**AptioV DXE Phase源代码分析**

**(参考PI CIS V1.5)**

目录

[**1** **Overview：** 3](#_Toc474241069)

[1.1 PI Architecture Firmware Phase 3](#_Toc474241070)

[1.2 承接：PEI -> DXE -> BDS 3](#_Toc474241071)

[1.3 DXE Foundation：produce a set of Boot Service、Runtime Service、and DXE service。 3](#_Toc474241072)

[1.4 DXE Dispatcher：调度DXE driver。 3](#_Toc474241073)

[1.5 DXE Driver功能 3](#_Toc474241074)

[1.6 Services&Structures **still Available during OS runtime** 3](#_Toc474241075)

[1.7 DXE只需要PEI提供的HobList 3](#_Toc474241076)

[**2** **DXE Phase基本架构：EFI\_SYSTEM\_TABLE、DXE Foundation、DXE Dispatcher、DXE Drivers、Protocols。** 3](#_Toc474241077)

[2.1 **DXE Foundation Data Structures：EFI\_SYSTEM\_TABLE** 3](#_Toc474241078)

[2.1.1 **Console device** 5](#_Toc474241079)

[2.1.2 **\*BootServices** 5](#_Toc474241080)

[2.1.3 **\*RuntimeServices** 6](#_Toc474241081)

[2.1.4 **\*ConfigurationTable** 7](#_Toc474241082)

[2.2 **DXE Foundation** 11](#_Toc474241083)

[2.2.1 **高度抽象的设计原则** 11](#_Toc474241084)

[2.2.2 **DXE Foundation Components** 11](#_Toc474241085)

[2.2.3 **DXE Foundation基本流程** 11](#_Toc474241086)

[2.2.4 DXE Foundation Entry Point 12](#_Toc474241087)

[2.2.5 HobList 12](#_Toc474241088)

[2.2.6 HOB Translations 14](#_Toc474241089)

[2.2.7 Handing Control to DXE Dispatcher 17](#_Toc474241090)

[2.3 **DXE Dispatcher** 19](#_Toc474241091)

[2.3.1 **DXE Drivers Dispatch流程图** 19](#_Toc474241092)

[2.3.2 20](#_Toc474241093)

[2.4 **DXE Drivers** 20](#_Toc474241094)

[2.5 **Protocols** 20](#_Toc474241095)

[2.5.1 **Architectural Protocol：**HobList.FV -> DXE Foundation -> Load&start DXE Driver -> Architectual Protocol -> 初始化EFI System Table、**RuntimeServices**、**BootServices**、DXE Services。 20](#_Toc474241096)

[2.5.2 **BootServices之Memory services：**HobList -> DXE Foundation -> Load&start DXE Driver -> DXE Services.GCD –> BootServices.memory。 20](#_Toc474241097)

[2.5.3 **Additional Runtime Protocol** 20](#_Toc474241098)

[**3** **Boot Manager** 20](#_Toc474241099)

[3.1 Boot Manager是个firmware policy engine，维护着 a list of NVRAM variables（即boot order list，每个NVRAM variable包括boot option name、a pointer to hw device and a pointer to UEFI image in that hw device、path to OS partition and directory、load option passed to UEFI image）；BootManager根据这些NVRAM变量来决定按何种顺序引导UEFI Drivers或UEFI APPs（包括UEFI OS Loader）。 20](#_Toc474241100)

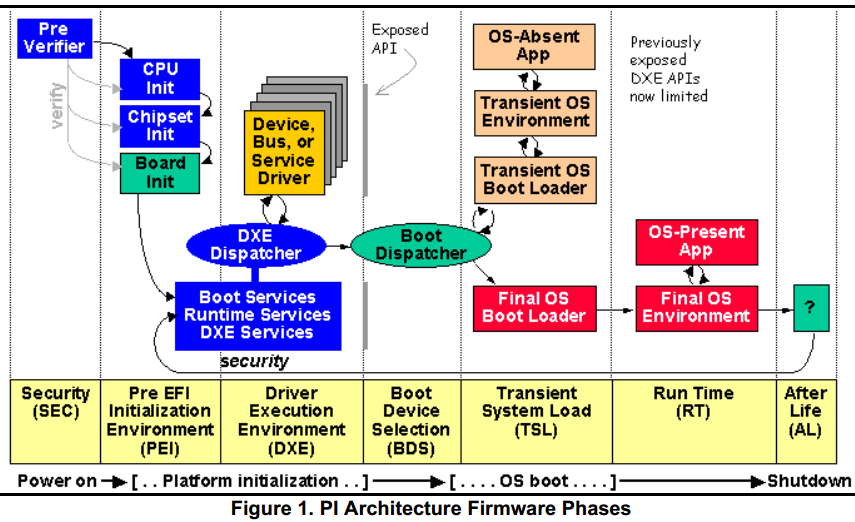
[3.2 DXE Dispatcher dispaches all other DXE Drivers 🡪 BDS Architectural Protocol 🡪 dispatch BootManager(也就进入BDS phase) 20](#_Toc474241101)

[3.3 BDS phase 主要工作： 20](#_Toc474241102)

[3.4 如果BDS phase不能走完，那么将回调DXE Disptcher。 22](#_Toc474241103)

[**4** **UEFI Image Entry Point Examples** 22](#_Toc474241104)

1. **Overview：**
   1. PI Architecture Firmware Phase



* 1. 承接：PEI -> DXE -> BDS
  2. DXE Foundation：produce a set of Boot Service、Runtime Service、and DXE service。
  3. DXE Dispatcher：调度DXE driver。
  4. DXE Driver功能

初始化processor、chipset、platform部件，及Service、Boot device、console device抽象。

* 1. Services&Structures **still Available during OS runtime**

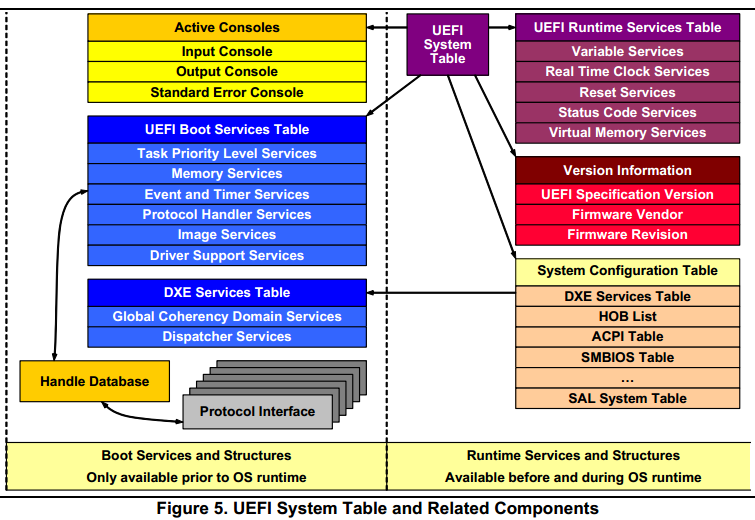
进入OS后只会保留Runtime service（DXE driver）和data structure（DXE Foundation和Driver）。

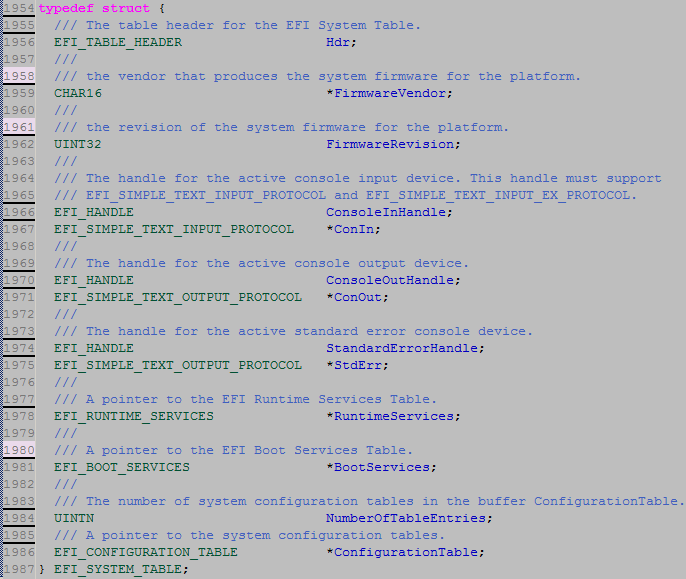
* 1. DXE只需要PEI提供的HobList

也就是说，如果有其它方案获取HobList，可删除PEI phase。

1. **DXE Phase基本架构：****EFI\_SYSTEM\_TABLE、DXE Foundation、DXE Dispatcher、DXE Drivers、Protocols。**
   1. **DXE Foundation Data Structures：EFI\_SYSTEM\_TABLE**

见下图，主要包括**Console device**、**\*BootServices**、**\*RuntimeServices**、**\*ConfigurationTable**（包括**DXE\_SERVICES**、HobList、ACPI Table、SMBIOS Table等）。

****

****

* + 1. **Console device**

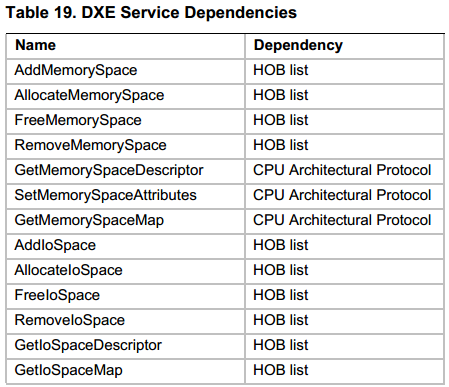
ConsoleInHandle、\*ConIn protocol；ConsoleOutHandle、\*ConOut protocol、StandardErrorHandle、\*StdErr protocol。

* + 1. **\*BootServices**
       1. **Task Priority**：可实现simple lock、disable interrupt。
          1. RaiseTPL
          2. RestoreTPL。
       2. **Memory**：allocate、free memory pool、page，get memory map
          1. AllocatePages
          2. FreePages
          3. GetMemoryMap
          4. AllocatePool
          5. FreePool
          6. CopyMem
          7. SetMem
       3. **Event and Timer event**：create、 wait for 、 signal 、 close、 check status of event，timer event支持单次或间隔多次类型的event。
          1. CreateEventEx
          2. CreateEvent
          3. WaitForEvent
          4. SignalEvent
          5. CloseEvent
          6. CheckEvent
          7. SetTimer
       4. **Protocols and Handles**：add、remove、look for Handles from Handle Database；add、remove、open、close protocols from Handle Database
          1. InstallProtocolInterface（install a protocol on 某Handle）
          2. ReinstallProtocolInterface
          3. UninstallProtocolInterface
          4. **InstallMultipleProtocolInterfaces**
          5. **UninstallMultipleProtocolInterfaces**
          6. **OpenProtocol**（返回某Handle上指定GUID的protocol）
          7. **HandleProtocol**（参数简化版的OpenProtocol）
          8. **LocateProtocol**（返回Handle Database里第1个指定GUID的protocol）
          9. **ProtocolsPerHandle**（返回某Handle的所有protocol）
          10. CloseProtocol
          11. OpenProtocolInformation
          12. RegisterProtocolNotify（register event on protocol，wait for which to be installed to notify the event）
          13. LocateHandle（返回support指定GUID的Handles，调用者申请内存）
          14. LocateHandleBuffer（返回support指定GUID的Handles buffer，本库函数申请内存）
          15. LocateDevicePath（查找某Device Handle及其对应的DevicePath，which support指定的protocol）
       5. **Image Services**：LoadImage、StartImage、Exit、UnloadImage、ExitBootServices
       6. **Driver Support**：
          1. ConnectController：connect \***Driver**ImageHandle to **Controller**Handle.
          2. DisconnectController
       7. **Misc Services**
          1. CalculateCrc32
          2. GetNextMonotonicCount
          3. Stall
          4. SetWatchdogTimer
    2. **\*RuntimeServices**
       1. **Time services**
          1. GetTime
          2. SetTime
          3. GetWakeupTime
          4. SetWakeupTime
       2. **Virtual Memory Services**
          1. SetVirtualAddressMap(仅能被调用1次，relocate runtime image，把gRT和gST里指针类型成员的地址由physical address转换成virtual address)。
          2. ConvertPointer(把physical address转换成virtual address)。
       3. **Variable services**
          1. Getvariable
          2. GetNextVariableName
          3. SetVariable
       4. **Capsule services**
          1. UpdateCapsule

直接更新BIOS，或

把firmware update capsule加载到RAM，soft reset system（soft reset不会导致RAM数据丢失），PEI阶段检查update capsule状态，如果须update，则执行update，最后lock flash protect

* + - * 1. QueryCapsuleCapabilities
      1. **Misc Services**
         1. GetNextHighMonotonicCount
         2. ResetSystem
         3. QueryVariableInfo
    1. **\*ConfigurationTable**
       1. 包括DXE\_SERVICES Table、HobLists、ACPI Table、SMBIOS Table、SAL System Table

****

* + - 1. DXE\_SERVICES Table：**Contain pointors to DXE-specific services, for DXE Foundation and Drivers.**

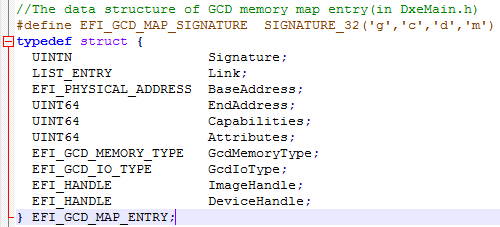
****

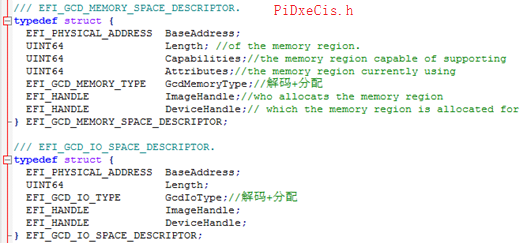
* + - * 1. Global Coherency Domain (GCD) Services：管理BootProcessor可用的system memory、MMIO、IO。

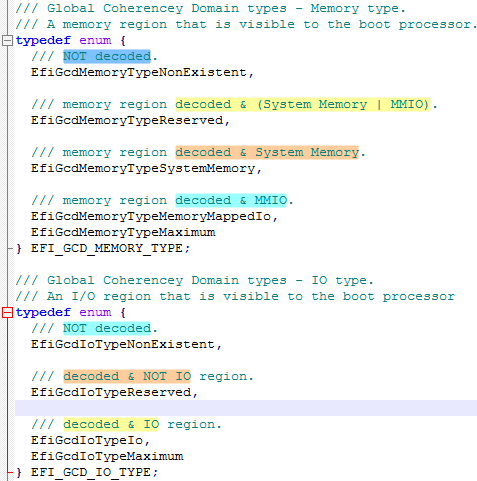
GCD：由HobList初始化，GCD将用来初始化memory相关的\*BootServices(如AllocatePages(), FreePages(), AllocatePool(), FreePool())。

GCD方法：见上图

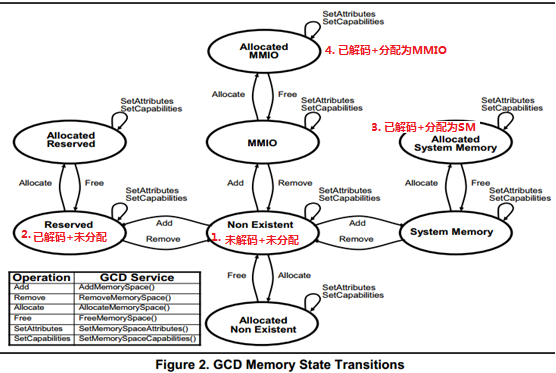
GCD数据结构



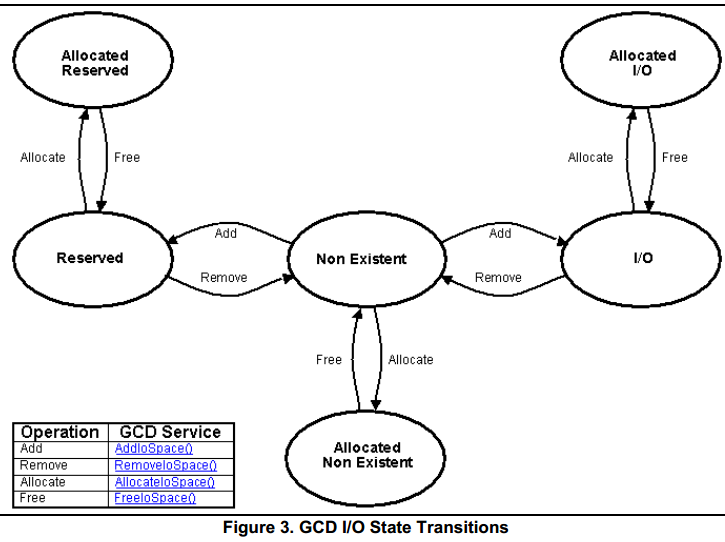




GCD memory state转换

****

GCD IO state转换

****

* + - * 1. Dispatch Services：设定DXE Driver状态以及DXE Driver调度。

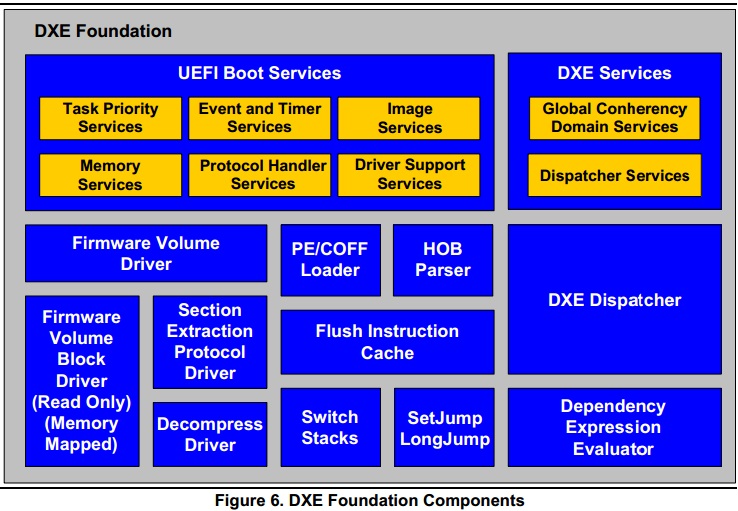
*Dispatch：*Load、check depex、execute DXE Driver。

*Schedule(FirmwareVolumeHandle, \*FileName)：*取消DXE Driver的Schedule On Request flag（即只要满足depex即可execute）。

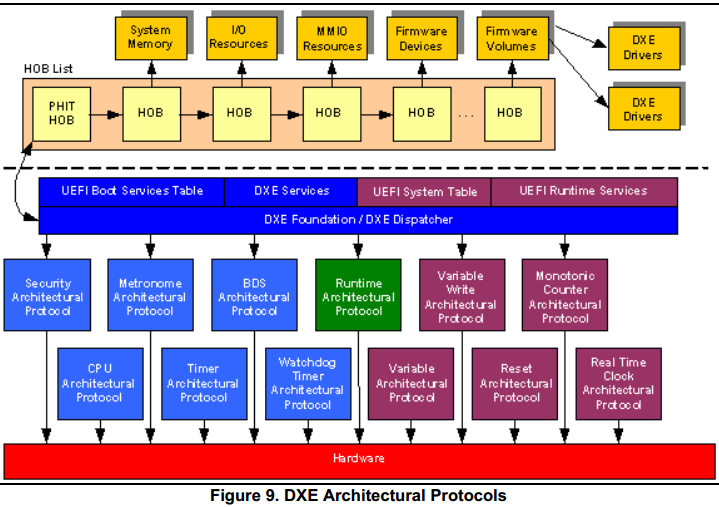
*Trust**(FirmwareVolumeHandle, \*FileName)：*标记DXE Driver为Trust。

*ProcessFirmwareVolume(\*FirmwareVolumeHeader, Size, OUT \*FirmwareVolumeHandle)：*从加载到system memory中的Firmware Volume中获取*\*FirmwareVolumeHandle*供*Dispatch、Schedule、Trust*使用；同时为该Handle安装**EFI\_FIRMWARE\_VOLUME2\_PROTOCOL、EFI\_DEVICE\_PATH\_PROTOCOL**。

* 1. **DXE Foundation**
     1. **高度抽象的设计原则**
        1. Only depend on HobList from PEI（即只要有其它方法提供HobList，PEI Phase可删）。
        2. DO NOT contain any hard-code
        3. DO NOT contain any processor-specific、chipset-specific、platform-specific information
     2. **DXE Foundation Components**
        1. 功能模块包括EFI System Table(BootServices、DXE Services)、DXE Dispatcher（order by priori file and depex in DXE image）、PE/COFF Loader、HobList Parse等等（详见下图）。

****

* + 1. **DXE Foundation基本流程**
       1. DXE Foundation解析PEI传递来的HobList
       2. 初始化EFI System Table、DXE Services、BootServices、DXE Services（部分Services指向空指针，即部分services只有壳，没实体）。
       3. 创建用来实体化DXE Architectual/Option Protocol（进而初始化上述空指针）的event及其处理函数。
       4. 调用DXE Dispatcher调度DXE Drivers。
       5. 早期DXE Drivers将install DXE Architectual/Option Protocol，触发event的处理函数，实体化DXE Architectual/Option Protocol，进而实体化上述空指针。
       6. 进入BDS阶段

****

* + 1. DXE Foundation Entry Point
       1. EntryPoint类型

**typedef** VOID (EFIAPI \*DXE\_ENTRY\_POINT) (IN CONST VOID \*HobStart);

* + - 1. EntryPoint（in DxeCoreEntryPoint.c）

VOID EFIAPI **\_ModuleEntryPoint** ( IN VOID \*HobStart)

|🡪**ProcessModuleEntryPointList** (HobStart);

|==**DxeMain** (HobStart) in DxeMain.c

|🡪初始化EFI System Table、DXE Services、BootServices、DXE Services（部分

Services指向空指针）。

|🡪install一些protocol，如FV、Section、Decompress

|🡪**CoreNotifyOnProtocolInstallation():**注册1个Event来更新DXE

| Architectural Protocol和DXE Option Protocol实体，当相应protocol

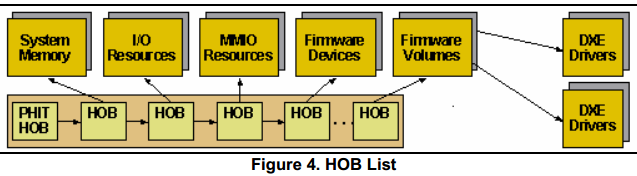
| 被某些DXE Driver installed时。

|🡪**CoreDispatcher();**早期DXE Driver将初始化Arch和Option Protocol。

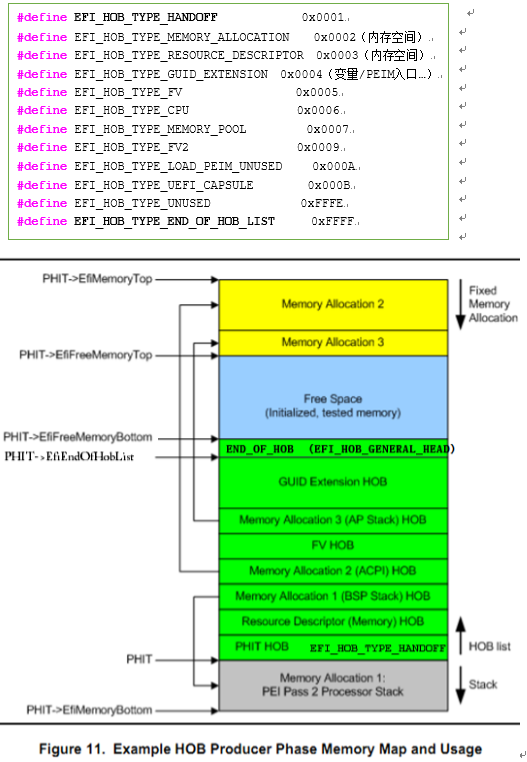
| 填充上述services的空指针。

|🡪**gBds->Entry (gBds)；**进入BDS阶段。

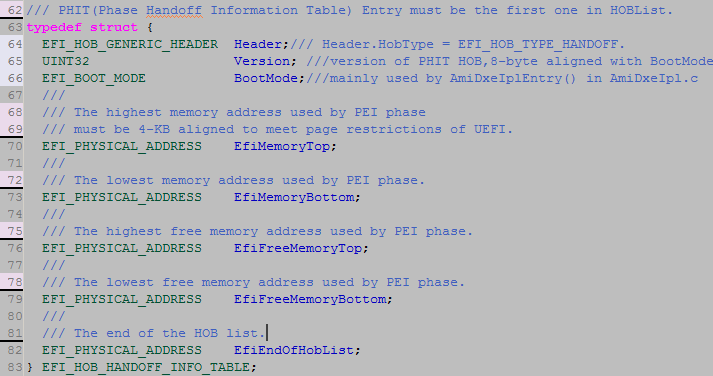
* + 1. HobList
       1. DXE phase仅用PEI phase传递HobList就可完成本phase的全部工作。
       2. HobList内容
          1. Information about BootMode
          2. System memory used by PEI phase
          3. **System memory maps（SM、IO、MMIO）** initialized by PEI phase
          4. Information about firmware device(system memory locations of **firmware device** and **fv**) discovered by PEI phase
       3. HobList示例



* + - * 1. 第1个HobList entry是Phase Handoff Information Table（PHIT），包含BootMode和description about system memory used by PEI phase。
        2. 最后1个HobList Entry是END\_OD\_HOB\_LIST。
        3. 对于DXE Foundation，system memory和fv HobList entry最重要
        4. GUIDed HobList entry用来允许PEIM传递私有数据给DXE Driver。
        5. HobList是location无关的，DXE Foundation可以relocate它。
      1. HobList Type和Memory Map



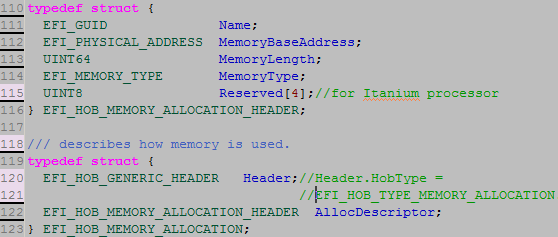
* + 1. HOB Translations
       1. HobList传输过程：HobList -> GCD services(memory/IO)、UEFI Memory map -> BootServices和DXEServices的memory部分。
       2. **PHIT** Hob Entry（类型：EFI\_HOB\_TYPE\_HANDOFF,in PiHob.h）
          1. PHIT entry类型（参考Figure11）

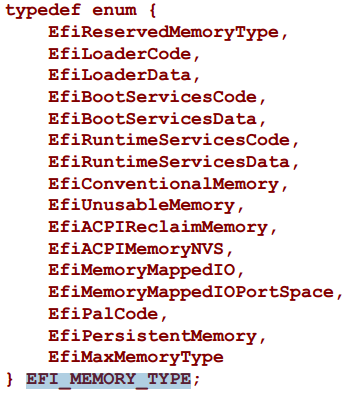


* + - * 1. 主要内容：见Figure11，包括**BootMode**、**HobList起止地址**（DXE Foundation的HobStart参数）、**free memory**（供DXE早期使用，直到UEFI Memory Services准备就绪）、其它logical memory。
      1. **Memory Allocation** Hob Entry（类型：EFI\_HOB\_TYPE\_MEMORY\_ALLOCATION）
         1. DXE foundation parses **Memory Allocation** Hob.
         2. Call GCD.AllocateMemorySpace() 以更新**GCD Memory Space Map**。
         3. 同时更新**UEFI Memory Map**，以使**Memory相关BootServices可用**

（AllocatePages、FreePages、AllocatePool、FreePool等）。

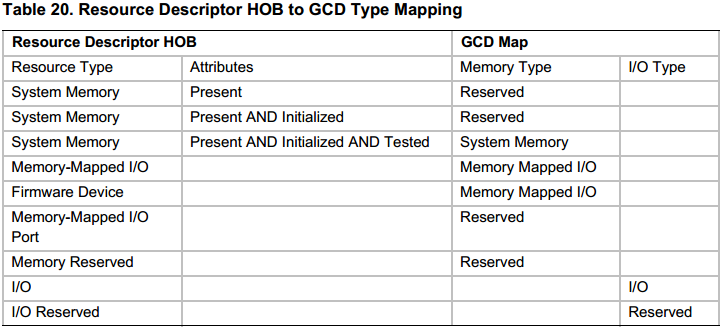
* + - * 1. Memory Allocation类型



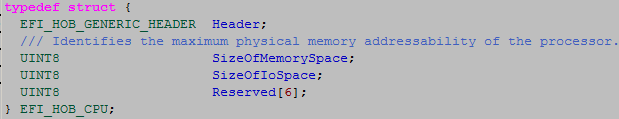


* + - 1. **Resource Descriptor** Hob Entry（类型：EFI\_HOB\_TYPE\_RESOURCE\_DESCRIPTOR）

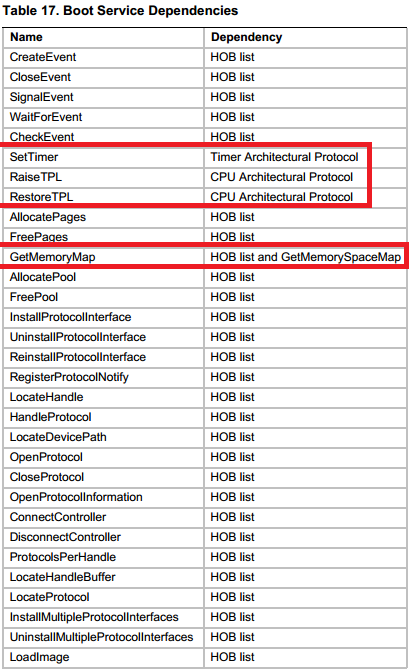
依据此Hob内容，DXE Foundation使用the DXE Services（AddMemorySpace() and AddIoSpace() ）to register these memory and I/O resources in the GCD map。

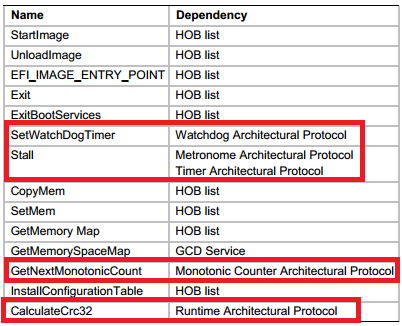


* + - 1. **GUID Extension** Hob entry（类型：EFI\_HOB\_TYPE\_GUID\_EXTENSION）
         1. 用来PEIM向DXE Driver传递私有数据，DXE Driver自己负责解析。
      2. **Firmware Volume** Hob entry（类型：EFI\_HOB\_TYPE\_FV、EFI\_HOB\_TYPE\_FV2）
         1. EFI\_HOB\_FIRMWARE\_VOLUME：描述FV起止地址
         2. EFI\_HOB\_FIRMWARE\_VOLUME2：描述FV.FSS.FV起止地址
         3. DXE Dispatcher确认该Fv没有被处理过后，install Handle in Handle Database，并于此install **EFI\_FIRMWARE\_VOLUME2\_PROTOCOL。**
         4. The DXE Service **AllocateMemorySpace()** is used to allocate the memory regions described in the Firmware Volume HOBs to the DXE Foundation.
         5. The UEFI Boot Service **InstallProtocolInterface()** is used to create new handles and install protocol interfaces.
      3. **CPU** Hob Entry（类型：EFI\_HOB\_TYPE\_CPU）
         1. SizeOfMemorySpace指定CPU能够访问的最大SM的bit位数。
         2. DXE foundation初始化GCD space map （0 ~ (1<<SizeOfMemorySpace））entry with **EfiGcdMemoryTypeNonExistent**。

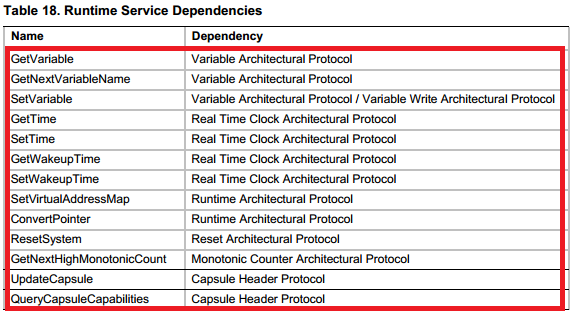


* + - 1. **End of HobList** Hob entry（类型：EFI\_HOB\_TYPE\_END\_OF\_HOB\_LIST）
    1. Handing Control to DXE Dispatcher
       1. **准备早期内存**：Use the **HOB list** to initialize the **GCD memory space map**, the **GCD I/O** space map, and **UEFI memory map**.
       2. **构建DXE Foundation框架（含早期memory services）**：gBS、gDS分配内存（BootMemory），并初始化内存管理相关services，其它services为**EFI\_NOT\_AVAILABLE\_YET**
       3. 为gRS分配内存（RuntimeMemory），所有services为**EFI\_NOT\_AVAILABLE\_YET**。
       4. 初始化gST（Runtime Memory）。
       5. **准备Load DXE Image**：为己build ImageHandle及**EFI\_LOADED\_IMAGE\_PROTOCOL**，插入Handle Database。
       6. Relocation HobList（Optional）。
       7. **初始化DXE Foundation框架1/2：**Add DXE services to gST.SystemConfiguration
       8. Add HobList to gST.SystemConfiguration
       9. **初始化DXE Foundation框架2/2：**Init UEFI Boot Services, Runtime Services, and DXE Services（Create event及关联的notify function：init gBS、gRS、gDS 🡨 **DXE Foundation**执行notify function 🡨 触发Event 🡨 DXE Driver install Architecture/Adtional Protocol 🡨 **DXE Dispatcher**调度执行DXE Driver）。
       10. 在处理compressed section前，初始化Decompress Protocol driver。
       11. 准备FV Handle：根据HobList描述的FV创建。
       12. 所有DXE Driver都执行完后，进入BDS。
    2. Dependencies
       1. Boot Services Dependencies

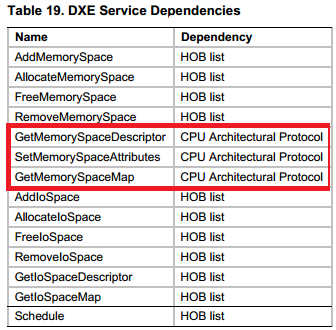




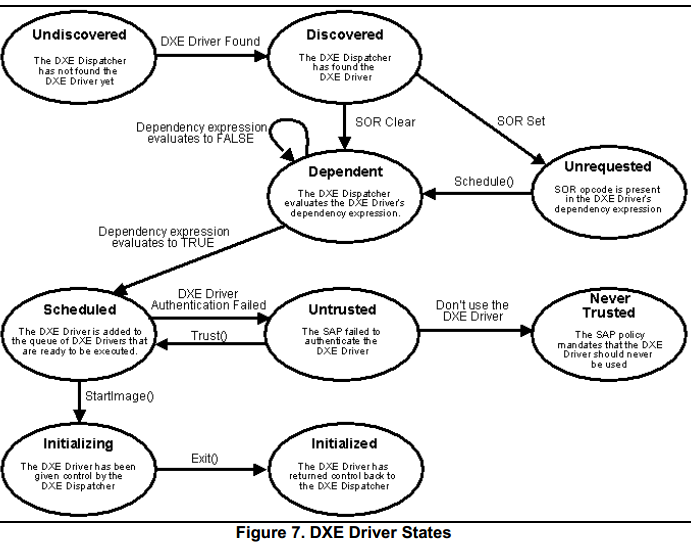
* + - 1. Runtime Services Dependencies



* + - 1. DXE Services Dependencies

****

* 1. **DXE Dispatcher**
     1. **DXE Drivers Dispatch流程图**

****

* 1. **DXE Drivers**
  2. **Protocols**
     1. **Architectural Protocol：**HobList.FV -> DXE Foundation -> Load&start DXE Driver -> Architectual Protocol -> 初始化EFI System Table、**RuntimeServices**、**BootServices**、DXE Services。
     2. **BootServices之Memory services：**HobList -> DXE Foundation -> Load&start DXE Driver -> DXE Services.GCD –> BootServices.memory。
     3. **Additional Runtime Protocol**

1. **Boot Manager**
   1. Boot Manager是个firmware policy engine，维护着 a list of NVRAM variables（即boot order list，每个NVRAM variable（Boot#### NVRAM）包括boot option name、a pointer to hw device and a pointer to UEFI image in that hw device、path to OS partition and directory、load option passed to UEFI image）；BootManager根据这些NVRAM变量来决定按何种顺序引导UEFI Drivers或UEFI APPs（包括UEFI OS Loader）。
   2. DXE Dispatcher dispaches all other DXE Drivers 🡪 BDS Architectural Protocol 🡪 dispatch BootManager(也就进入BDS phase)
   3. BDS phase 主要工作：

**BDS\_CONTROL\_FLOW = \**

BeforeBdsFlow,\

PrepareS3Info,\

SignalConnectDriversEvent,\

AmiGopPolicyProtocolInsatll,\

ConnectRootBridgeHandles,\

TbtDxeInvokeSmiHandler,\

RegisterMemoryTypeInformationUpdateCallback,\

FastBootHook,\

**/\* Connect Console Devices /\**

/\* Connect ConOut Devices \*/\

ReportConnectConOutProgressCode,\

ConnectAmiGopPolicyConOut,\

ConnectVgaConOut,\

ConnectConOutVariable,\

InstallConOutStartedProtocol,\

SignalEndOfDxeEvent,\

/\* Connect ConIn Devices \*/\

ReportConnectConInProgressCode,\

ConnectPs2ConIn,\

ConnectUsbConIn,\

ConnectConInVariable,\

InstallConInStartedProtocol,\

**/\* Full System Initialization \*/\**

ConnectEverything,\

RunDrivers,\

InitConVars,\

ShadowAllLegacyOptionRoms,\

LocateAndInitializePcieSsd,\

SignalAllDriversConnectedEvent,\

SignalExitPmAuthProtocolEvent,\

InstallFwLoadFile,\

DefaultFixedBootOrder\_Init,\

UpdateBootOptionVariables,\

AdjustEfiOsBootOrder,\

FixedBootOrderPatchWithCFast,\

FBO\_Init,\

ReadBootOptions,\

RemoveFastBootOption,\

FboUpdateBoTagDeviceType,\

FBO\_AdjustWtgPriority,\

UnmaskOrphanDevices,\

CollectUsbBbsDevices,\

CollectBootDevices,\

FilterBootDeviceList,\

CreateEfiOsBootOption,\

MatchBootOptionsToDevices,\

DeleteUnmatchedUefiHddBootDevices,\

FboSyncBootOptionsFromDevOrder,\

FboSceCheck,\

FboAdjustEthernetDefaultPriority,\

CreateBootOptionsForNewBootDevices,\

**/\*Process Boot Option List\*/\**

FBO\_SetBootOptionTags,\

EfiOsName\_NormalizeBootOptions,\

PreProcessBootOptions,\

RestoreSpecialOrphanBootOptions,\

SetBootOptionPriorities,\

AdjustNewUefiOsOptionPriority,\

PostProcessBootOptions,\

MaskOrphanDevices,\

FBO\_UpdateBootListPriority,\

FboSetLegacyDeviceType,\

SetDefaultTseBootOrder,\

SetBootNextBootOption,\

FBO\_SetDefaultUEFIDevOrder,\

FBO\_AdjustDeviceName,\

FboSceBuildDefault,\

SaveBootOptions,\

CreateOldBootOrder,\

CallTheDispatcher,\

RecoverTheMemoryAbove4Gb,\

PerformCapsuleVariableCleanup,\

HandoffToTse,\

ReadBootOptionAndBoot,

* 1. 如果BDS phase不能走完，那么将回调DXE Disptcher。

1. **UEFI Image Entry Point Examples**