr->VddpVddrVoltage
                                       );
    }

    return Status;
}



/*----------------------------------------------------------------------------------------*/
/**
 * PCIE slot reset control
 *
 *
 *
```

```
 * @param[in]  ResetInfo            Reset information
 * @param[in]  StdHeader            Standard configuration header
 * @retval     AGESA_UNSUPPORTED    This feature is not supported
 */
/*----------------------------------------------------------------------------------------*/

AGESA_STATUS
AgesaPcieSlotResetControl (
       IN      UINTN                   FcnData,
       IN      PCIe_SLOT_RESET_INFO    *ResetInfo
)
{
    AMD_CPM_TABLE_PPI               *AmdCpmTablePpiPtr;
    AGESA_STATUS                    Status;
    EFI_PEI_SERVICES                **PeiServices;

    Status      = AGESA_UNSUPPORTED;

    PeiServices = (EFI_PEI_SERVICES **)ResetInfo->StdHeader.ImageBasePtr;

    Status = (*PeiServices)->LocatePpi (
                                        PeiServices,
                                        &gAmdCpmTablePpiGuid,
                                        0,
                                        NULL,
                                        &AmdCpmTablePpiPtr
                                       );

    if (!EFI_ERROR (Status)) {
        AmdCpmTablePpiPtr->PeimPublicFunction.PcieReset(
                                        AmdCpmTablePpiPtr,
                                        ResetInfo->ResetId,
                                        ResetInfo->ResetControl
                                       );
        Status = AGESA_SUCCESS;
    } else {
        Status = AGESA_UNSUPPORTED;
    }
    return  Status;
}




/*----------------------------------------------------------------------------------------*/
/**
 *  OemCustomizeInitEarly
 *
 *  Description:
 *    This is the stub function will call the host environment through the binary block
 *    interface (call-out port) to provide a user hook opportunity
 *
 *  Parameters:
 *    @param[in]      **PeiServices
 *    @param[in]      *InitEarly
 *
 *    @retval         VOID
 *
 **/
/*----------------------------------------------------------------------------------------*/
```

```
VOID
OemCustomizeInitEarly (
  IN  EFI_PEI_SERVICES    **PeiServices,
  IN  AMD_EARLY_PARAMS    *InitEarly
  )
{
  //
  // WARNING WARNING WARNING
  // This section should have the implementation to customize the structure
  // which doesn't require any PeiServices to retrieve any settings.
  // In this section, any customization could impact BSP and AP both
  //

  if (PeiServices) {
    //
    // This section should have implementation to customize the structure which may
    // require to use PeiServices to retrieve some info which will help customize the
    // structure. This will be done when this API gets called by BSP.
    //
    EFI_STATUS            Status;
    SYSTEM_CONFIGURATION            SystemConfiguration;
    AMD_CPM_TABLE_PPI               *AmdCpmTablePpiPtr;

    Status = GetSystemConfiguration (PeiServices, &SystemConfiguration);
    if (EFI_ERROR(Status)) {
        SystemConfiguration.SpecialVgaFeature = 0;
    }
    Status = (*PeiServices)->LocatePpi (
                                        PeiServices,
                                        &gAmdCpmTablePpiGuid,
                                        0,
                                        NULL,
                                        &AmdCpmTablePpiPtr
                                       );
    if (!EFI_ERROR (Status)) {
        InitEarly->GnbConfig.PcieComplexList =
                        AmdCpmTablePpiPtr->PeimPublicFunction.PcieComplexDescriporPtr;
        InitEarly->GnbConfig.PsppPolicy         = 0;
    }
}
```