

UEFI & EDK II TRAINING

EDK II BUILD SPECIFICATION FILES LAB

See also [Lab Guide.md](#) for Copy & Paste examples in labs

tianocore.org

LESSON OBJECTIVE

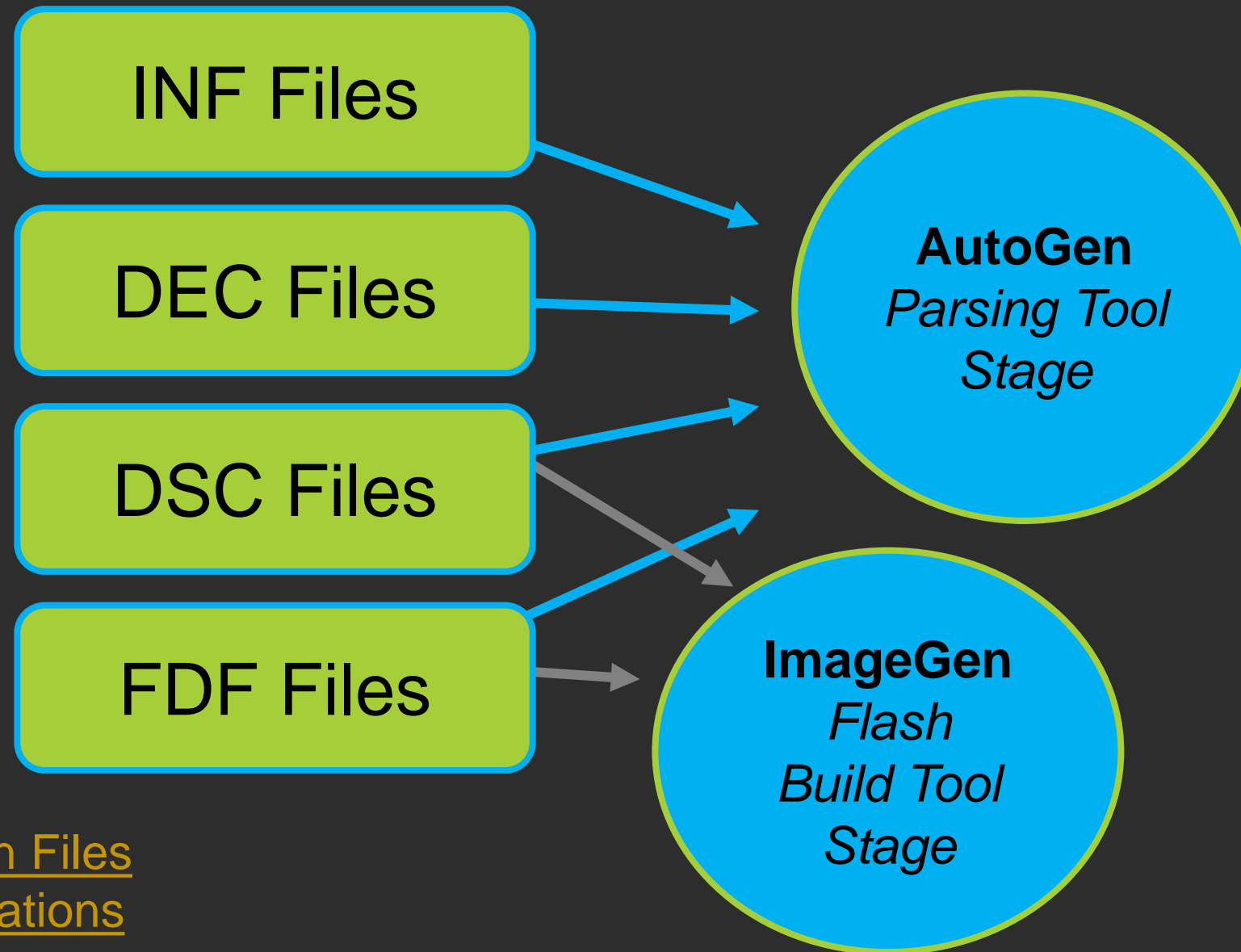
- ★ Examine the Build components and build text files DSC, DEC, & FDF

EDK II BUILD TEXT FILES

EDK II tools use INI-style text-based files to describe components, platforms and firmware volumes.

Build Description File Types

**EDK II
Spec**



Wiki Link: [Build Description Files
Edk II Specifications](https://wiki.tianocore.org/BuildDescriptionFiles/EdkIISpecifications)

General Format for All Build Text Files

INI

- The EDK II Build Text Files use meta-data files using the INI format style

Section “[]”

- All Build text files consists of sections delineated by section tags enclosed within Square “[” “]” brackets

Case

- Section tag entries are case-insensitive

Mult-Sections

- Text of a given section can be used for multiple section names by separating the section names with a comma

Section End

- Sections are terminated by the start of another section or the end of the file.

Comments

- The hash-tag “#” indicates text following to EOL is a comment (exception is within a quoted string)

Include

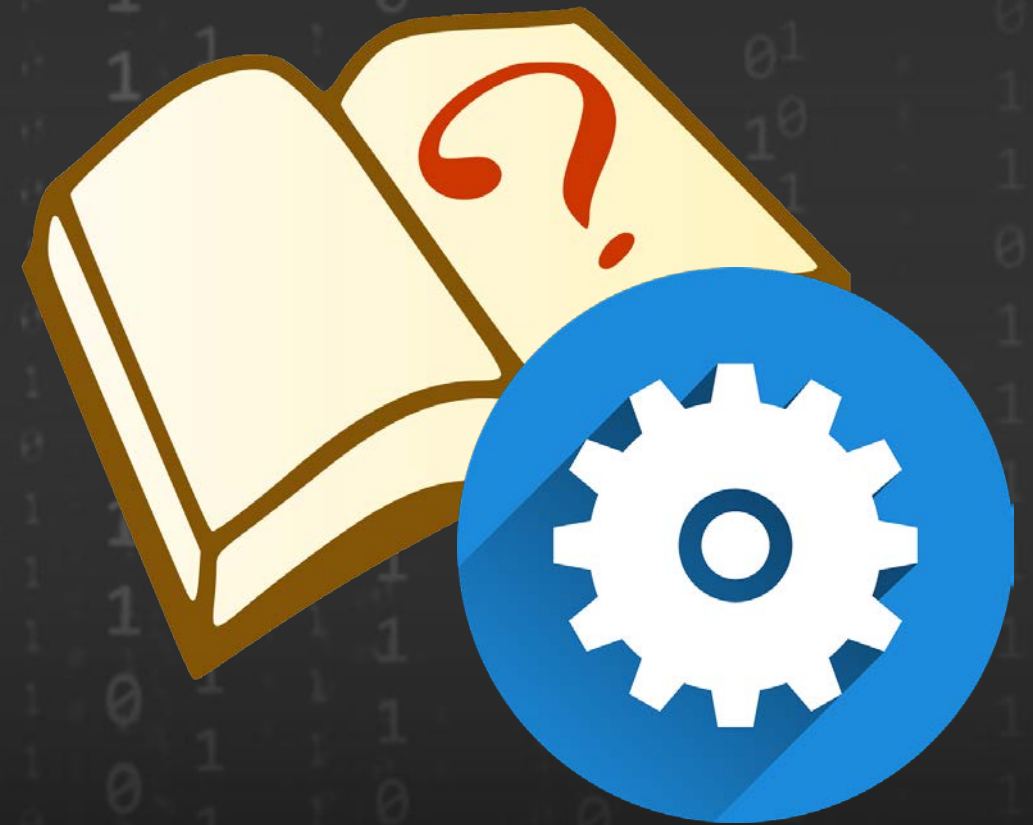
- The “!include” statements are permitted in .DSC and .FDF but NOT .DEC

Conditional

- Condition Statements Supported in .DSC and .FDF but NOT .DEC
- !ifdef, !ifndef, !if, !elseif, !else and !endif

Lab 1: Examine the DEC, DSC and FDF files

In this lab, you'll learn about the layout of the DEC, DSC and FDF files.



Package Declaration File (DEC)

Syntax:

```
<DECfile> ::= <Defines>  
                Include  
                [<LibraryClass>]  
                [<Guids>]  
                [<Protocols>]  
                [<Ppis>]  
                [<Pcd>]  
                [<UserExtensions>]
```

Declare

Review the Wiki Explanation: <https://github.com/tianocore/tianocore.github.io/wiki/Build-Description-Files#the-dec-file>

Example DEC File

```
[Defines]
  DEC_SPECIFICATION      = 0x00010005
  PACKAGE_NAME           = OvmfPkg
  PACKAGE_GUID           = 2daf5f34-50e5-4b9d-b8e3-5562334d87e5
  PACKAGE_VERSION        = 0.1

[Includes]
  Include

[LibraryClasses]
  ## @libraryclass  Loads and boots a Linux kernel image
  #
  LoadLinuxLib|Include/Library/LoadLinuxLib.h

[Guids]
  gUefiOvmfPkgTokenSpaceGuid      = {0x93bb96af, 0xb9f2, 0x4eb8, {0x94, 0x62, 0xe0, 0xba, 0x74, 0x56, 0x42, 0x36}}
  gEfiXenInfoGuid                 = {0xd3b46f3b, 0xd441, 0x1244, {0x9a, 0x12, 0x0, 0x12, 0x27, 0x3f, 0xc1, 0x4d}}

[Protocols]
  gVirtioDeviceProtocolGuid       = {0xfa920010, 0x6785, 0x4941, {0xb6, 0xec, 0x49, 0x8c, 0x57, 0x9f, 0x16, 0x0a}}
  gXenBusProtocolGuid             = {0x3d3ca290, 0xb9a5, 0x11e3, {0xb7, 0x5d, 0xb8, 0xac, 0x6f, 0x7d, 0x65, 0xe6}}

[PcdsFixedAtBuild]
  gUefiOvmfPkgTokenSpaceGuid.PcdOvmfPeiMemFvBase|0x0|UINT32|0x00001014
  gUefiOvmfPkgTokenSpaceGuid.PcdOvmfPeiMemFvSize|0x0|UINT32|0x00001015
```

Tokens need to be unique
to the DEC file (1 per PCD)

Examine the Dec File Details

Follow the following Links and examine the examples of the EmulatorPkg.dec file

Next open the same EmulatorPkg.dec in the %WORKSPACE% and become familiar with the different sections

[EmulatorPkg.dec.md#dec-file-for-emulatorpkg](#)

[Link](#): List of List of Defines, Package Name, GUILD, Version ...

[Link](#): The Include section

[Link](#): Library classes section

[Link](#): Protocols Section

[Link](#): GUIDs section

[Link](#): PCDs Section

[Link](#): Patchable PCDs Section

Platform Description File (DSC)

Syntax:

```
DSCfile ::= [<Header>]
           <Defines>
           [<SkuIds>]
           [<Libraries>]
           [<LibraryClasses>]
           [<Pcds>]
           [<Components>]
           [<UserExtensions>]
```

Description

Review the Wiki Explanation: <https://github.com/tianocore/tianocore.github.io/wiki/Build-Description-Files#the-dsc-file>

Platform Description File (DSC)

DSC file is the recipe for creating a package

Definitions for the package build

EDK II Library Class Instance Mappings (for EDK II Modules)

EDK II PCD Entry Settings

Components / Modules to build (list of .inf files)

DSC file must define all libraries, components and/or modules that will be used by one package

Example: DSC File

```
[Defines]
PLATFORM_NAME                = Ovmf
PLATFORM_GUID                = 5a9e7754-d81b-49ea-85ad-69eaa7b1539b
PLATFORM_VERSION              = 0.1
DSC_SPECIFICATION             = 0x00010005
OUTPUT_DIRECTORY             = Build/OvmfX64
SUPPORTED_ARCHITECTURES      = X64
BUILD_TARGETS                 = NOOPT|DEBUG|RELEASE
SKUID_IDENTIFIER              = DEFAULT
FLASH_DEFINITION              = OvmfPkg/OvmfPkgX64.fdf

#
# Defines for default states.  These can be changed on the command line.
# -D FLAG=VALUE
. . .
[BuildOptions.common.EDKII.DXE_RUNTIME_DRIVER]
GCC:*_*_*_DLINK_FLAGS = -z common-page-size=0x1000
XCODE:*_*_*_DLINK_FLAGS =
[LibraryClasses]
PcdLib|MdePkg/Library/BasePcdLibNull/BasePcdLibNull.inf
TimerLib|OvmfPkg/Library/AcpiTimerLib/BaseAcpiTimerLib.inf
```

```
. . .
#####
# Pcd Section
#####
. . .
#####
#
# Components Section - list of all
# EDK II Modules needed by this
# Platform.
#
#####
[Components]

OvmfPkg/ResetVector/ResetVector.inf
. . .
```

DSC must contain a
[Components] Section

Examine : DSC File Details

Follow the following Links and examine the examples of the EmulatorPkg.dsc file

Next open the same EmulatorPkg.dsc in the %WORKSPACE% and become familiar with the different sections

[EmulatorPkg.dsc.md#dsc-file-for-emulatorpkg](#)

[Link](#): List of Defines

[Link](#): Define Switches to determine some configurations

[Link](#): Library Classes – Global

[Link](#): Library Classes for UEFI Boot phases

[Link](#): PCDs Section, changing the default

[Link](#): Dynamic PCDs Section

[Link](#): Components Section

[Link](#): Build Options Section

[Link](#): Adding More

Flash Description File(FDF)

Syntax:

```
FDFfile ::= [<Header>]
           [<Defines>]
           <FD>
           <FV>
           [<Capsule>]
           [<VTF>]
           [<Rules>]
           [<OptionRom>]
           [<UserExtensions>]
```

Flash Layout

Must have a FD (Flash Device) and FV (Firmware Volume) Section

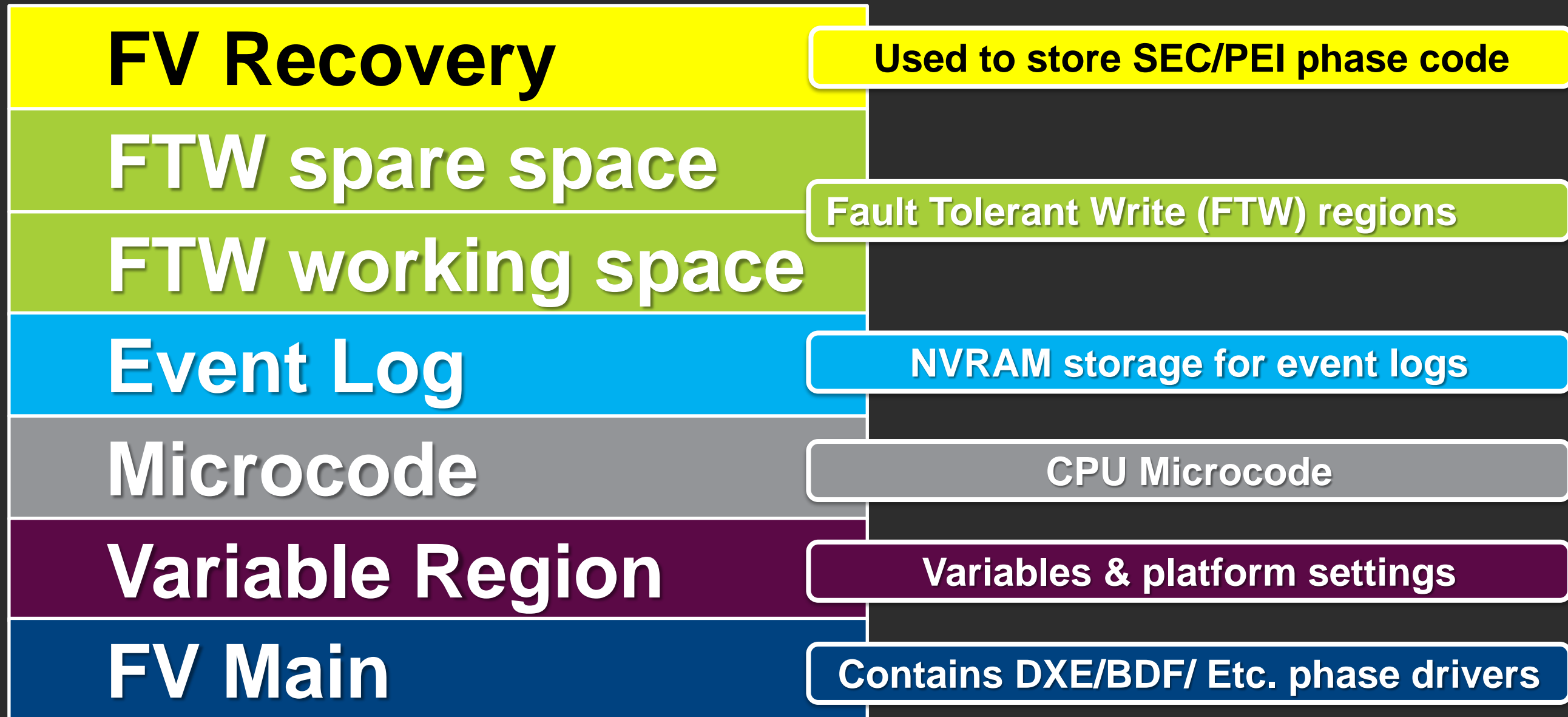
Flash Description File(FDF)

Describes information about flash parts

Used to create firmware images, Option
ROM images or bootable images

Rules for combining binaries (Firmware
Image) built from a DSC file

FLASH DEVICE CONFIGURATION COMMON LAYOUT FILE (.FDF)



Example: FDF File

```
[Defines]
!include OvmfPkg.fdf.inc

#
# Build the variable store and the firmware code
# as one unified flash device image.
```

```
#
[FD.OVMF]
BaseAddress = $(FW_BASE_ADDRESS)
Size        = $(FW_SIZE)
ErasePolarity = 1
BlockSize  = $(BLOCK_SIZE)
NumBlocks  = $(FW_BLOCKS)
```

Ovmf.fdf file
created by
Build

Offset | Size
\$(VARS_SIZE)|\$(FVMAIN_SIZE)
FV = FVMAIN_COMPACT

Firmware
Volumes
created by
Build

Offset | Size
\$(SECFV_OFFSET)|\$(SECFV_SIZE)
FV = SECFV

Included Mapping file

```
DEFINE BLOCK_SIZE      = 0x1000
DEFINE VARS_OFFSET     = 0

!if ($(FD_SIZE_IN_KB) == 1024) || ($(FD_SIZE_IN_KB) == 2048)
DEFINE VARS_SIZE       = 0x20000
DEFINE VARS_BLOCKS     = 0x20
DEFINE VARS_LIVE_SIZE  = 0xE000
DEFINE VARS_SPARE_SIZE = 0x10000
!endif
# . . .

SET gUefiOvmfPkgTokenSpaceGuid.PcdOvmfFdBaseAddress =
$(FW_BASE_ADDRESS)
SET gUefiOvmfPkgTokenSpaceGuid.PcdOvmfFirmwareFdSize =
$(FW_SIZE)
SET gUefiOvmfPkgTokenSpaceGuid.PcdOvmfFirmwareBlockSize =
$(BLOCK_SIZE)

SET gUefiOvmfPkgTokenSpaceGuid.PcdOvmfFlashNvStorageVariableBase =
$(FW_BASE_ADDRESS)
SET gEfiMdeModulePkgTokenSpaceGuid.PcdFlashNvStorageVariableSize =
$(VARS_LIVE_SIZE)
```

Examine : FDF File Details

Follow the following Links and examine the examples of the EmulatorPkg.fdf file

Next open the same EmulatorPkg.fdf in the %WORKSPACE% and become familiar with the different sections

[EmulatorPkg.fdf.md#fdf-file-for-the-emulatorpkg](#)

[Link](#): FD Section

[Link](#): Firmware Volume – FvRecovery

[Link](#): Begin Firmware Layout Regions

[Link](#): Declaring each Firmware Volumes

[Link](#): Apriori Section

[Link](#): Example: #include of fdf file

[Link](#): Rules Section

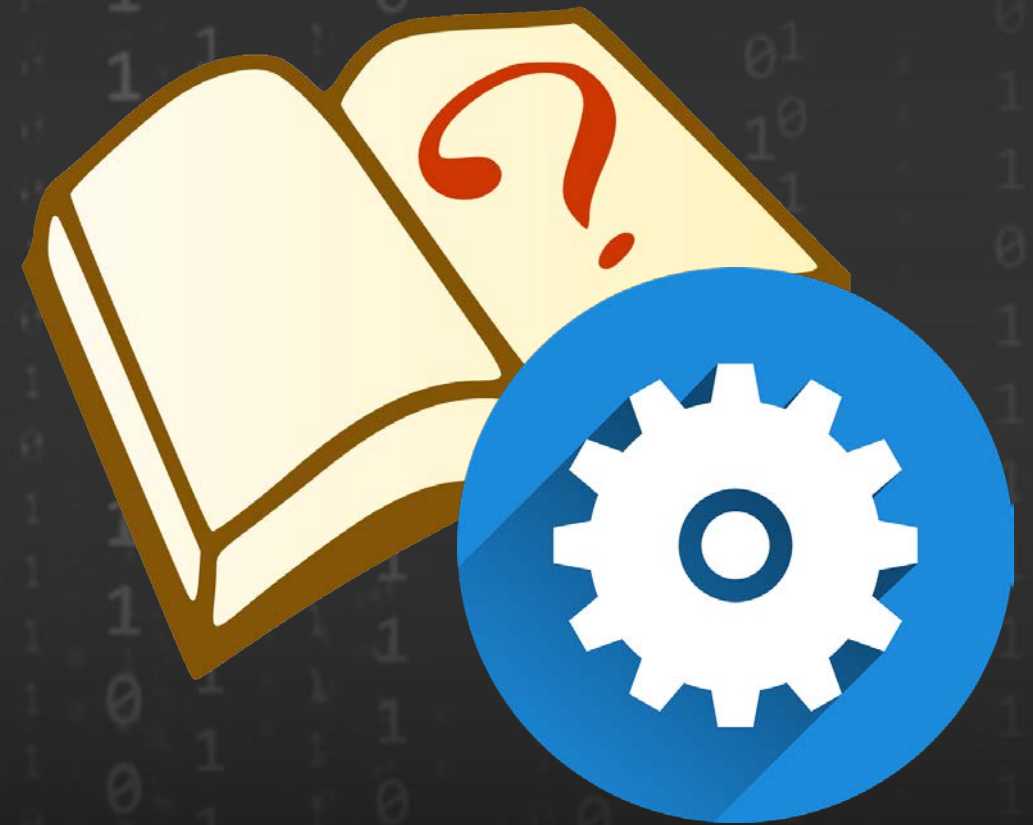
Following are for the Whiskey Lake UPX (these examples will be used in later projects

[Link](#): FDF For Whiskey Lake Up Xtreme

[Link](#): Flash Map of Up Xtreme

Lab 2: Add a Simple Driver to the Build

In this lab, you'll learn how to add a UEFI Driver to the Build and final image .FD file.



Add a UEFI Driver to a Platform

Requirements:

Add a simple UEFI driver to a platform based on a Macro switch passed to the build using “-D ADD_BLANKDRV”

Requires Building the Platform Lab first

- Windows Build Emulator Platform Lab [Link](#)
- Linux Build Ovmf Platform Lab [Link](#)

Add a UEFI Driver to a Platform

Windows

1. Copy the LabSampleCode/LabSolutions/BlankDrv directory to C:/FW/edk2-ws/edk2
2. Edit EmulatorPkg.dsc and add the BlankDrv component at the end and use a “if” statement based on macro ADD_BLANKDRV
3. Edit EmulatorPkg.fdf and add the BlankDrv Driver to the DXE section of Firmware Volumes and use a “if” statement based on macro ADD_BLANKDRV

```
C:/FW/edk2-ws/edk2> Build -D ADD_BLANKDRV
C:/FW/edk2-ws/edk2> RunEmulator.bat
```

Linux

1. Copy the LabSampleCode/LabSolutions/BlankDrv directory to ~/src/edk2-ws/edk2
2. Edit OvmfPkgX64.dsc and add the BlankDrv component at the end and use a “if” statement based on macro ADD_BLANKDRV
3. Edit OvmfPkgX64.fdf and add the BlankDrv Driver to the DXE section of Firmware Volumes and use a “if” statement based on macro ADD_BLANKDRV

```
bash$> cd ~/src/edk-ws/edk2
bash$> build -D ADD_BLANKDRV -a X64
bash$ cd $HOME/run-ovmf
bash$ cp ~/src/edk2-
ws/Build/OvmfX64/DEBUG_GCC5/FV/OVMF.fd bios.bin
bash$ . RunQemu.sh
```

Verify the BlankDrv Driver was Added

At the Shell Prompt:

Shell> Exit

This will open the BIOS Setup

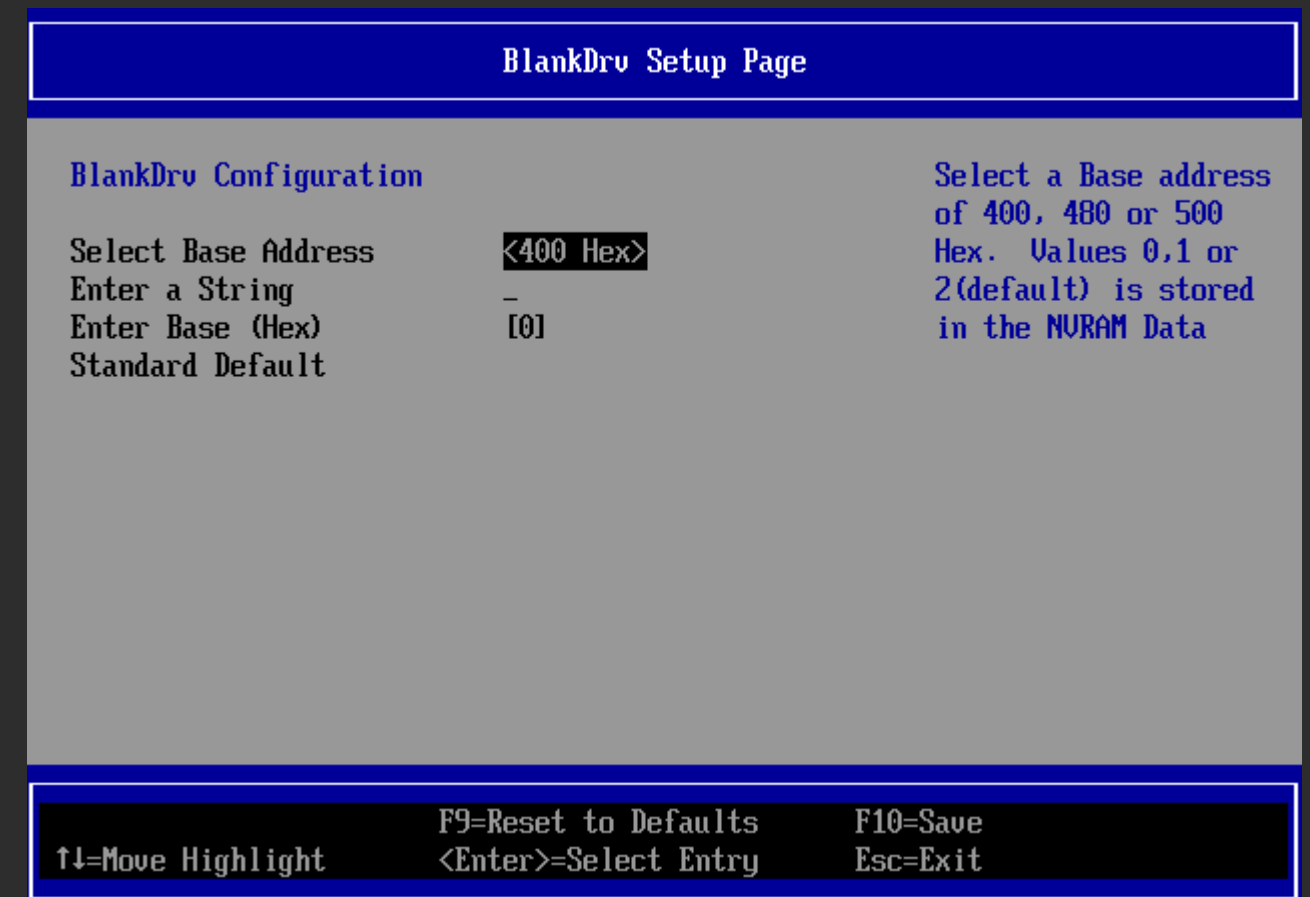
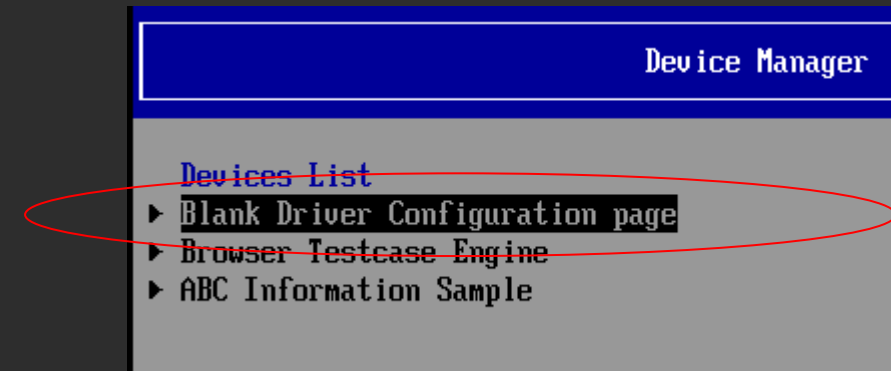
Go to the "Device Manager" menu and
Verify the "Blank Driver Configuration
page" is available

Enter into the BlankDrv Setup Page

Exit Emulation

Windows: ESC key twice then use the "Reset"

Linux: Exit QEMU Linux



Summary

- ★ Examine the Build components and build text files DSC, DEC, & FDF

Questions?



Return to Main Training Page



Return to Training Table of contents for next presentation [link](#)



ACKNOWLEDGEMENTS

Redistribution and use in source (original document form) and 'compiled' forms (converted to PDF, epub, HTML and other formats) with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code (original document form) must retain the above copyright notice, this list of conditions and the following disclaimer as the first lines of this file unmodified.

Redistributions in compiled form (transformed to other DTDs, converted to PDF, epub, HTML and other formats) must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS DOCUMENTATION IS PROVIDED BY TIANOCORE PROJECT "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL TIANOCORE PROJECT BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (c) 2021-2022, Intel Corporation. All rights reserved.