tianocore

UEFI & EDK II Training

EDK II Modules: Libraries, Drivers & Applications

tianocore.org

Lesson Objective

- What is a EDK II Module
- Use EDK II libraries to write UEFI apps/drivers
- How to Define a UEFI application
- Differences between UEFI App / Drivers INF file

tianocore

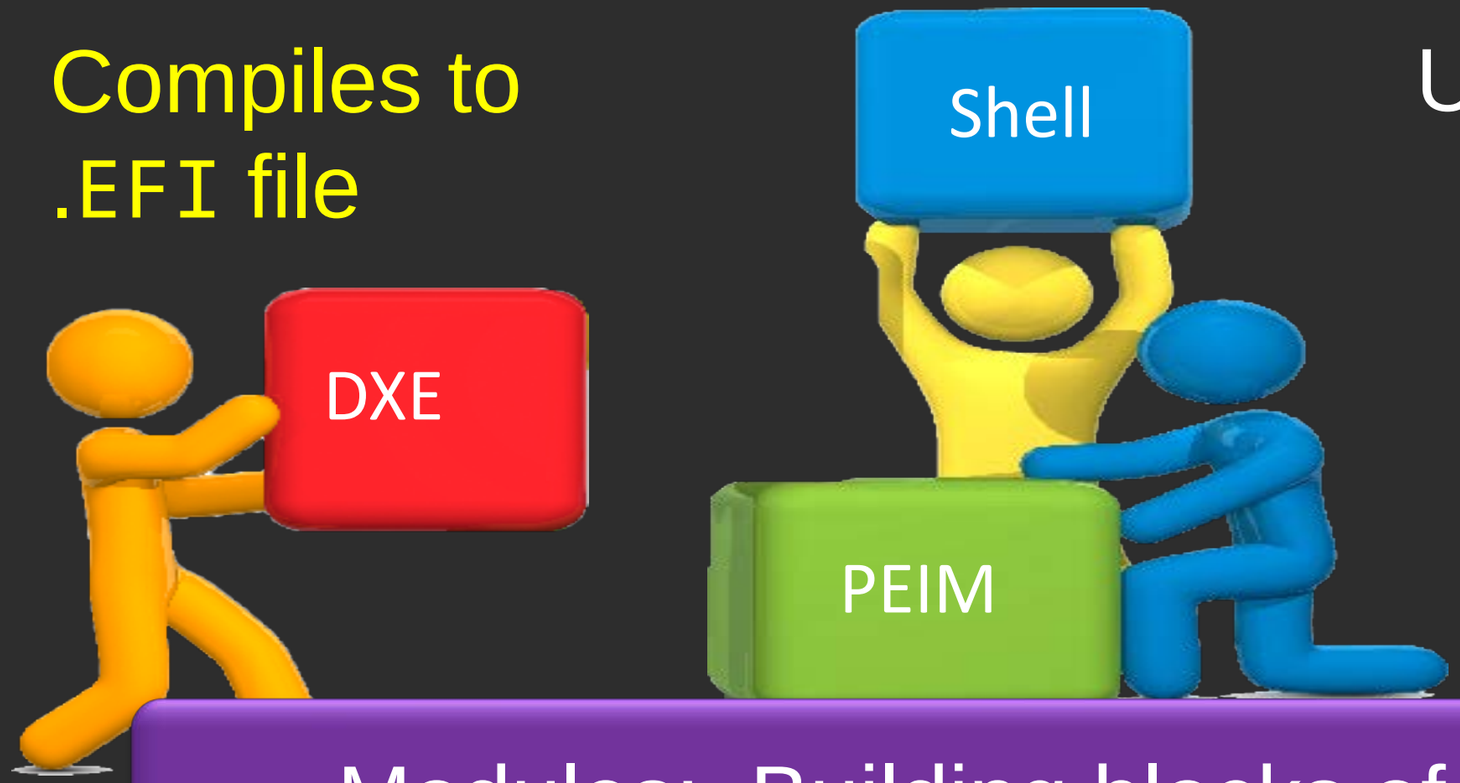
EDK II MODULES OVERVIEW

What are EDK II Modules

MODULES

Smallest separate object compiled in EDK II

Compiles to
`.EFI` file



UEFI/DXE Driver
PEIM
UEFI App. or
Library

Modules: Building blocks of EDK II

MODULE TYPES

Most Used Module Types

PEI_CORE --> UEFI_APPLICATION DXE_CORE

BASE DXE_RUNTIME_DRIVER

PEIM UEFI_DRIVER DXE_DRIVER

Syntax:

```
<ModuleTypes>    ::= <ModuleType> [<Space> <ModuleType>]
```

MODULE SOURCE CONTENTS - MINIMUM FILE

MODULE_TYPE	Example Source files
UEFI_APPLICATION	`Foo.c, Foo.inf`
UEFI_DRIVER	`FooDriver.c, FooDriver.h, FooDriver.vfr, FooDriver.uni, FooDriver.inf`
DXE_DRIVER	`FooDxe.c, FooDxe.h, FooDxe.inf`
DXE, UEFI or PEIM Library	`FooLib.c, FooLib.h, FooLib.inf` (w/ LIBRARY_CLASS=)

Complexity - Greater number of source files

.INF file - One file is required per module

.EFI file - Sources compiled to a single .EFI file

EDK II LIBRARY MODULES

Library Class

Syntax:

```
[LibraryClasses.common]
  <LibraryClassName>|<LibraryInstancePathToInf/Name.inf>

  DebugLib|MdePkg/Library/BaseDebugLibNull/BaseDebugLibNull.inf
```

Name | **Implementation[3]**

Consistent set of interfaces

Does not describe implementation of the interfaces

“NULL” Library Class

Constructors

Special Cases

NOT “. . . LibNull” instance

Syntax

```
Pkg/MyModule/MyModule.inf {
  <LibraryClasses>
      NULL|Pkg/Library/LibName/LibName.inf
      NULL|Pkg/Library/LibName2/LibName2.inf
  }
```

Open Source Example

DxeCrc32GuidedSectionExtractLib
ShellPkg as used with Profiles

UEFI Shell example:

```
ShellPkg/Application/Shell/Shell.inf {
    <LibraryClasses>
      NULL|ShellPkg/Library/UefiShellDriver1CommandsLib/UefiShellDriver1CommandsLib.inf
      NULL|ShellPkg/Library/UefiShellNetwork1CommandsLib/UefiShellNetwork1CommandsLib.inf
          . . .
}
```

More on “NULL” Lib Link

Locating Library Classes

Library based upon

1. Industry specs (UEFI, etc.)
   MdePkg/MdeModulePkg
2. Features
   NetworkPkg/SecurityPkg

Use the package help files (.CHM) to find a library or function

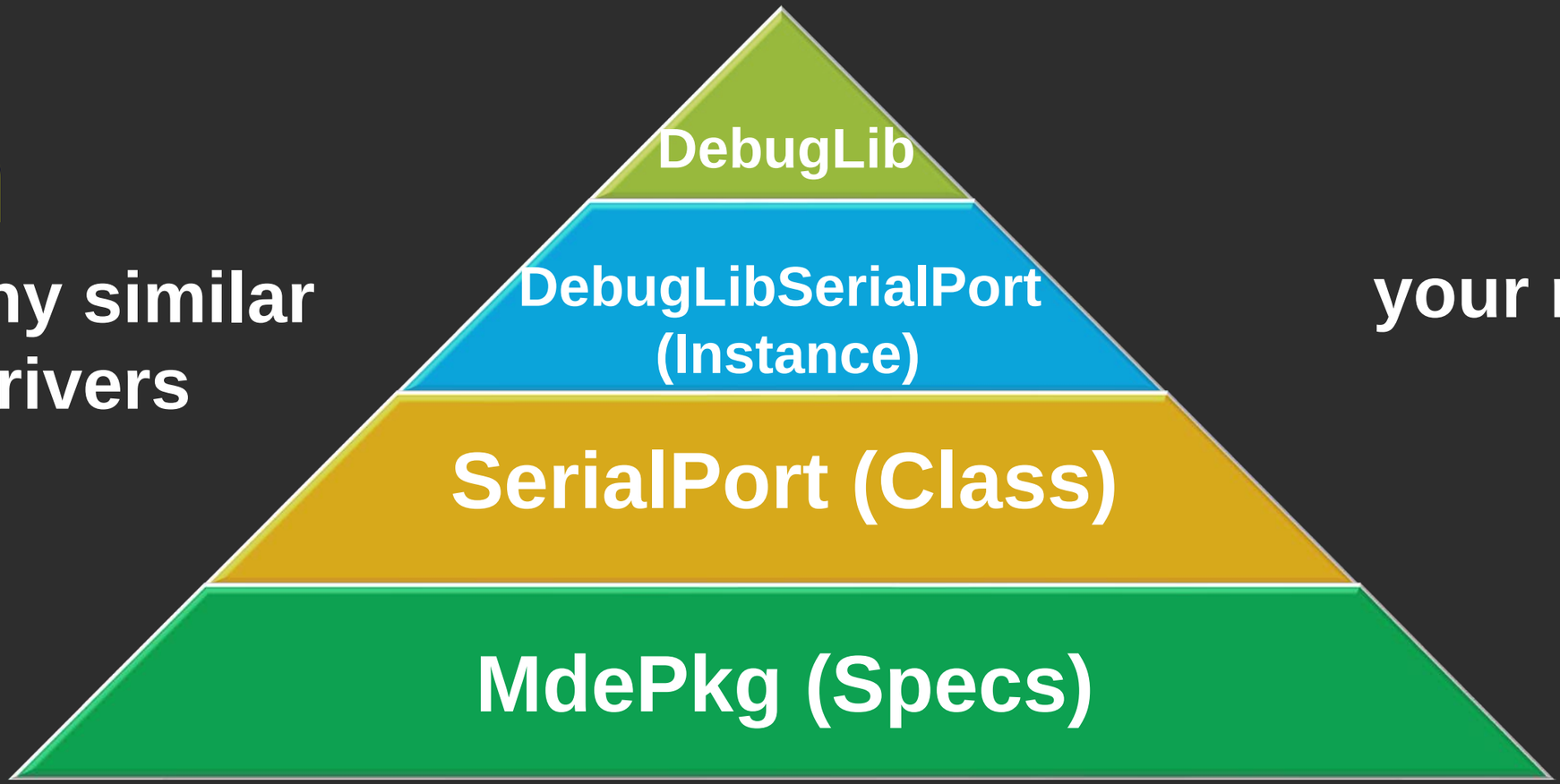
Example: `MdePkg.chm`

Search WorkSpace (.INF) **"LIBRARY_CLASS"**

Library Instance Hierarchy

Form

a hierarchy similar to UEFI drivers



Link

your module to another

Build error : Instance of Library class [*Foo…Lib*] is not found ….
Consumed by module [*My Module.inf*]

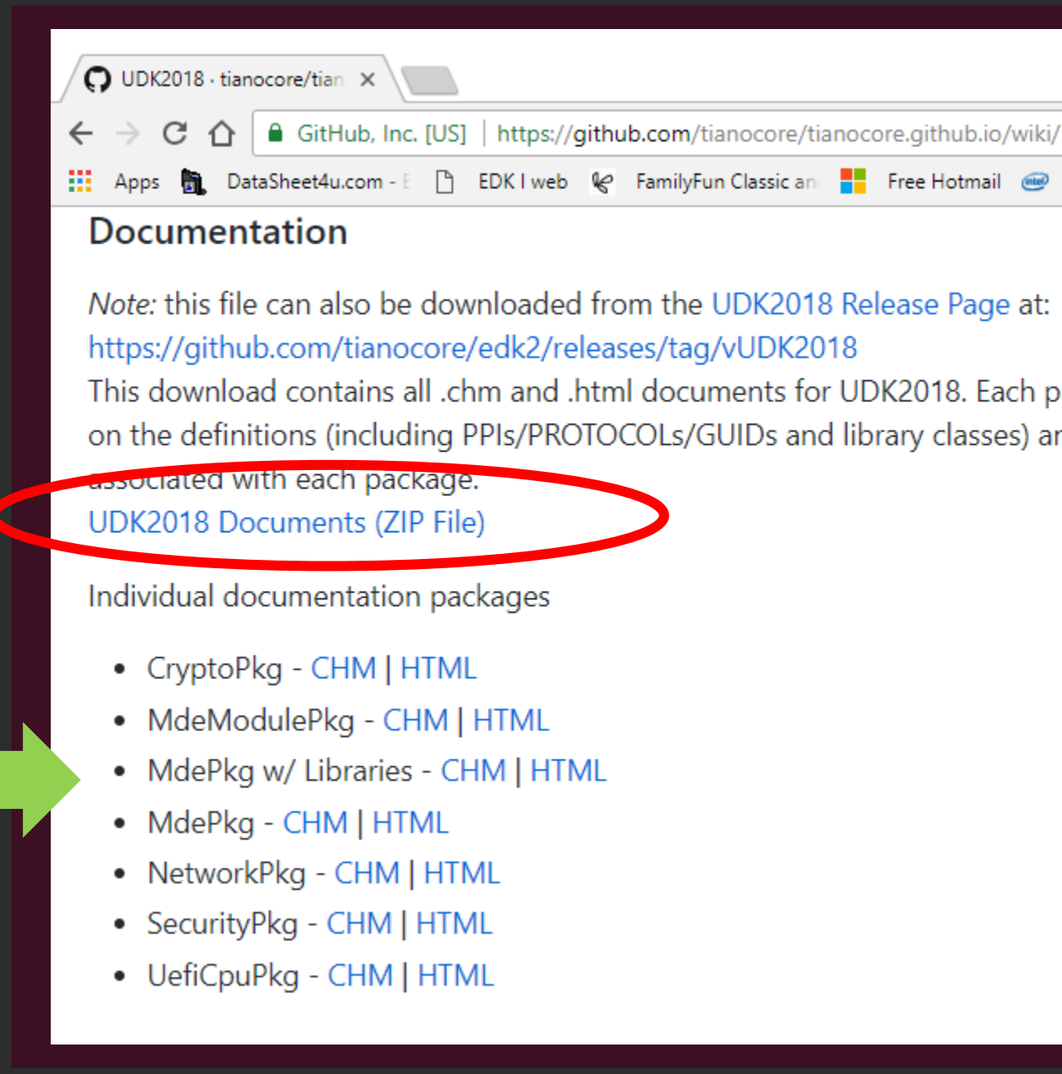
Commonly Used Base Library Classes

BaseLib DebugLib UefiDriverEntryPoint
UefiLib UefiBootServicesTableLib
UefiApplicationEntryPoint DxeCoreEntryPoint
CpuLib UefiUsbLib PciLib DevicePathLib IoLib
MemoryAllocationLib PrintLib PeimEntryPoint
PeiCoreEntryPoint UefiScsiLib BaseMemoryLib
UefiRuntimeLib SmmMemLib
DxeSerivesLib SynchronizationLib PciExpressLib
DxePcdLib UefiRuntimeServicesTableLib
PciSegmentLibLib PeiServicesLib
PeiPcdLib DxeHobLib UefiFileHandleLib

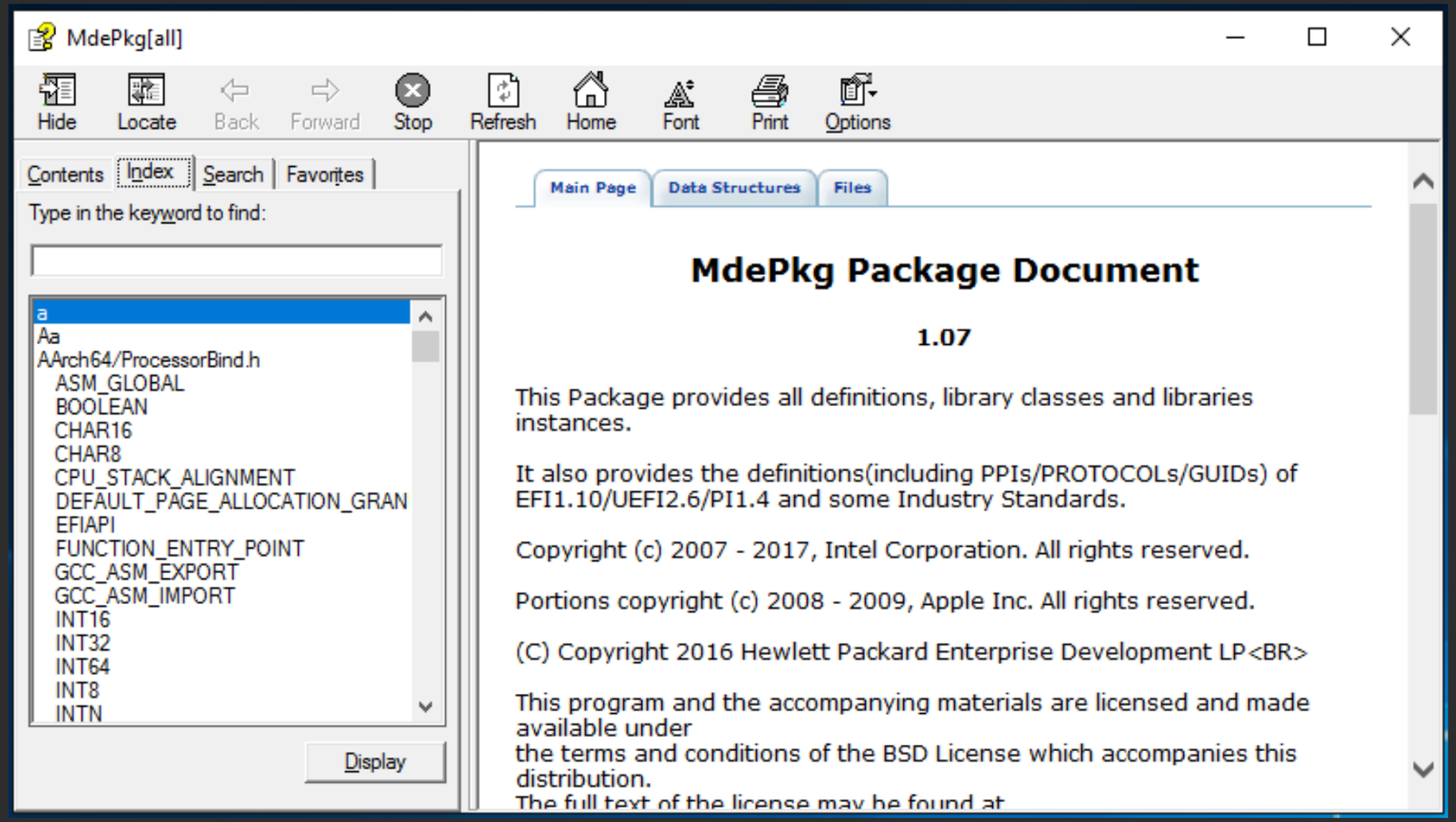
MdePkg Library .CHM file Location

tianocore.org UDK2018 documentation on

- Latest UDK Release
- UDK2018



Library Navigation Demonstration

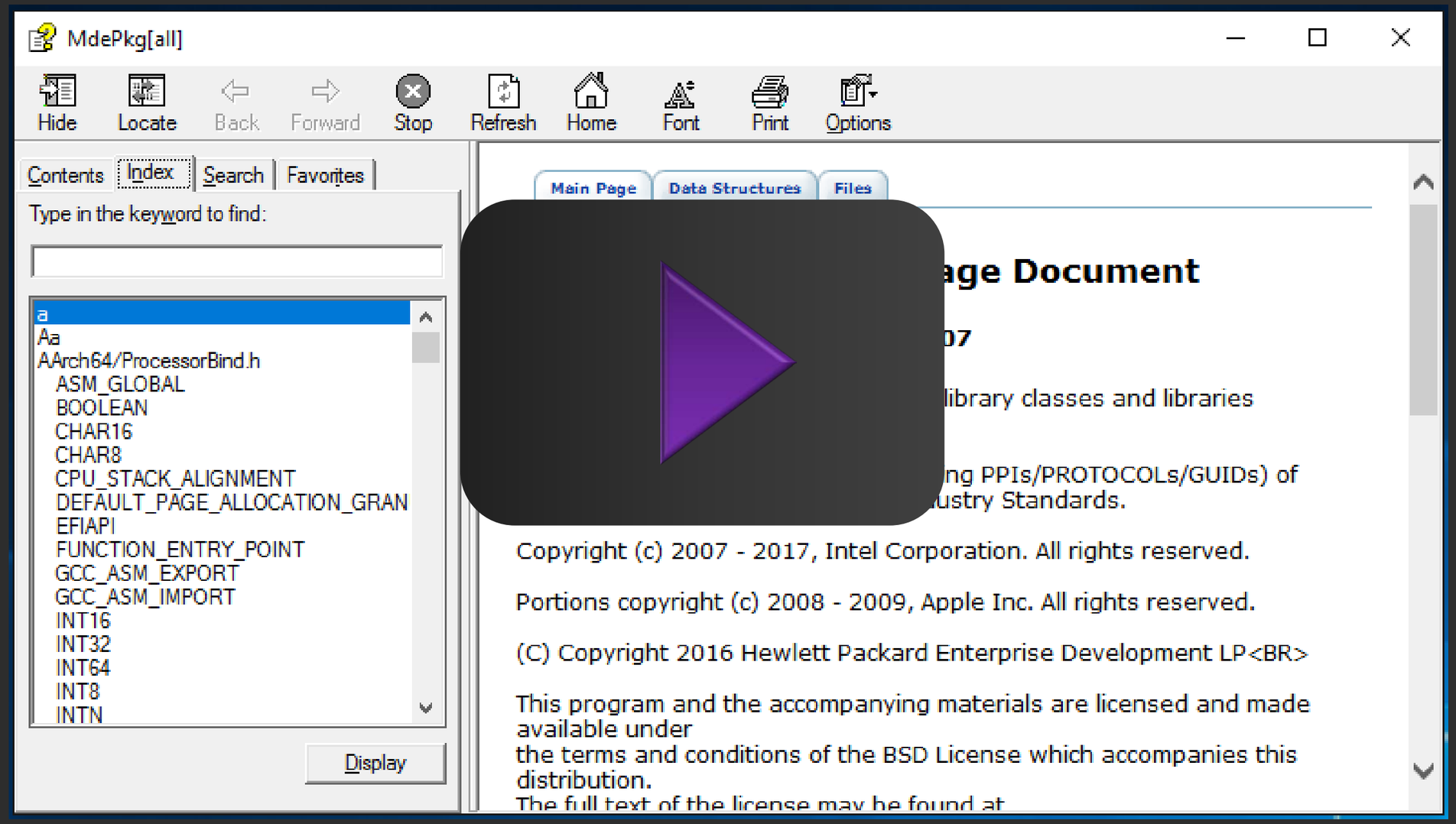


Open file: /FW/Documentation/”MdePkg Document With LibrariesMdePkg.chm”

NOTE: Install a CHM Viewer for Ubuntu

```
bash$ sudo aptitude install kchmviewer
```

Library Navigation Demonstration



https://youtu.be/s8Zw1w1iQS4

tianocore

EDK II UEFI APPLICATION

Defining a UEFI Application

Characteristics of a UEFI Application

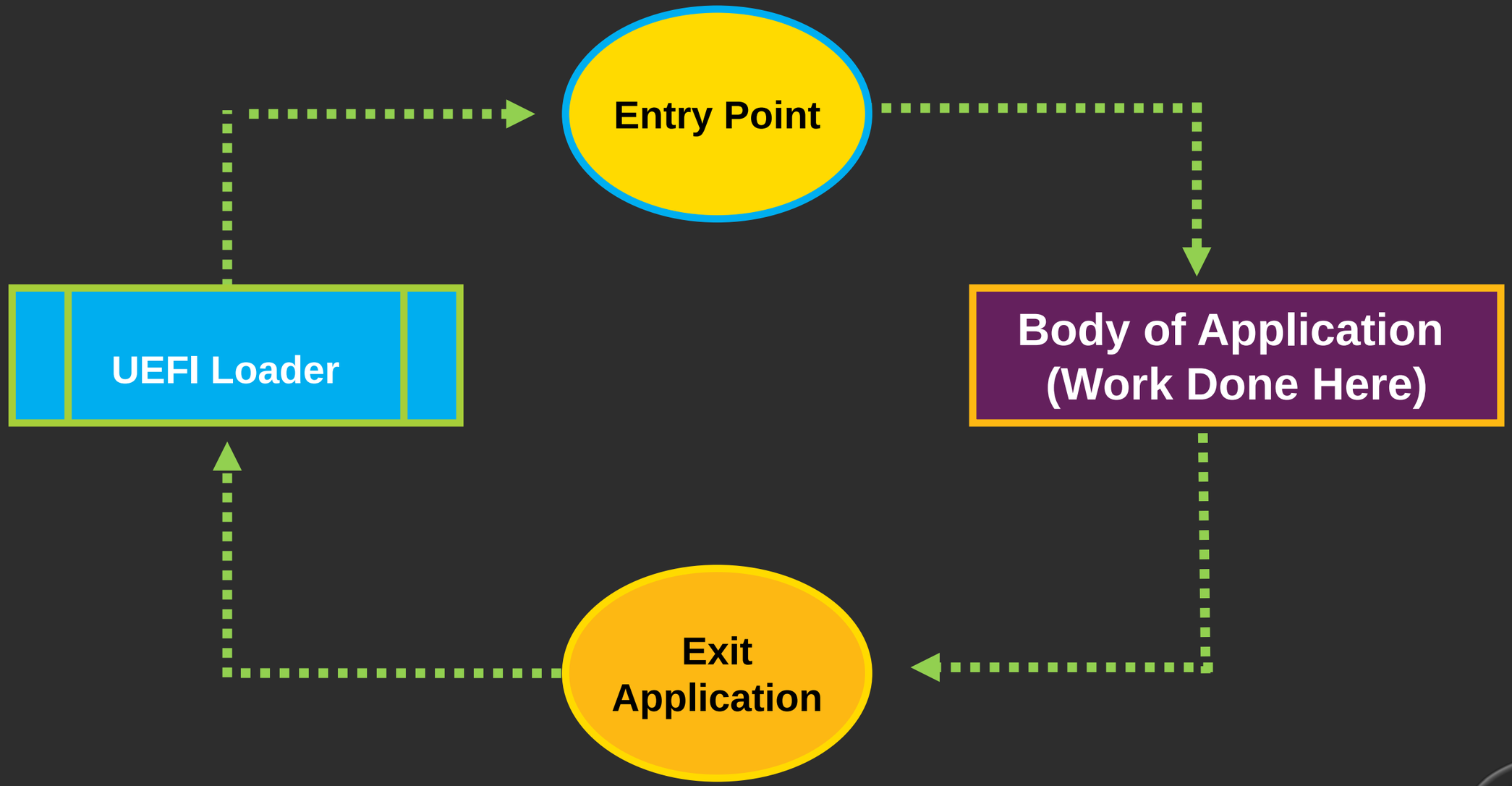
- Loaded by UEFI loader, just like drivers
- Does not register protocols
- Consumes protocols
- Typically exits when completed (user driven)
- Same set of interfaces as drivers available

Defining a UEFI Application

UEFI Application Usages

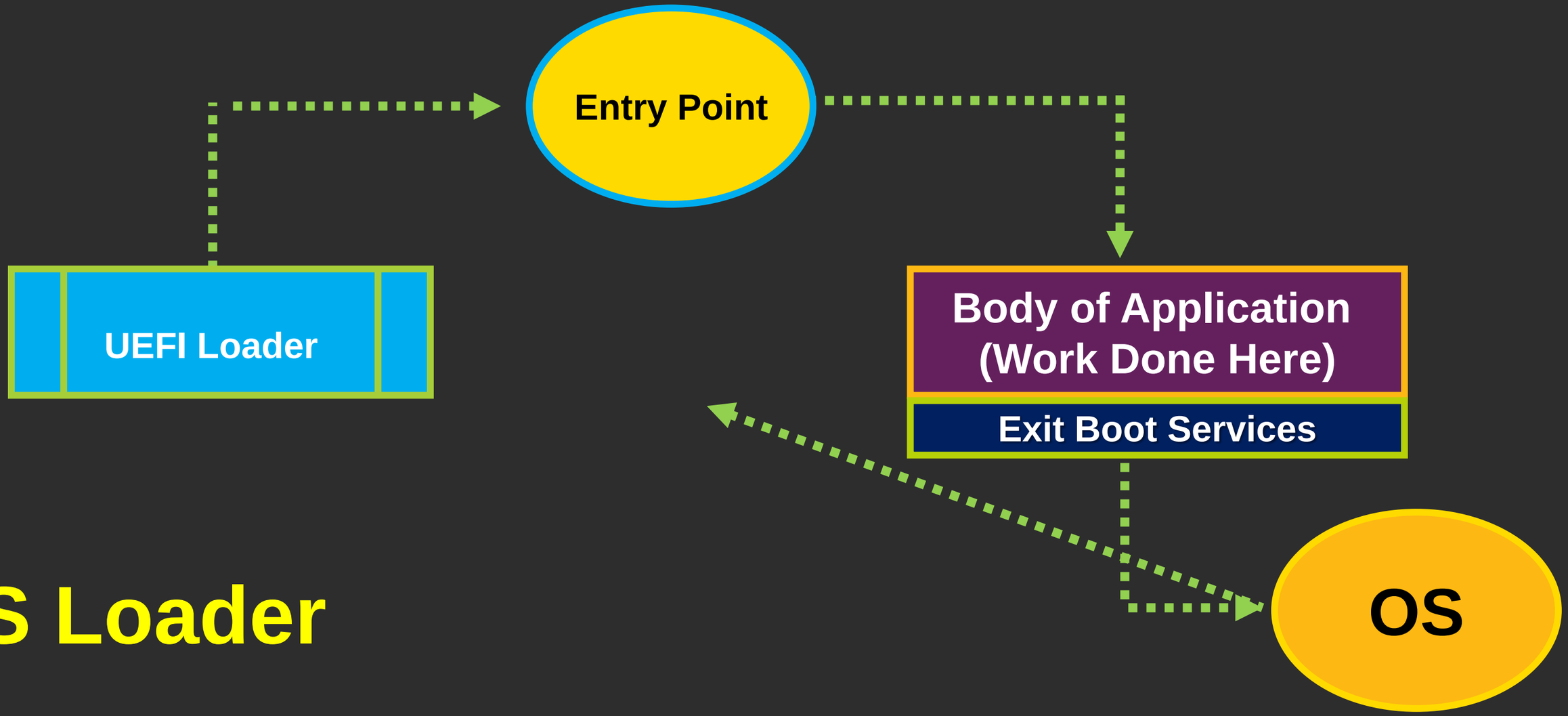
- Platform Diagnostics
- Factory Diagnostics
- Utilities
- Driver Prototyping
- “Platform” Applications
- Portable Across Platforms (IA32, X64, ARM, Itanium, etc.)

Executing Applications



Executing Applications

Executing Applications



OS Loader

Executing Applications

OS Loader

tianocore

Driver Vs. Application

	Driver	Application
Loaded by:	UEFI Loader	UEFI Loader
Interfaces available:	ALL	ALL
Consume protocols?	YES	YES
Produce protocols?	YES	NO
Typically driven by?	System	User
Typical use	Support Hardware	Any

EDK II UEFI APPLICATIONS

How to Write a EDK II UEFI Application

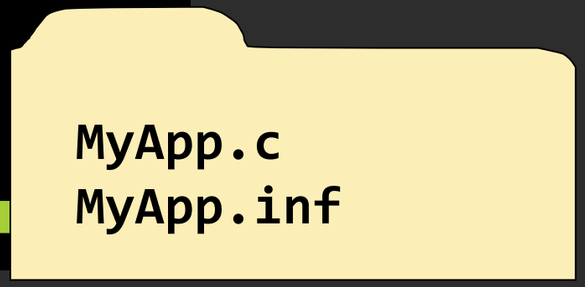
Application Files Placement

- Application source files can be located anywhere in the EDK II workspace including PACKAGES_PATH
- All code and include files go under a single directory containing the driver INF
- EDK II Sample Applications can be found here:

```
edk2/MdeModulePkg/Application
```

- Typically, modules reside within a package:

```
MyWorkSpace/
  edk2/
    MyPkg/
      Application/
        MyApp/
```



Module File [INF]

Syntax

```
INFfile ::=[<Header>]
           <Defines>
           [<BuildOptions>]
           [<Sources>]
           [<Binaries>]
           [<Guids>]
           [<Protocols>]
           [<Ppis>]
           [<Packages>]
           [<LibraryClasses>]
           [<Pcds>]
           [<UserExtensions>]
           [<Depex>]
```

PreMake

INF text file example

Application INF Files [DEFINES]

Field	Description
INF_VERSION	1.25* - Version of the INF spec.
BASE_NAME	What's the name of the application
FILE_GUID	Create a GUID for your module
MODULE_UNI_FILE	Meta-data - localization for Description & Abstract
VERSION_STRING	Version number
ENTRY_POINT	Name of the function to call
MODULE_TYPE	UEFI_APPLICATION

* EDK II Specifications: https://github.com/tianocore/tianocore.github.io/wiki/EDK-II-Specifications

Sample INF file

```
[Defines]
  INF_VERSION                    = 0x00010005
  BASE_NAME                      = MyApplication
  MODULE_UNI_FILE                = MyFile.uni
  FILE_GUID                      = 10C75C00-30 . . .
  MODULE_TYPE                    = UEFI_APPLICATION
  VERSION_STRING                 = 1.0
  ENTRY_POINT                    = UefiMain
[Sources]
  MyFile.c

[Packages]
  MdePkg/MdePkg.dec

[LibraryClasses]
  UefiApplicationEntryPoint

[Guids]

[Ppis]

[Protocols]
```

Sample INF file

```
[Defines]
  INF_VERSION                    = 0x00010005
  BASE_NAME                      = MyApplication
  MODULE_UNI_FILE                = MyFile.uni
  FILE_GUID                      = 10C75C00-30 . . .
  MODULE_TYPE                    = UEFI_APPLICATION
  VERSION_STRING                 = 1.0
  ENTRY_POINT                    = UefiMain

[Sources]
  MyFile.c

[Packages]
  MdePkg/MdePkg.dec

[LibraryClasses]
  UefiApplicationEntryPoint

[Guids]

[Ppis]
```

Building an Application

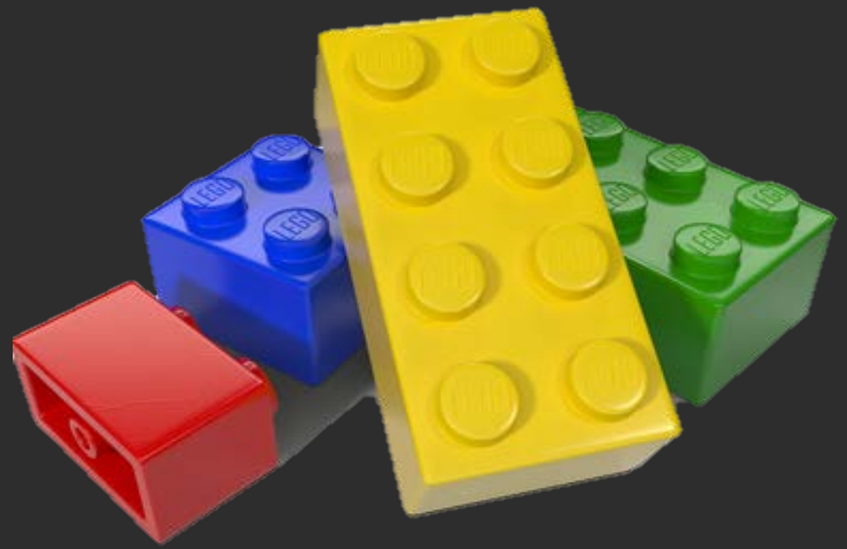
Platform .DSC references .INF

Runs:

"Build" for the entire platform

OR

"Build" in the application's directory



Sample Application ‘C’ file

```
#include <Uefi.h>
#include <Library/UefiApplicationEntryPoint.h>


EFI_STATUS
EFIAPI
UefiMain (
  IN EFI_HANDLE        ImageHandle,
  IN EFI_SYSTEM_TABLE  *SystemTable
  )
{
  return EFI_SUCCESS;
}
```

Sample Application ‘C’ file

```
#include <Uefi.h>
#include <Library/UefiApplicationEntryPoint.h>


EFI_STATUS
EFIAPI
UefiMain (
  IN EFI_HANDLE        ImageHandle,
  IN EFI_SYSTEM_TABLE  *SystemTable
  )
{
  return EFI_SUCCESS;
}
```

tianocore

EDK II UEFI DRIVERS

DXE Drivers, PEIM, Etc.

Driver Files Placement

- Driver source code can go anywhere in the EDK II workspace
- All code and include files go under a single directory containing an INF
- Good example of UEFI Drivers can be found here:

```
edk2/MdeModulePkg/Bus/ScsiDiskDxe
```

- Typically, Driver modules reside within a package:

```
MyWorkSpace/
  edk2/
    MyPkg/
      Include/
      MyDriver/
```

```
MyDriver.c
MyDriver.h
MyDriver.inf
```

Driver INF Files: [DEFINES]

Field	Description
INF_VERSION	1.25* - Version of the INF spec.
BASE_NAME	What's the name of the driver
FILE_GUID	Create a GUID for your module
MODULE_UNI_FILE	Meta-data - localization for Description & Abstract
VERSION_STRING	Version number
ENTRY_POINT	Name of the function to call
MODULE_TYPE	UEFI_DRIVER, DXE_DRIVER, PEIM, or others

Changes for a UEFI Driver Module

Applications can be converted to a driver

But …It remains in memory after it runs

UEFI Driver Module requirements:

- Driver Binding Protocol
- Component Name2 Protocol (recommended

DXE/PEIM/other Driver requirements

Sample Driver INF file

```
[Defines]
  INF_VERSION                    = 0x00010005
  BASE_NAME                      = MyDriver
  FILE_GUID                      = 10C75C00-30 . . .
  MODULE_TYPE                    = UEFI_DRIVER
  VERSION_STRING                 = 1.0
  ENTRY_POINT                    = UefiMain
[Sources]
  MyDriverFile.c

[Packages]
  MdePkg/MdePkg.dec

[LibraryClasses]
  UefiDriverEntryPoint
     . . .

[Guids]
     . . .
[Protocols]
     . . .
```

INF Usage Fields – DIST files

Optional **UEFI Spec – Package Distribution**

Usage Key Word

```
## UNDEFINED
## CONSUMES
## SOMETIMES_CONSUMES
## PRODUCES
## SOMETIMES_PRODUCES
## TO_START
## BY_START
## NOTIFY
```

Usage Fields used by Build tools for creating the .Dist files for binary modules

[GUID]

[PCD]

[PROTOCOL]

[PPIS]

1 Usage Block – “##” After the entry

n Usage Blocks – “##” Precede the entry

} UEFI Protocol

INF File Usage Block examples

```
[Guids]
  ## SOMETIMES_PRODUCES ## Variable:L"ConInDev"
  ## SOMETIMES_CONSUMES ## Variable:L"ConInDev"
  ## SOMETIMES_PRODUCES ## Variable:L"ConOutDev"
  ## SOMETIMES_CONSUMES ## Variable:L"ConOutDev“
  ## SOMETIMES_PRODUCES ## Variable:L"ErrOutDev"
  ## SOMETIMES_CONSUMES ## Variable:L"ErrOutDev"
  gEfiGlobalVariableGuid
  gEfiVTUTF8Guid                          ## SOMETIMES_CONSUMES ## GUID # used with a Vendor-Defined
  gEfiVT100Guid                           ## SOMETIMES_CONSUMES ## GUID # used with a Vendor-Defined
  gEfiVT100PlusGuid                       ## SOMETIMES_CONSUMES ## GUID # used with a Vendor-Defined
  gEfiPcAnsiGuid                          ## SOMETIMES_CONSUMES ## GUID # used with a Vendor-Defined
  gEfiTtyTermGuid                         ## SOMETIMES_CONSUMES ## GUID # used with a Vendor-Defined
  gEdkiiStatusCodeDataTypeVariableGuid ## SOMETIMES_CONSUMES ## GUID
```

Example: TerminalDxe.inf

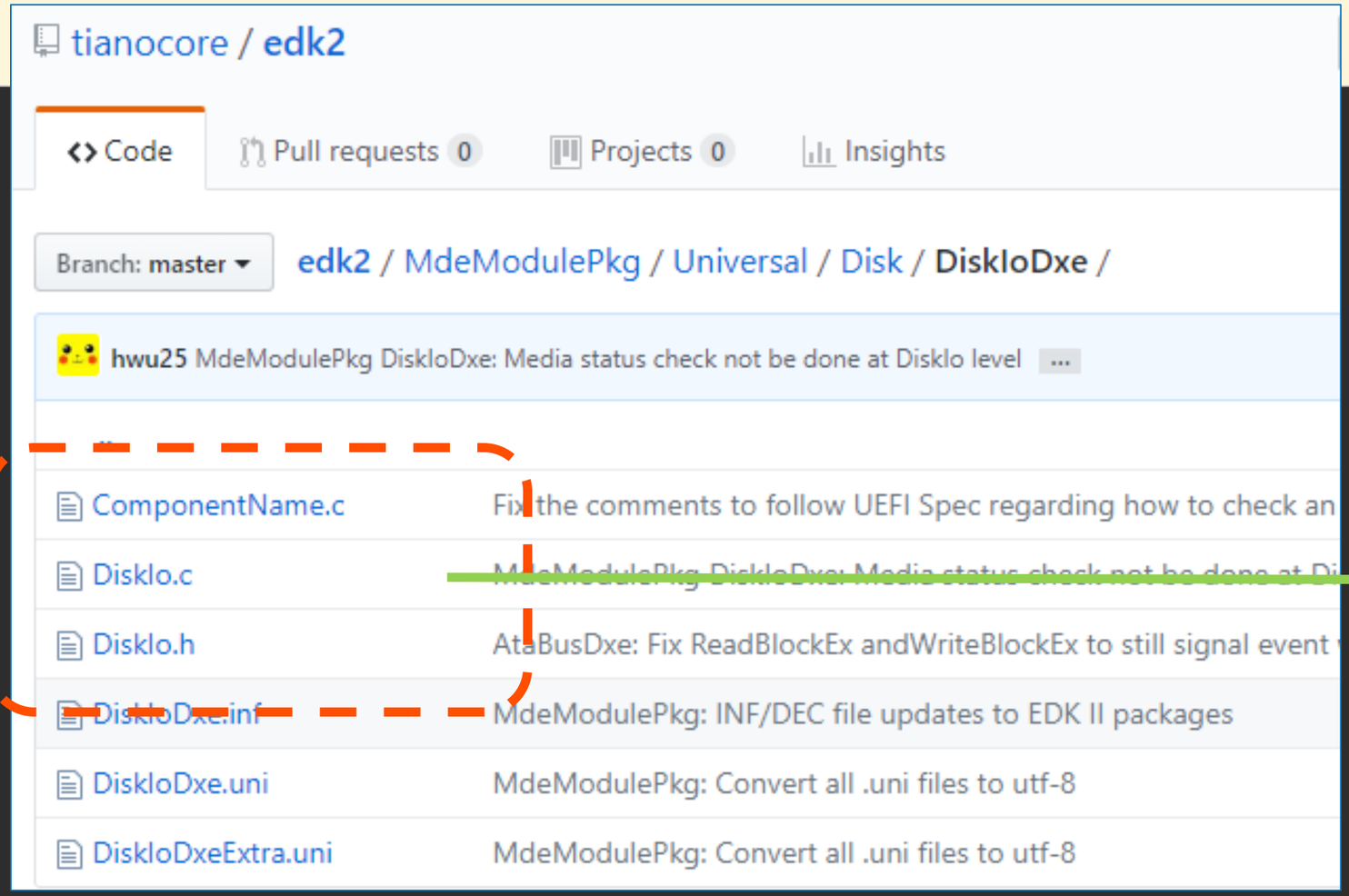
INF File Usage Block examples

```
[Protocols]
  gEfiSerialIoProtocolGuid              ## TO_START
  ## BY_START
  ## TO_START
  gEfiDevicePathProtocolGuid
  gEfiSimpleTextInProtocolGuid          ## BY_START
  gEfiSimpleTextInputExProtocolGuid     ## BY_START
  gEfiSimpleTextOutProtocolGuid         ## BY_START

[Pcd]
  gEfiMdePkgTokenSpaceGuid.PcdDefaultTerminalType           ## SOMETIMES_CONSUMES
  gEfiMdeModulePkgTokenSpaceGuid.PcdErrorCodeSetVariable    ## CONSUMES
```

Example: TerminalDxe.inf

UEFI Driver Example – Disk I/O

https://github.com/tianocore/edk2/tree/master/MdeModulePkg/Universal/Disk/DiskIoDxe



Driver Binding
- Supported
- Start
- Stop

UEFI Driver Example – Disk I/O

github.com/tianocore/edk2/.../Disk/DiskIoDxe

Entry Point

"C" File

```
EFI_STATUS
EFIAPI
InitializeDiskIo (
  IN EFI_HANDLE           ImageHandle,
  IN EFI_SYSTEM_TABLE     *SystemTable
  )
{
  // . . .
  Status = EfiLibInstallDriverBindingComponentName2 (
             ImageHandle,
             SystemTable,
             &gDiskIoDriverBinding,
             ImageHandle,
             &gDiskIoComponentName,
             &gDiskIoComponentName2
             );
  ASSERT_EFI_ERROR (Status);
  return Status;
}
```

INF File

```
[Defines]
  . . .
ENTRY_POINT     = InitializeDiskIo
```

UEFI Driver Example – Disk I/O

github.com/tianocore/edk2/.../Disk/DiskIoDxe

Supported

"C" File

```
EFI_STATUS
EFIAPI
DiskIoDriverBindingSupported (
  IN EFI_DRIVER_BINDING_PROTOCOL  *This,
  IN EFI_HANDLE                   ControllerHandle,
  IN EFI_DEVICE_PATH_PROTOCOL     *RemainingDevicePath
OPTIONAL
  )
{
  Status = gBS->OpenProtocol (
    ControllerHandle,
      &gEfiBlockIoProtocolGuid,
      (VOID **) &BlockIo,
      This->DriverBindingHandle,
      ControllerHandle,
      EFI_OPEN_PROTOCOL_BY_DRIVER
    );
```

INF File

```
[Protocols]
  . . .
gEfiBlockIoProtocolGuid   ## TO_START
```

UEFI Driver Example – Disk I/O

github.com/tianocore/edk2/.../Disk/DiskIoDxe

Start

"C" File

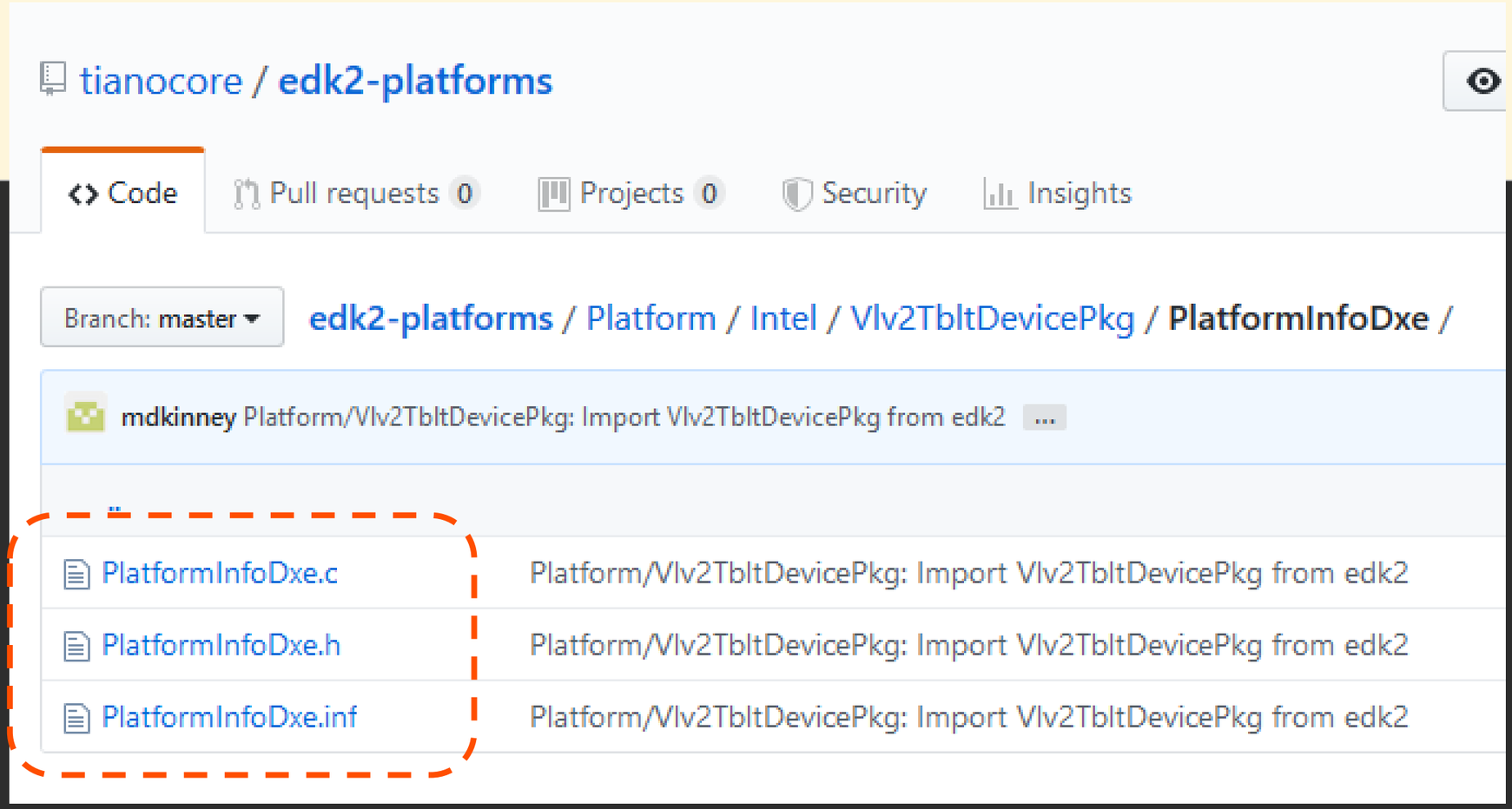
```
EFI_STATUS
EFIAPI
 DiskIoDriverBindingStart (
  IN EFI_DRIVER_BINDING_PROTOCOL  *This,
  IN EFI_HANDLE                   ControllerHandle,
  IN EFI_DEVICE_PATH_PROTOCOL     *RemainingDevicePath
OPTIONAL
  )
{

  if (Instance->BlockIo2 != NULL) {
    Status = gBS->InstallMultipleProtocolInterfaces (
    &ControllerHandle,
    &gEfiDiskIoProtocolGuid,  &Instance->DiskIo,
    &gEfiDiskIo2ProtocolGuid, &Instance->DiskIo2,
    NULL
    );
```

INF File

```
[Protocols]
  . . .
gEfiDiskIoProtocolGuid  ## BY_START
gEfiDiskIo2ProtocolGuid ## BY_START
```

DXE Driver Example - PlatformInfoDxe

https://github.com/tianocore/edk2-platforms/. . ./Vlv2TbltDevicePkg/PlatformInfoDxe



DXE Driver Example – PlatformInfoDxe

https://github.com/tianocore/edk2-platforms/ PlatformInfoDxe

Entry Point

"C" File

```
#include "PlatformInfoDxe.h"
• • •
EFI_STATUS
EFIAPI
PlatformInfoInit (
  IN EFI_HANDLE         ImageHandle,
  IN EFI_SYSTEM_TABLE   *SystemTable
  )
/*++
{
// • • •
  return Status;
}
```

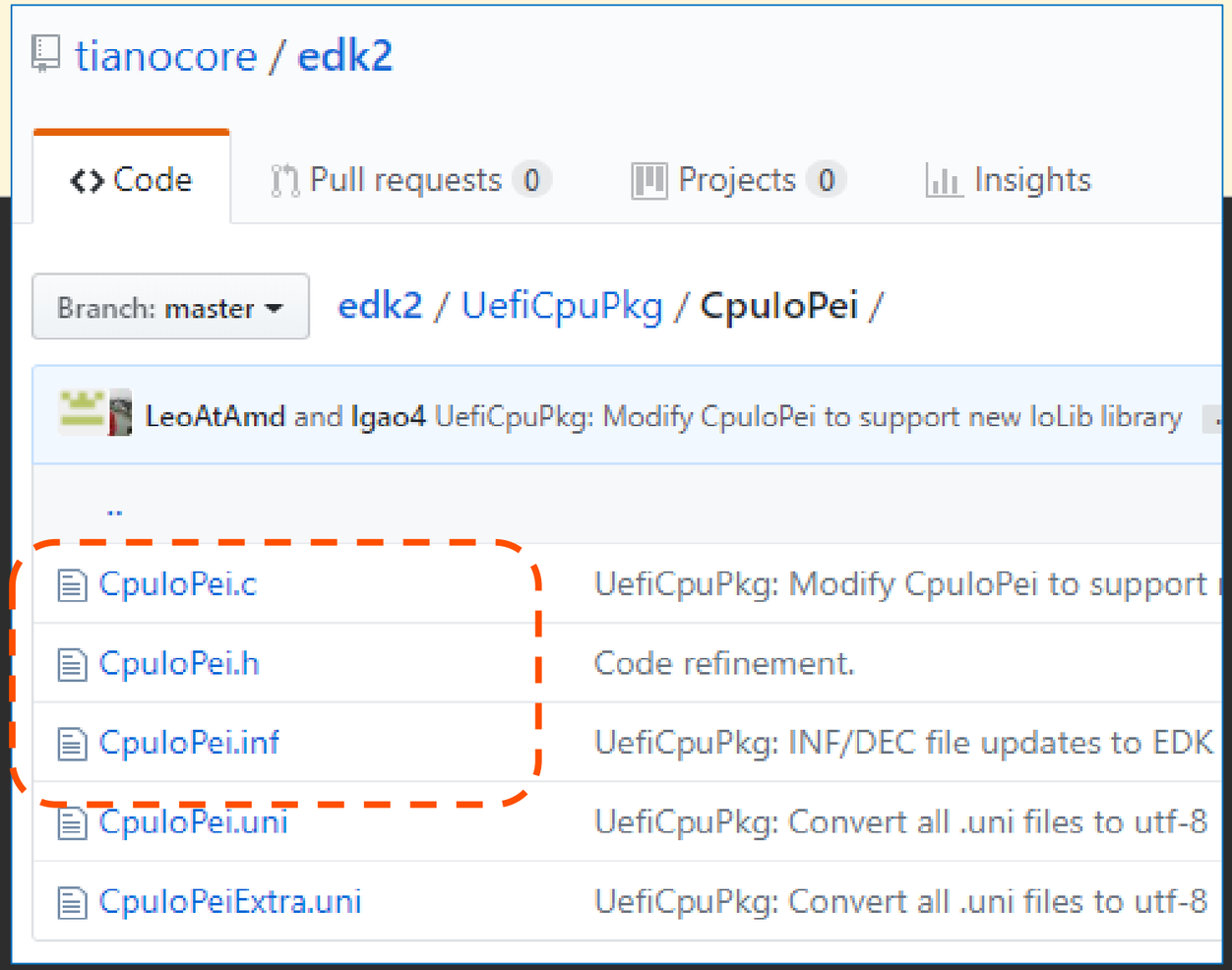
INF File

```
[Defines]
 • • •
  MODULE_TYPE      = DXE_DRIVER
  VERSION_STRING   = 1.0
  ENTRY_POINT      = PlatformInfoInit
 • • •

[Depex]
  gEfiVariableArchProtocolGuid AND
   gEfiVariableWriteArchProtocolGuid
```

Notice the MODULE TYPE, C function Entry point and the [Depex] differences in the INF file

PEI Driver (PEIM) Example - CpuIoPei

https://github.com/tianocore/edk2/tree/master/UefiCpuPkg/CpuIoPei



PEI Driver (PEIM) Example – CpuIoPei

github.com/tianocore/edk2/UefiCpuPkg/CpuIoPei

Entry Point

"C" File

```
#include "CpuIoPei.h"
 //• • •
EFI_STATUS
EFIAPI
CpuIoInitialize (
  IN EFI_PEI_FILE_HANDLE     FileHandle,
  IN CONST EFI_PEI_SERVICES  **PeiServices
  )
{
  EFI_STATUS  Status;
 // • • •
  return EFI_SUCCESS;
}
```

INF File

```
[Defines]
 • • •

  MODULE_TYPE        = PEIM
  VERSION_STRING     = 1.0
  ENTRY_POINT        = CpuIoInitialize
 • • •

[Depex]
   TRUE
```

SUMMARY

- What is a EDK II Module
- Use EDK II libraries to write UEFI apps/drivers
- How to Define a UEFI application
- Differences between UEFI App / Drivers INF file

Questions?

Return to Main Training Page

Return to Training Table of contents for next presentation link

tianocore

ACKNOWLEDGEMENTS

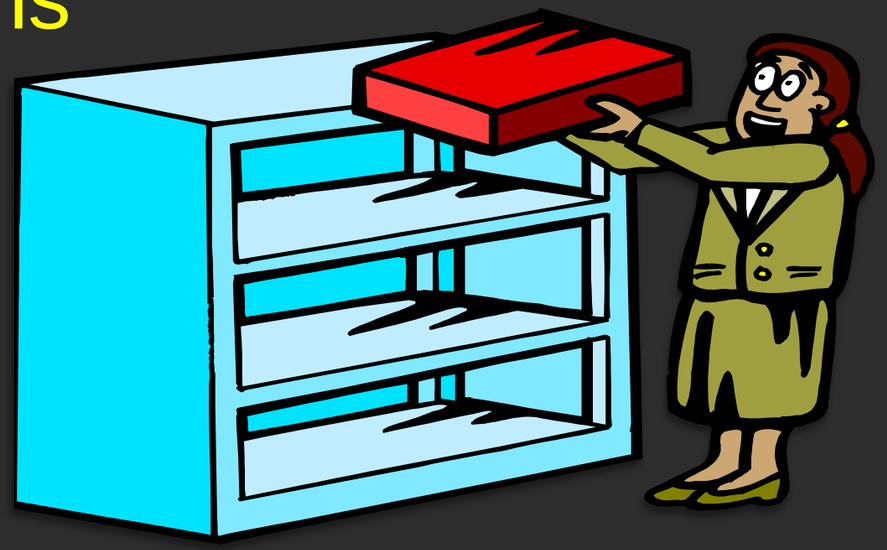
tianocore

BACK UP

UEFI Application Vs. EADK Application

EDK II Application Development
Kit includes the Standard C
Libraries in UEFI Shell Applications

Off the shelf “C” application
Converted to UEFI application



Sample INF file using EDK II EADK

```
[Defines]
  INF_VERSION                    = 0x00010005
  BASE_NAME                      = MyApplication
  FILE_GUID                      = 10C75C00-30 . . .
  MODULE_TYPE                    = UEFI_APPLICATION
  VERSION_STRING                 = 1.0
  ENTRY_POINT                    = ShellCEntryLib
[Sources]
  MyFile.c

[Packages]
  StdLib/StdLib.dec
  ShellPkg/ShellPkg.dec
  MdePkg/MdePkg.dec

[LibraryClasses]
  LibC
  LibStdio
```

Sample INF file using EDK II EADK

```
[Defines]
  INF_VERSION                    = 0x00010005
  BASE_NAME                      = MyApplication
  FILE_GUID                      = 10C75C00-30 . . .
  MODULE_TYPE                    = UEFI_APPLICATION
  VERSION_STRING                 = 1.0
  ENTRY_POINT                    = ShellCEntryLib
[Sources]
  MyFile.c

[Packages]
  StdLib/StdLib.dec
  ShellPkg/ShellPkg.dec
  MdePkg/MdePkg.dec

[LibraryClasses]
  LibC
  LibStdio
```

Sample Application ‘C’ file Using EDK II EADK

This sample looks a lot like actual “C” source.

```
#include <stdio.h>

int
Main
     int Argc,
     char **Argv
  )
{
  return 0
}
```

More on “NULL” named Library

So, “NULL” library classes are conceptually an "anonymous library". It enables one to statically link code into a module even if the module doesn't directly call functions in that library. All libraries, both regular libraries with a declared LibraryClass as well as these anonymous libraries, can publish both a constructor and a destructor. The EDK II build system will automatically generate a small amount of C code that invokes all library constructors before the entry point for the module is invoked, and all destructors after the entry point returns. This is useful for building statically linked plug-ins.

The favorite example of this in-action is the UEFI Shell, here is what a typical DSC declaration for the UEFI Shell looks like:

```
ShellPkg/Application/Shell/Shell.inf {
  <PcdsFixedAtBuild>
    gEfiShellPkgTokenSpaceGuid.PcdShellLibAutoInitialize|FALSE
  <LibraryClasses>
    NULL|ShellPkg/Library/UefiShellLevel1CommandsLib/UefiShellLevel1CommandsLib.inf
    NULL|ShellPkg/Library/UefiShellLevel2CommandsLib/UefiShellLevel2CommandsLib.inf
    NULL|ShellPkg/Library/UefiShellLevel3CommandsLib/UefiShellLevel3CommandsLib.inf
    NULL|ShellPkg/Library/UefiShellDriver1CommandsLib/UefiShellDriver1CommandsLib.inf
    NULL|ShellPkg/Library/UefiShellInstall1CommandsLib/UefiShellInstall1CommandsLib.inf
    NULL|ShellPkg/Library/UefiShellDebug1CommandsLib/UefiShellDebug1CommandsLib.inf
    NULL|ShellPkg/Library/UefiShellNetwork1CommandsLib/UefiShellNetwork1CommandsLib.inf
    NULL|ShellPkg/Library/UefiShellNetwork2CommandsLib/UefiShellNetwork2CommandsLib.inf
    ShellCommandLib|ShellPkg/Library/UefiShellCommandLib/UefiShellCommandLib.inf
    HandleParsingLib|ShellPkg/Library/UefiHandleParsingLib/UefiHandleParsingLib.inf
    BcfgCommandLib|ShellPkg/Library/UefiShellBcfgCommandLib/UefiShellBcfgCommandLib.inf
    ShellCEntryLib|ShellPkg/Library/UefiShellCEntryLib/UefiShellCEntryLib.inf
    ShellLib|ShellPkg/Library/UefiShellLib/UefiShellLib.inf
}
```

More on “NULL” named Library Cont.

If you take a look at ShellPkg/Library/UefiShellLevel1CommandsLib/UefiShellLevel1CommandsLib.inf, you will see that the constructor for that anonymous library is named ShellLevel1CommandsLibConstructor(). Now, let's go and look at the definition for ShellLevel1CommandsLibConstructor() in ShellPkg/Library/UefiShellLevel1CommandsLib/UefiShellLevel1CommandsLib.c and note the following code snippet:

```
ShellCommandRegisterCommandName(L"stall",  ShellCommandRunStall   ...
ShellCommandRegisterCommandName(L"for",    ShellCommandRunFor     ...
ShellCommandRegisterCommandName(L"goto",   ShellCommandRunGoto    ...
ShellCommandRegisterCommandName(L"if",     ShellCommandRunIf      ...
ShellCommandRegisterCommandName(L"shift",  ShellCommandRunShift   ...
ShellCommandRegisterCommandName(L"exit",   ShellCommandRunExit    ...
ShellCommandRegisterCommandName(L"else",   ShellCommandRunElse    ...
ShellCommandRegisterCommandName(L"endif",  ShellCommandRunEndIf   ...
ShellCommandRegisterCommandName(L"endfor", ShellCommandRunEndFor  ...
```

This library is installing new commands into the UEFI shell during its initialization procedure. This allows one to add custom commands to the shell as statically linked built-ins. A typical use case would be implementing platform specific diagnostic/recovery utilities.