

UEFI & EDK II Training

How to Write a UEFI Application w/ Windows Lab
- Simics

tianocore.org

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

Lesson Objective

First Setup for Building EDK II, See [Lab Setup](#) then [Platform Build Lab for Simics](#)

- ★ UEFI Application with PCDs
- ★ Simple UEFI Application
- ★ Add functionality to UEFI Application
- ★ Using EADK with UEFI Application (Optional)

UEFI APPLICATION W/ PCDS

 Documentaton : [MdeModulePkg/Universal/PCD/Dxe/Pcd.inf](https://github.com/tianocore/edk2/blob/master/MdeModulePkg/Universal/PCD/Dxe/Pcd.inf)

Purpose

- Establishes platform common definitions
- Build-time/Run-time aspects
- Binary Editing Capabilities

Goals

- Simplify porting
- Easy to associate with a module or platform

PCDs can be located anywhere within the Workspace even though a different package will use those PCDs for a given project

.DEC

**Define
PCD**

Package

.INF

**Reference
PCD**

Module

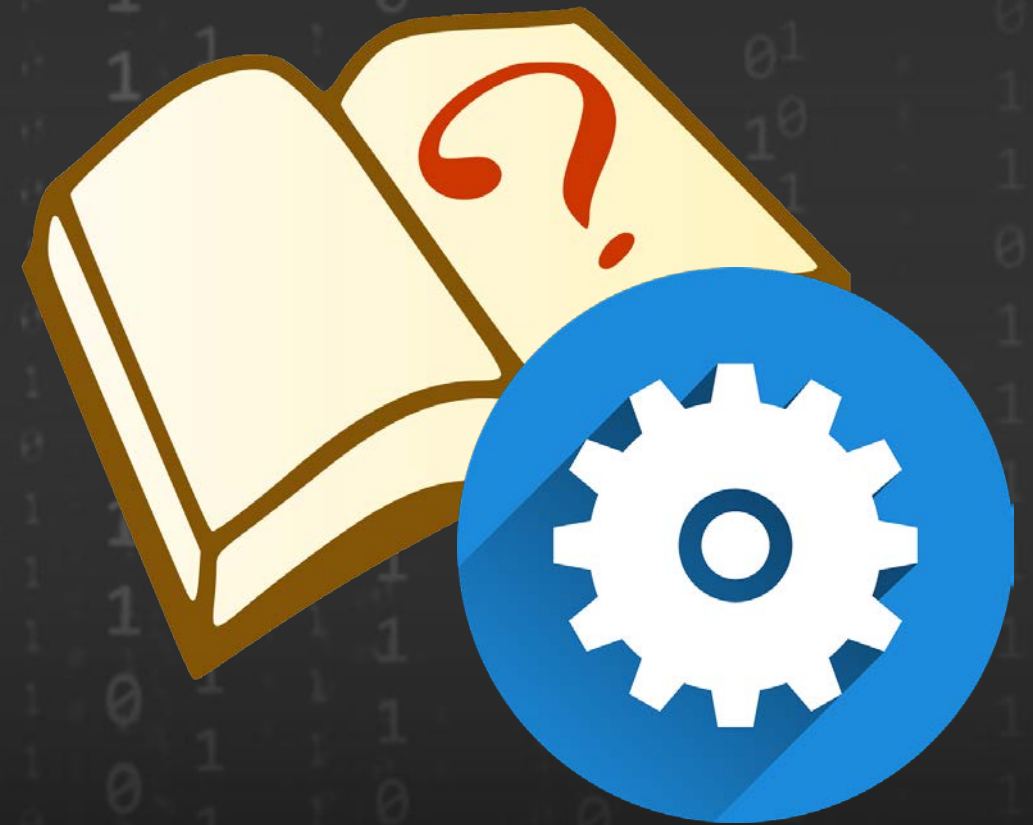
.DSC

**Modify
PCD**

Platform

Lab 1: Writing UEFI Applications with PCDs

In this lab, you'll learn how to write UEFI applications with PCDs.



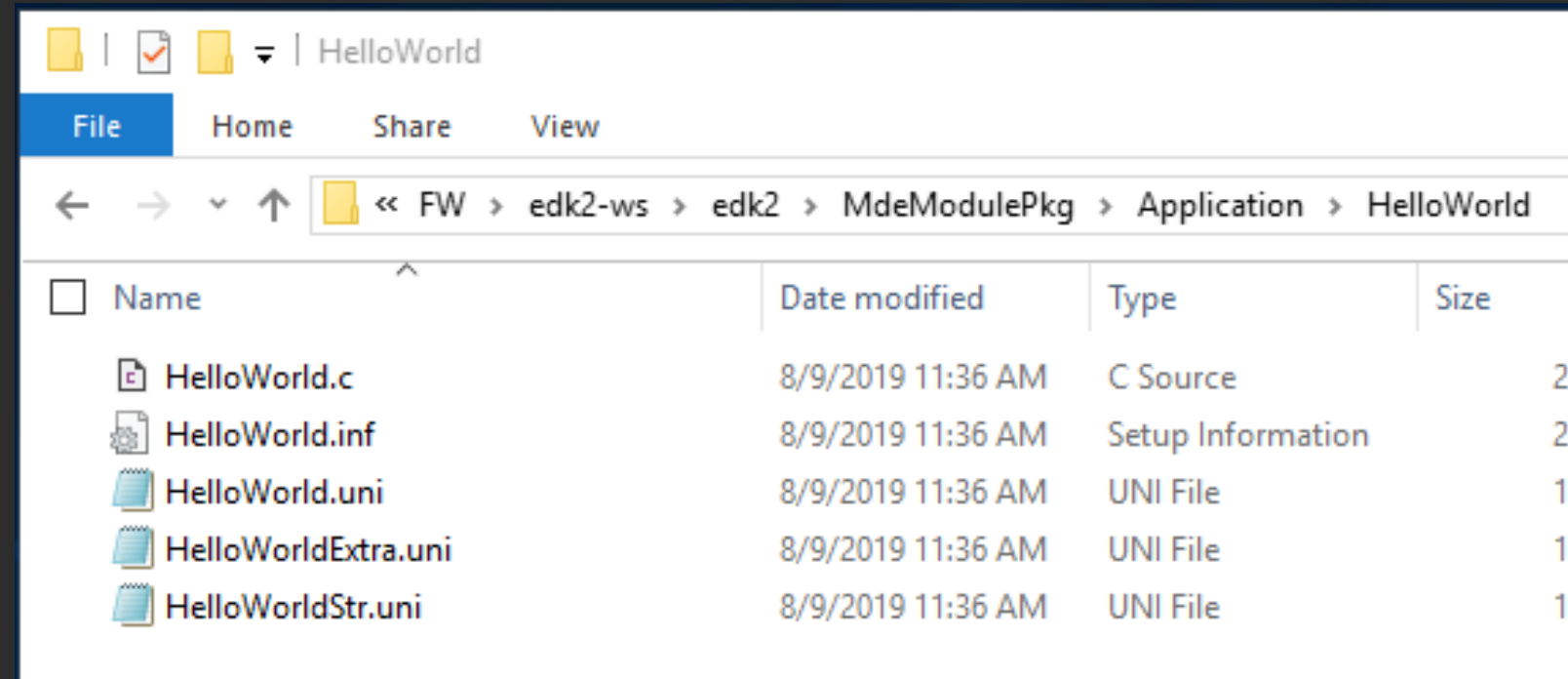
EDK II HelloWorld App Lab

 [MdeModulePkg/Application/HelloWorld](https://github.com/tianocore/MdeModulePkg/Application/HelloWorld)

Locate and Open edk2\MdeModulePkg\Application\HelloWorld\HelloWorld.c

Notice the PCD values

Then Run HelloWorld in Simics



Copy UefiAppLab.vhd file

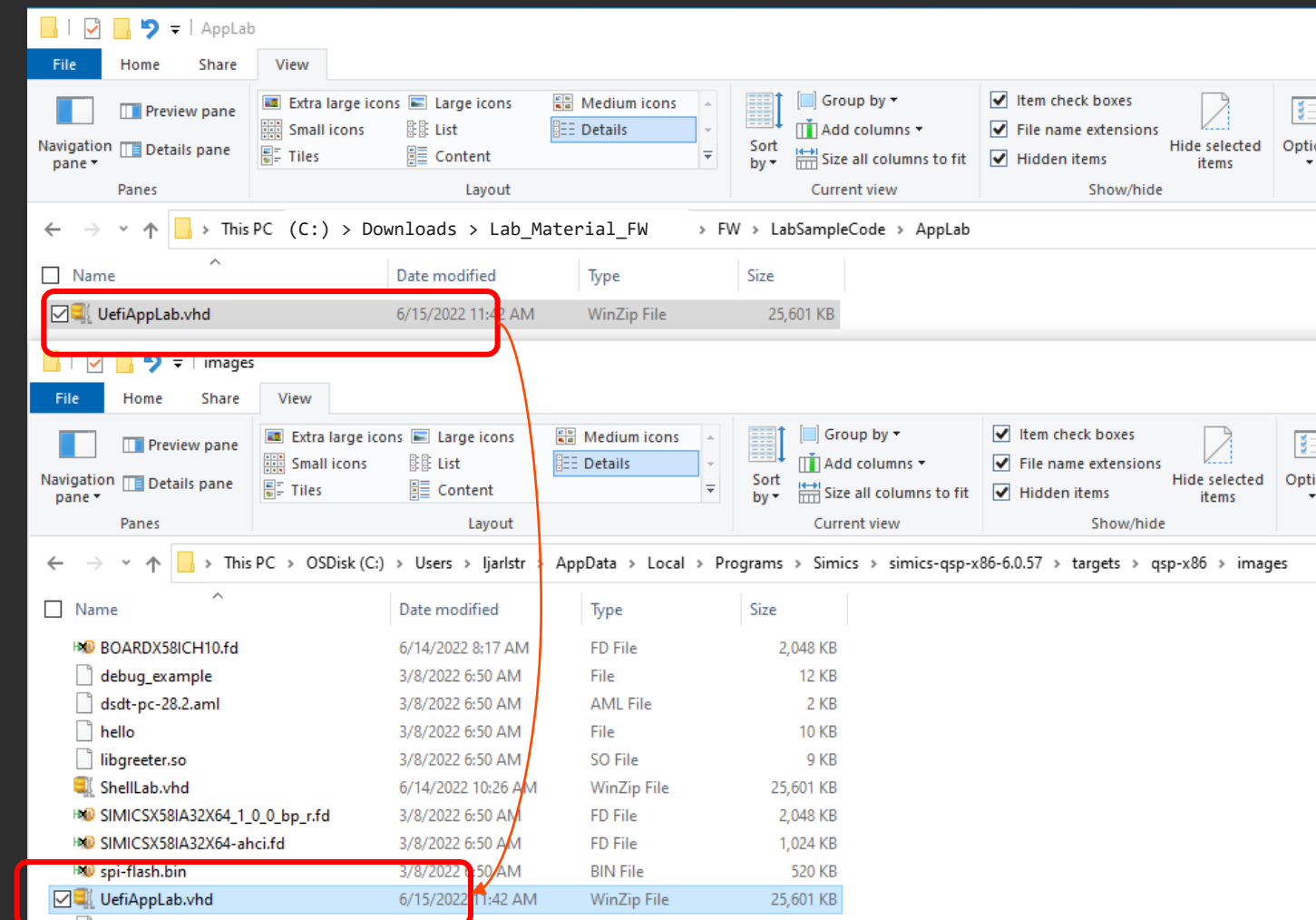
Copy the UefiAppLab.vhd

From:

.../Lab_Material_FW/FW/LabSampleCode/
AppLab/UefiAppLab.vhd

to

%USERPROFILE%\AppData\Local\Pr
ograms\Simics\simics-qsp-x86-
6.0.57\targets\qsp-x86\images



Update the Simics Script to Use the UefiAppLab.vhd image as a file system

Edit the file: qsp-modern-core.simics from

%USERPROFILE%\

\AppData\Local\Programs\Simics\simics-qsp-cpu-6.0.4\targets\qsp-x86\qsp-modern-core.simics

Add the following Line:

```
$disk1_image="%simics%/targets/qsp-x86/images/UefiAppLab.vhd"
```

Before the "run-command-file" line

Save qsp-modern-core.simics

Update the Simics Script

File: qsp-modern-core.simics

```
Decl{
decl {
! Script that runs the Quick Start Platform (QSP) with a modern
!   processor core.

params from "%simics%/targets/qsp-x86/qsp-clear-linux.simics"
default cpu_comp_class = "x86QSP2"
default num_cores = 2
default num_threads = 2
}
$disk1_image="%simics%/targets/qsp-x86/images/UefiAppLab.vhd"

run-command-file "%simics%/targets/qsp-x86/qsp-clear-linux.simics"
```

Comment out if \$disk1_image was added from a previous lab using "#" at the line beginning

Build Platform BoardX58Ich10

1. Open the Visual Studio command prompt
2. Build the Simics BoardX58Ich10

```
$> cd C:\fw\edk2-ws\edk2-platforms\Platform\Intel  
$> python build_bios.py -p BoardX58Ich10 -t VS20XX
```

Where XX is 15x86 or 17 or 19

3. Copy

C:\fw\edk2-ws\Build\SimicsOpenBoardPkg\BoardX58Ich10\DEBUG_VS20XX\FV\BOARDX58ICH10.fd

To

%USERPROFILE%\AppData\Local\Programs\Simics\simics-qsp-x86-6.0.57\targets\qsp-x86\images

Invoke Simics & Run HelloWorld App

1. Open a Windows Command Prompt

```
$> cd simics-projects\my-simics-project-1
```

2. Run the Simics QSP script :

```
$> .\simics targets/qsp-x86/qsp-modern-core.simics  
simics> run
```

Press “F2” at the logo, then Select “Boot Manger” followed by “EFI Internal Shell”

3. At the UEFI Shell prompt

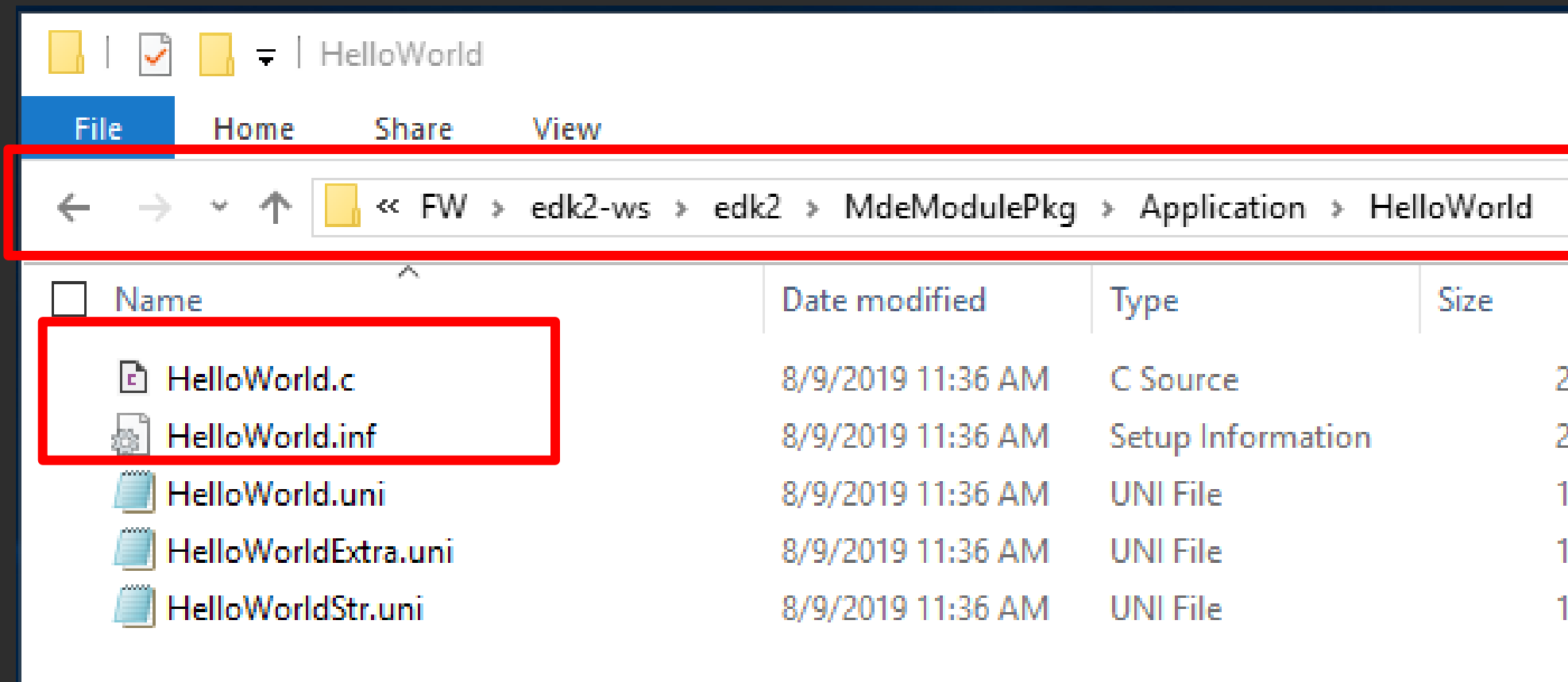
```
Shell> Fs1:  
FS1:\> Helloworld  
UEFI Hello World!  
FS1:\>
```

4. Exit Simics (stop, quit)

How can we force the HelloWorld application to print out 3 times ?

EDK II HelloWorld App Lab

 [MdeModulePkg/Application/HelloWorld](https://github.com/tianocore/edk2/tree/master/MdeModulePkg/Application/HelloWorld)



```

EFI_STATUS
EFIAPI
UefiMain (
    IN EFI_HANDLE      ImageHandle,
    IN EFI_SYSTEM_TABLE *SystemTable
)
{
    UINT32 Index;
    Index = 0;
    // Three PCD type (FeatureFlag, UINT32
    // and String) are used as the sample.
    if (FeaturePcdGet (PcdHelloWorldPrintEnable)) {
        for (Index = 0; Index < PcdGet32 (PcdHelloWorldPrintTimes); Index++) {

            // Use UefiLib Print API to print
            // string to UEFI console

            Print ((CHAR16*)PcdGetPtr (PcdHelloWorldPrintString));

        }
    }

    return EFI_SUCCESS;
}

```


Notice the 3 PCDs

EDK II HelloWorld App Solution

1. Edit the file:

C:\FW\edk2-ws\edk2-platforms\Platform\Intel\SimicsOpenBoardPkg\BoardX58Ich10\OpenBoardPkg.dsc

After the section [PcdsFixedAtBuild] (search for "PcdsFixedAtBuild" or "Hello")



```
OpenBoardPkg.dsc-Notepad
File Edit Format View Help

[PcdsFixedAtBuild]
gEfiMdeModulePkgTokenSpaceGuid.PcdHelloWorldPrintTimes|3
```

Note: it is best to update PCD values in the Platform DSC file.

2. Re-Build BoardX58Ich10

Open A Visual studio Command Prompt

```
$> Cd C:\FW\edk2-ws\edk2-platforms\Platform\Intel\
$> python build_bios.py -p BoardX58Ich10 -t VS20XX
```

Where XX is 15x86 or 17 or 19

Update UefiAppLab.vhd File

3. Mount the UefiAppLab.vhd using Disk Manager: [How To mount VHD](#)

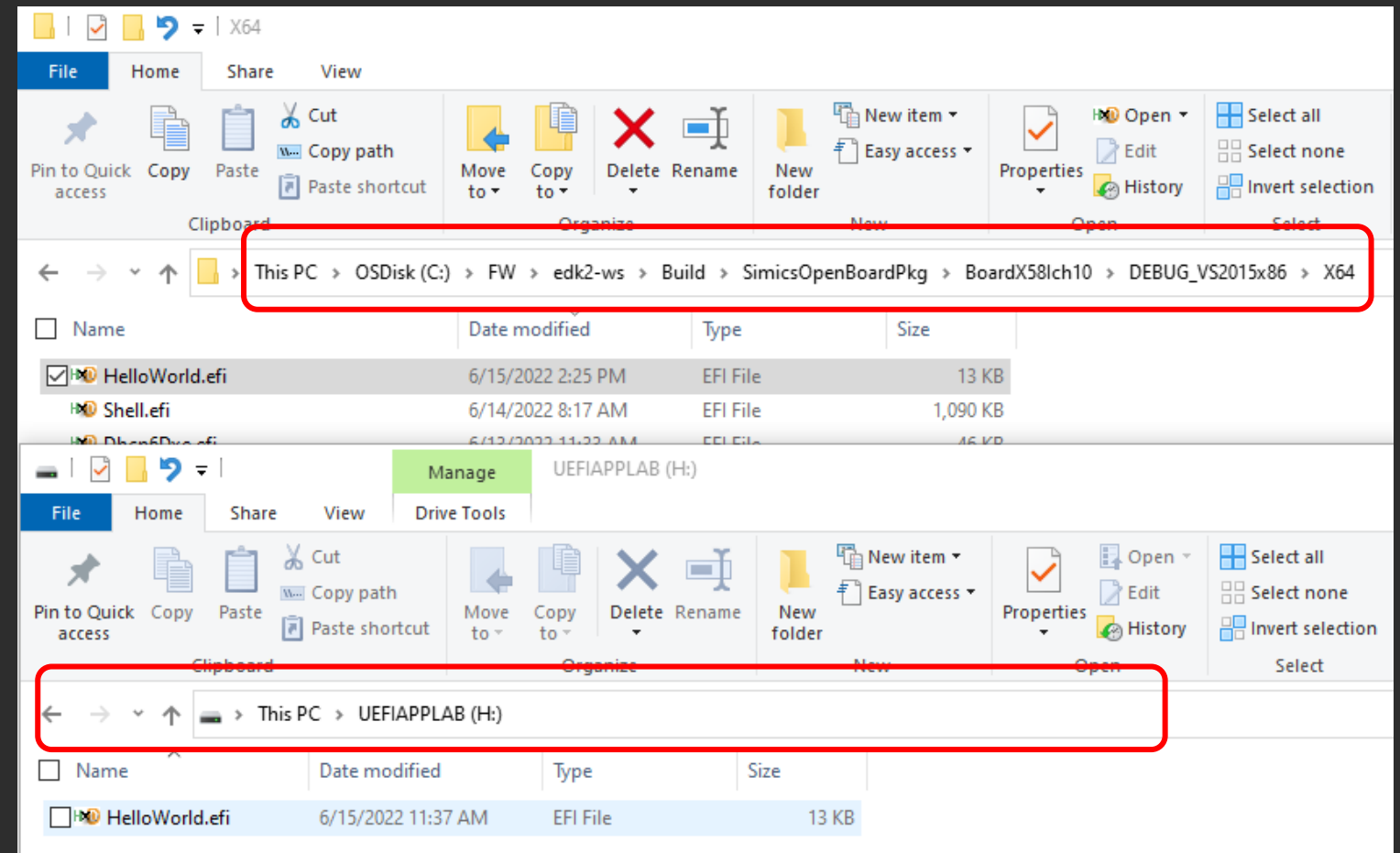
4. Copy & Paste HelloWorld.efi

C:\FW\edk2-ws\Build\SimicsOpenBoardPkg\BoardX58Ich10\DEBUG_VS2015x86\X64\HelloWorld.efi

To

X:\UEFIAPPLAB\ (where X is the VHD Drive)

Detach UefiAppLab.vhd
(can keep open for other Labs)



EDK II HelloWorld App Solution

5. Run Simics script

```
$> .\simics targets/qsp-x86/qsp-modern-core.simics  
simics> run
```

(Press “F2” at the logo, then Select “Boot Manger” followed by “EFI Internal Shell”)

6. At the Shell prompt

```
Shell> Fs1:  
FS1:\> HelloWorld  
UEFI Hello World!  
UEFI Hello World!  
UEFI Hello World!  
FS1:/>
```

7. Exit Simics

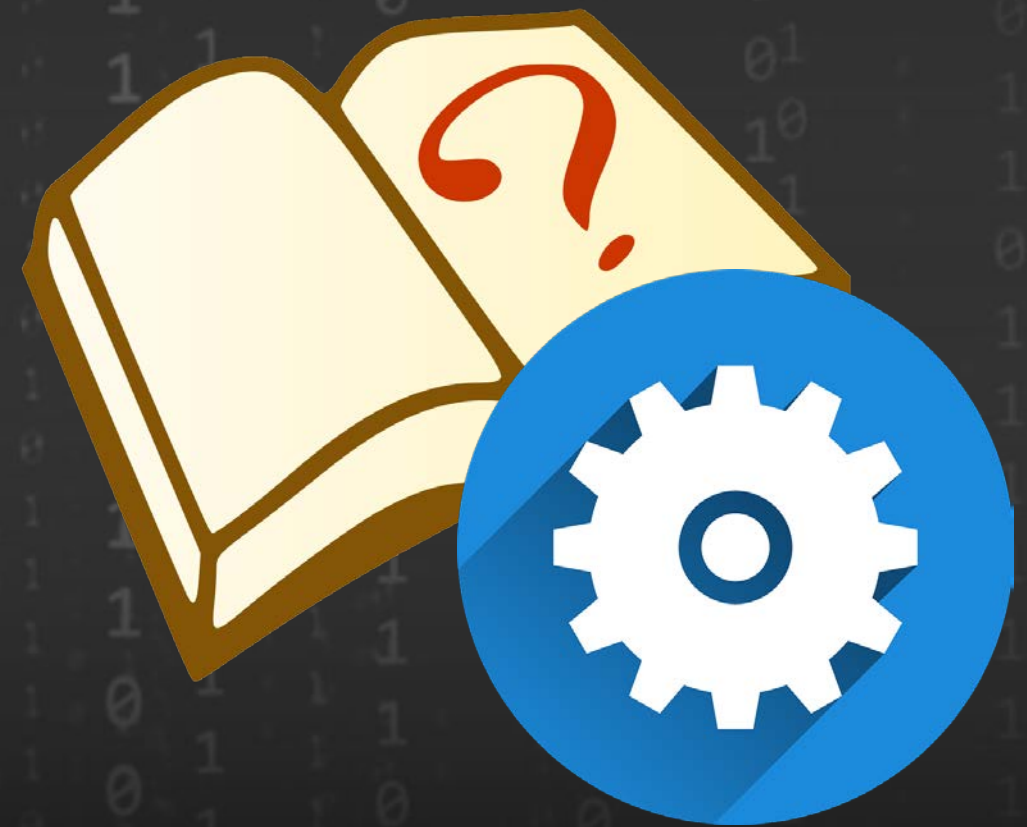
```
simics> Stop then simics> quit
```

How can we change the **string** of the HelloWorld application?

Also see `../edk2/MdeModulePkg/MdeModulePkg.Dec`

Lab 2: Write a Simple UEFI Applications

In this lab, you'll learn how to write simple UEFI applications.



LAB 2 Writing a Simple UEFI Application

In this lab, you'll learn how to write simple UEFI applications.

“C” file

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

.inf file

```
[Defines]
  INF_VERSION           =
  BASE_NAME             =
  FILE_GUID             =
  MODULE_TYPE           =
  VERSION_STRING        =
  ENTRY_POINT           =

[Sources]

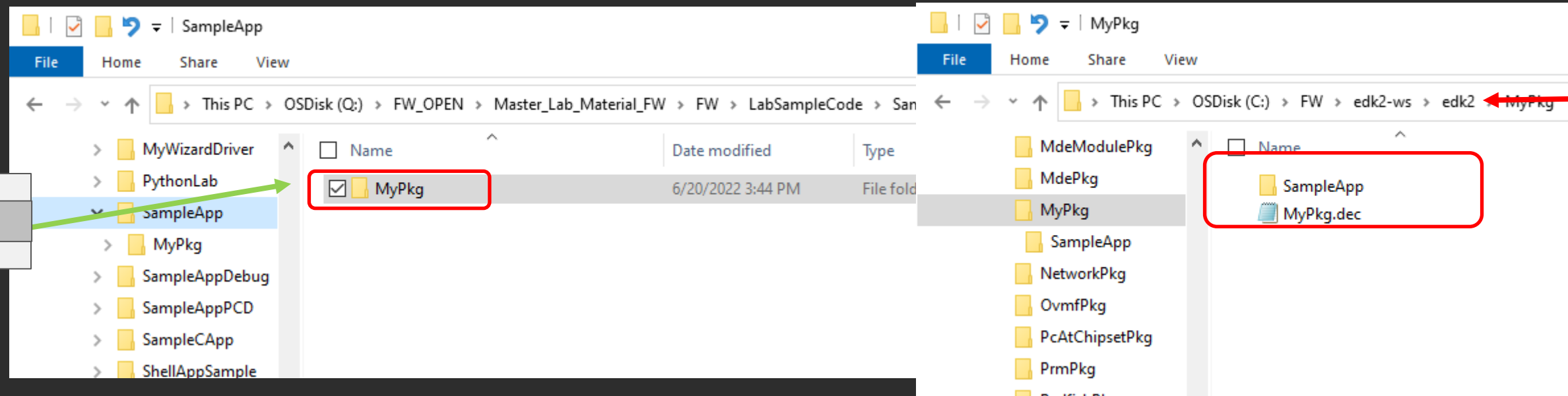
[Packages]

[LibraryClasses]
```

- What goes into a Simplest “C”
- Start with what should go into the Simplest .INF file

Application Lab –start with .c and .inf template

Copy the LabSampleCode/SampleApp/MyPkg directory to C:/FW/edk2-ws/edk2



Edit MyPkg/SampleApp/SampleApp.inf

- Look in the INF for “xxxxxxxxxxxx” sections that will need information
- Create Name & GUID, and then fill in the MODULE_TYPE

Lab 2: Sample Application INF file

```
SampleApp.inf - Notepad
File Edit Format View Help

[Defines]
  INF_VERSION           = 0x00010005
  BASE_NAME             = XXXXXXXXXXXX
  FILE_GUID             = XXXXXXXXXXXX
  MODULE_TYPE           = XXXXXXXXXXXX
  VERSION_STRING        = 1.0
  ENTRY_POINT           = UefiMain

[Sources]
  XXXXXXXXXXXX

[Packages]
  #XXXXXXXXXX

[LibraryClasses]
  #XXXXXXXXXXXXXXXXXX

[Guids]
  # . . .
```

SampleApp
Get a GUID
UEFI_APPLICATION

SampleApp.c

Get a GUID [guidgenerator.com/](https://www.guidgen.com/) or
<https://www.guidgen.com/>

Copy and paste [LabGuide.md](#)

Lab 2: Sample Application 'C' file

```
SampleApp.c - Notepad
File Edit Format View Help

/** @file
  This is a simple shell application
**/
EFI_STATUS
EFIAPI
UefiMain (
    IN EFI_HANDLE          ImageHandle,
    IN EFI_SYSTEM_TABLE    *SystemTable
)
{
    return EFI_SUCCESS;
}
```

*Does not do anything
but return Success*

Lab 2: Will it compile now?

Not yet ...

1. Need to add headers to the .C file
2. Need to add a reference to INF from the platform DSC
3. Need to add a few Package dependencies and libraries to the .INF

Application Lab – Update Files

1. **.DSC** (
edk2-platforms/Platform/Intel/SimicsOpenBoardPkg/BoardX58Ich10/OpenBoard.dsc)
[Components . . .]

Add INF to components section, before build options

Hint: add after comment: # Add new modules here

```
MyPkg/SampleApp/SampleApp.inf
```

1. **.INF** File (MyPkg/SampleApp/SampleApp.inf)

Packages (all depend on MdePkg)

```
[Packages]
```

```
    MdePkg/MdePkg.dec
```

```
[LibraryClasses]
```

```
    UefiApplicationEntryPoint
```

2. **.C** file - Header references File (MyPkg/SampleApp/SampleApp.c)

```
#include <Uefi.h>
```

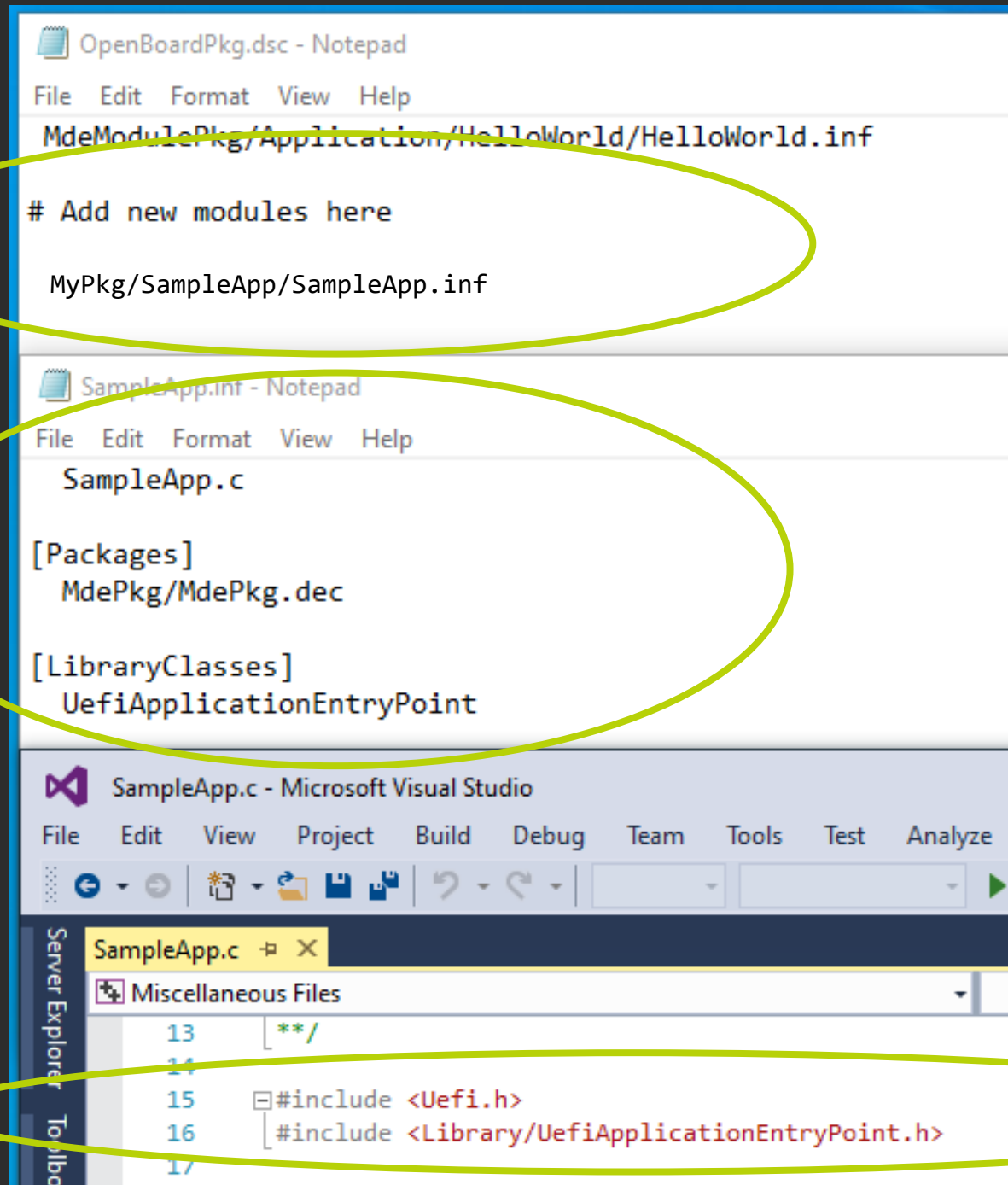
```
#include <Library/UefiApplicationEntryPoint.h>
```

Lab 2: cont. Solution

edk2-platforms/ ...
 SimicsOpenBoardPkg/BoardX58Ich10/
 OpenBoardPkg.dsc

MyPkg/SampleApp/SampleApp.inf

MyPkg/SampleApp/SampleApp.c



```

OpenBoardPkg.dsc - Notepad
File Edit Format View Help
MdeModulePkg/Application/HelloWorld/HelloWorld.inf

# Add new modules here

MyPkg/SampleApp/SampleApp.inf

SampleApp.inf - Notepad
File Edit Format View Help
SampleApp.c

[Packages]
MdePkg/MdePkg.dec

[LibraryClasses]
UefiApplicationEntryPoint

SampleApp.c - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze

SampleApp.c
Miscellaneous Files
13 /**/
14
15 #include <Uefi.h>
16 #include <Library/UefiApplicationEntryPoint.h>
17
  
```

Lab 2 : Will it compile now?

At the VS Command Prompt, Build BoardX58Ich10

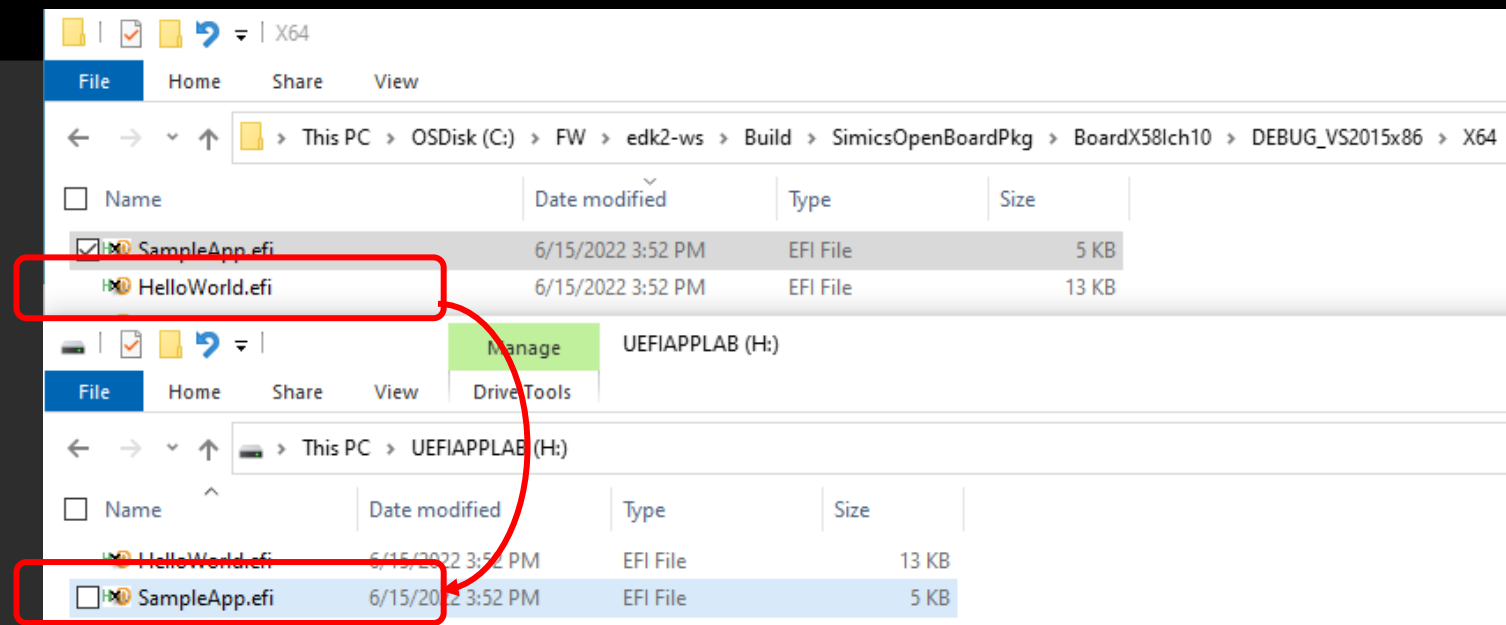
```
$> cd C:\FW\edk2-ws\edk2-platforms\Platform\Intel\
$> python build_bios.py -p BoardX58Ich10 -t VS20XX
```

Copy **SampleApp.efi** from the build directory to the **VHD Disk**

Copy
`..\Build\SimicsOpenBoardPkg\BoardX58Ich10\DEBUG_VS2015x86\X64\SampleApp.efi`
 UefiAppLab Disk

BUILD ERRORS:

If there are build errors, set
 NUMBER_OF_PROCESSORS = 1 in the
 build.cfg file to better see where
 the build error occurs



Build
Directory

VHD
Disk

Invoke Simics & Run SampleApp

1. Run the Simics QSP script from Windows Command Prompt :

```
$> .\simics targets/qsp-x86/qsp-modern-core.simics  
simics> run
```

Press “F2” at the logo, then Select “Boot Manger” followed by “EFI Internal Shell”

3. At the UEFI Shell prompt

```
Shell> Fs1:  
FS1:\> SampleApp.efi  
FS1:\>
```

4. Exit Simics `simics> stop, simics> quit`

Notice that the program will immediately unload because the main function is empty

Possible Build Errors

BUILD ERRORS:

If there are build errors, set `NUMBER_OF_PROCESSORS = 1` in the `build.cfg` file to better see where the build error occurs

Error on SampleApp.inf

```

c:\ Developer Command Prompt for VS2015

Processing meta-data ..

build...
c:\fw\edk2\SampleApp\SampleApp.inf(21): error 3000: No value specified
FILE_GUID

- Failed -
Build end time: 09:11:30, Jul.25 2018
Build total time: 00:00:03

```

This failure if
FILE_GUID blank

```

c:\ Developer Command Prompt for VS2015

Processing meta-data .
Architecture(s) = X64
Build target    = DEBUG
Toolchain       = VS2015x86

Active Platform      = c:\fw\edk2-ws\edk2\EmulatorPkg\EmulatorPkg.dsc
.....

- Failed -
Build end time: 10:31:17, Feb.02 2022
Build total time: 00:00:13

```

```

c:\ Developer Command Prompt for VS2015

Processing meta-data .....

build...
: error C0DE: Unknown fatal error when processing [c:\fw\edk2\SampleApp\SampleApp.inf]

(Please send email to edk2-devel@lists.01.org for help, attaching following call stack trace!)

(Python 2.7.14 on win32) Traceback (most recent call last):
  File "build\build.py", line 2493, in Main
  File "build\build.py", line 2226, in Launch
  File "build\build.py", line 2047, in _MultiThreadBuildPlatform
  File "c:\Users\Public\Documents\BuildPool\BaseTools\build\Source\Python\AutoGen\AutoGen.py",
line 4391, in CreateCodeFile
  File "c:\Users\Public\Documents\BuildPool\BaseTools\build\Source\Python\AutoGen\AutoGen.py",
line 3604, in _GetAutoGenFileList
  File "c:\Users\Public\Documents\BuildPool\BaseTools\build\Source\Python\AutoGen\GenC.py", line
e 2075, in CreateCode
  File "c:\Users\Public\Documents\BuildPool\BaseTools\build\Source\Python\AutoGen\GenC.py", line
e 2033, in CreateHeaderCode
  File "c:\Users\Public\Documents\BuildPool\BaseTools\build\Source\Python\Common\Misc.py", line
308, in GuidStringToGuidStructureString
IndexError: list index out of range

- Failed -
Build end time: 09:15:55, Jul.25 2018
Build total time: 00:00:24

```

This failure if "XXXX. . ." are left or other
improper format of the GUID

The FILE_GUID was invalid or not updated from "XXX..." to a proper formatted GUID in INF

Possible Build Errors

Error on SampleApp.inf

```
c:\Developer Command Prompt for VS2015

Building ... c:\fw\edk2\MdeModulePkg\Universal\LoadFileOnFv2\LoadFileOnFv2.inf [IA32]
SampleApp.c
  B Creating library c:\fw\edk2\Build\NT32IA32\DEBUG_VS2013x86\IA32\SecMain.lib and object c:\fw\edk2\Build\NT32IA32\DEBUG_VS2013x86\IA32\SecMain.exp
Building ... c:\fw\edk2\MdeModulePkg\Universal\PlatformDriOverrideDxe\PlatformDriOverrideDxe.inf [IA32]
Generating code
Building ... c:\fw\edk2\MdeModulePkg\Application\VariableInfo\VariableInfo.inf [IA32]
c:\fw\edk2\Build\NT32IA32\DEBUG_VS2013x86\IA32\SampleApp\SampleApp\DEBUG\AutoGen.h(16) : fatal error C1083: Cannot open include file: 'Base.h': No such file or directory
NMAKE : fatal error U1077: '"C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin\cl.exe"' : return code '0x2'
Stop.

build...
: error 7000: Failed to execute command
C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin\nmake.exe /nologo tbuild [c:\fw\edk2\Build\NT32IA32\DEBUG_VS2013x86\IA32\SampleApp\SampleApp]

build...
: error F002: Failed to build module
c:\fw\edk2\SampleApp\SampleApp.inf [IA32, VS2013x86, DEBUG]

- Failed -
Build end time: 09:23:56, Jul.25 2018
Build total time: 00:00:41
```

The [Packages] was invalid or did not specify MdePkg/MdePkg.dec properly

Possible Build Errors

Compiler Error on SampleApp.c

```
c:\ Developer Command Prompt for VS2015
2\SampleApp\SampleApp\DEBUG /Ic:\fw\edk2\MdePkg /Ic:\fw\edk2\MdePkg\Include /Ic:\fw\edk
2\MdePkg\Include\Ia32 c:\fw\edk2\SampleApp\SampleApp.c
cl : Command line warning D9025 : overriding '/O1' with '/Od'
SampleApp.c
Building ... c:\fw\edk2\MdeModulePkg\Application\BootManagerMenuApp\BootManagerMenuApp.inf
[IA32]
c:\fw\edk2\SampleApp\SampleApp.c(16) : fatal error C1083: Cannot open include file: 'Libra
ry/UefiAplicationEntryPoint.h': No such file or directory
Building ... c:\fw\edk2\MdeModulePkg\Universal\LoadFileOnFv2\LoadFileOnFv2.inf [IA32]
NMAKE : fatal error U1077: '"C:\Program Files (x86)\Microsoft Visual Studio 12.0\vc\bin\cl
.exe"' : return code '0x2'
Stop.
Building ... c:\fw\edk2\MdeModulePkg\Universal\PlatformDriOverrideDxe\PlatformDriOverrideD
xe.inf [IA32]

build...
: error 7000: Failed to execute command
C:\Program Files (x86)\Microsoft Visual Studio 12.0\vc\bin\nmake.exe /nologo tbuil
d [c:\fw\edk2\Build\NT32IA32\DEBUG_VS2013x86\IA32\SampleApp\SampleApp]

build...
: error F002: Failed to build module
c:\fw\edk2\SampleApp\SampleApp.inf [IA32, VS2013x86, DEBUG]
```

The #include <Library/UefiApplicationEntryPoint.h> has a typo (“Application” not “Aplication”)

Possible Build Errors

Compile Linker Error on unresolved reference

```

c:\ Developer Command Prompt for VS2015
"C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin\link.exe" /OUT:c:\fw\edk2\Build\NT32IA32\DEBUG_VS2013x86\IA32\SampleApp\SampleApp\DEBUG\SampleApp.dll /NOLOGO /NODEFAULTLIB /IGNORE:4001 /OPT:REF /OPT:ICF=10 /MAP /ALIGN:32 /SECTION:.xdata,D /SECTION:.pdata,D /MACHINE:X86 /LTCG /DLL /ENTRY:_ModuleEntryPoint /SUBSYSTEM:EFI_BOOT_SERVICE_DRIVER /SAFESEH:NO /BASE:0 /DRIVER /DEBUG /EXPORT InitializeDriver=_ModuleEntryPoint /BASE:0x10000 /ALIGN:4096 /FILEALIGN:4096 /SUBSYSTEM:CONSOLE @c:\fw\edk2\Build\NT32IA32\DEBUG_VS2013x86\IA32\SampleApp\SampleApp\OUTPUT\static_library_files.lst
Building ... c:\fw\edk2\MdeModulePkg\Universal\SetupBrowserDxe\SetupBrowserDxe.inf [IA32]
LINK : error LNK2001: unresolved external symbol _ModuleEntryPoint
c:\fw\edk2\Build\NT32IA32\DEBUG_VS2013x86\IA32\SampleApp\SampleApp\DEBUG\SampleApp.lib : fatal error LNK1120: 1 unresolved externals
Building ... c:\fw\edk2\MdeModulePkg\Universal\DisplayEngineDxe\DisplayEngineDxe.inf [IA32]
NMAKE : fatal error U1077: '"C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin\link.exe"' : return code '0x400'
Stop.

build...
: error 7000: Failed to execute command
C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin\nmake.exe /nologo tbuild [c:\fw\edk2\Build\NT32IA32\DEBUG_VS2013x86\IA32\SampleApp\SampleApp]

build...
: error F002: Failed to build module
c:\fw\edk2\SampleApp\SampleApp.inf [IA32, VS2013x86, DEBUG]

```

The SampleApp.inf section [LibraryClasses] did not reference UefiApplicationEntryPoint

Possible Build Errors

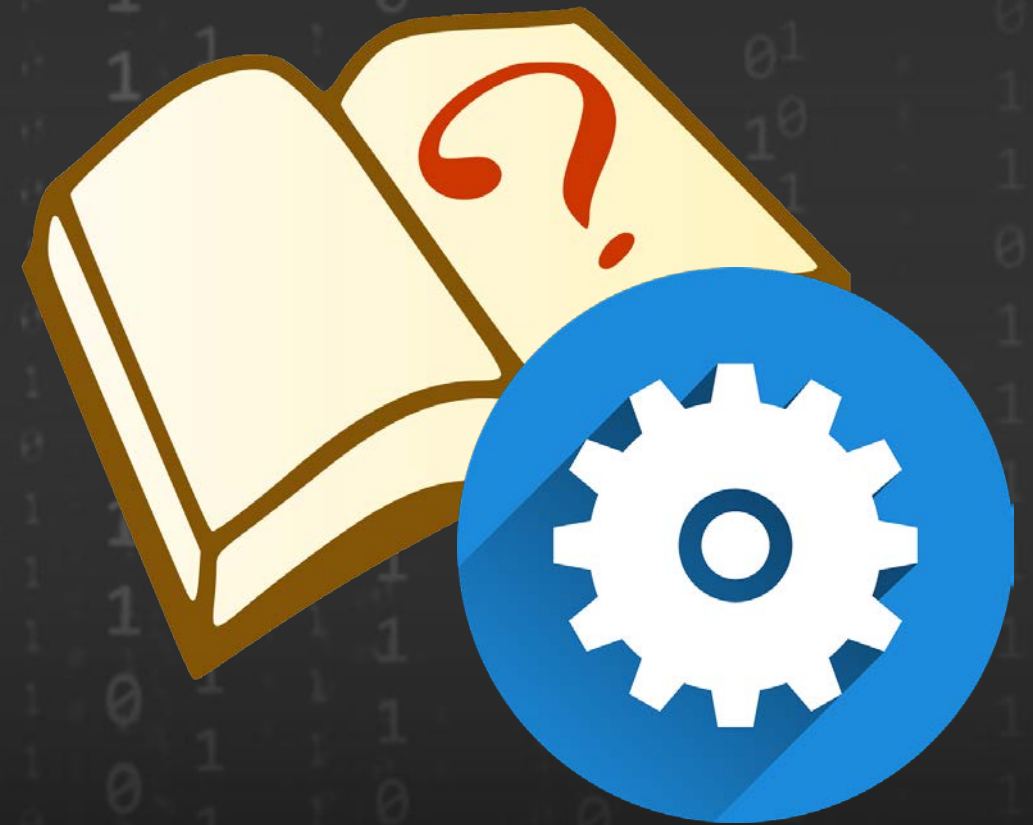
Error at the Shell prompt

```
Shell> fs1:  
FS1:\> SampleApp  
'SampleApp' is not recognized as an internal or external command, operable program, or script file.  
FS1:\> ls SampleApp.efi  
ls: File Not Found - 'FS1:\'  
FS1:\> _
```

Ensure the SampleApp.inf BaseName is SampleApp

Lab 2.1: Build Switches

In this lab, you'll remove the build switches to be always TRUE




Build MACRO Switches

The build for BoardX58Ich10 OpenBoardPkg is using build MACRO Switch:

-D ADD_SHELL_STRING – used to change a string in the UEFI Shell application, only used for EDK II Training (requires ShellPkg be re-built on a change of this switch)

OpenBoardPkg.dsc - Notepad

File Edit Format View Help



```
ShellPkg/Application/Shell/Shell.inf {
  <PcdsFixedAtBuild>
    gEfiShellPkgTokenSpaceGuid.PcdShellLibAutoInitialize|FALSE
  <LibraryClasses>
    NULL|ShellPkg/Library/UefiShellLevel2CommandsLib/UefiShellLevel2CommandsLib.inf
    NULL|ShellPkg/Library/UefiShellLevel1CommandsLib/UefiShellLevel1CommandsLib.inf
!if $(ADD_SHELL_STRING) == TRUE
  # Training Lib for build switch lab
  NULL|ShellPkg/Library/UefiShellLevel3CommandsLib_Training_Lib/UefiShellLevel3Commands_Training_Lib.inf
!else
  # normal Lib for build switch
  NULL|ShellPkg/Library/UefiShellLevel3CommandsLib/UefiShellLevel3CommandsLib.inf
!endif
```

Lab 2.1: Compiling w/ Build Switch

Result without the build switch

```
Shell> ver
UEFI Interactive Shell v2.2
EDK II
UEFI v2.70 (EDK II, 0x00010000)
Shell> _
```

Result with the build switch

```
Shell> ver
UEFI Interactive Shell v2.2 -From ADD_SHELL_STRING Switch
EDK II
UEFI v2.70 (EDK II, 0x00010000)
Shell> _
```

Lab 2.1: Compiling w/ Build Switch

Check Result without the build switch

Run the Simics QSP script from Windows Command Prompt :

```
$> .\simics targets/qsp-x86/qsp-modern-core.simics  
simics> run
```

Press “F2” at the logo, then Select “Boot Manger” followed by “EFI Internal Shell”

At the UEFI Shell prompt

```
Shell> ver  
UEFI Interactive Shell v2.2  
EDK II  
UEFI v2.70 (EDK II, 0x00010000)  
Shell> _
```

Exit Simics `simics> stop`, `simics> quit`

Lab 2.1: Compiling w/ Build Switch

Add the Build Switch “-D ADD_SHELL_STRING”

Two Ways: 1. update Python Script or 2. Add list of DEFINES in .DSC file (Preferred)

1. The Build command is part of the python script, build_bios.py
Notice the Build command from the python script for BoardX58Ich10:

```
=====  
Calling build -n 0 --log=Build.log --report-file=BuildReport.log
```

Update line 388 of build_bios.py to include the “-D ADD_SHELL_STRING”

```
command = ["build", "-n", config["NUMBER_OF_PROCESSORS"], "-D", "ADD_SHELL_STRING"]
```

Re-Build BoardX58Ich10

From a Visual Studio Command Prompt

```
$> Cd C:\FW\edk2-ws\edk2-platforms\Platform\Intel\  
$> python build_bios.py -p BoardX58Ich10 -t VS20XX
```

```
=====  
Calling build -n 0 -D ADD_SHELL_STRING --log=Build.log --report-file=BuildReport.log  
Build environment: Windows 10 10.0.10041.500
```

Lab 2.1: Compiling w/ Build Switch

1

Invoke Simics and Test Shell Ver command

Copy C:\fw\edk2-ws\Build\SimicsOpenBoardPkg\BoardX58Ich10\DEBUG_VS20XX\FV\BOARDX58ICH10.fd To
%USERPROFILE%\AppData\Local\Programs\Simics\simics-qsp-x86-6.0.57\targets\qsp-x86\images

Run the Simics QSP script from Windows Command Prompt :

```
$> .\simics targets/qsp-x86/qsp-modern-core.simics  
simics> run
```

Press “F2” at the logo, then Select “Boot Manger” followed by “EFI Internal Shell”

At the UEFI Shell prompt, type: ver

```
Shell> ver  
UEFI Interactive Shell v2.2 -From ADD_SHELL_STRING Switch  
EDK II  
UEFI v2.70 (EDK II, 0x00010000)  
Shell> _
```

Exit Simics `simics> stop`, `simics> quit`

Lab 2.1: Compiling w/ Build Switch

2 Add list of DEFINES in .DSC file. This is the preferred method for EDK II when a Build script is used

1. Update line 388 of build_bios.py and remove the “-D ADD_SHELL_STRING” , then Save
2. Edit the file `C:/fw/edk2-ws/edk2-platforms/Platform/Intel/SimicsOpenBoardPkg/BoardX58Ich10/OpenBoardPkgBuildOption.dsc`
 - add the following DEFINE after Line 7 – Save after update:

```
[Defines]
# For UEFI / EDK II Training
# This flag is to enable a different ver string for building of the ShellPkg
# These can be changed on the command line.
#
  DEFINE  ADD_SHELL_STRING          = TRUE
```

Re-Build BoardX58Ich10

From a Visual Studio Command Prompt

```
$> Cd C:\FW\edk2-ws\edk2-platforms\Platform\Intel\
$> python build_bios.py -p BoardX58Ich10 -t VS20XX
```

Lab 2.1: Compiling w/ Build Switch

2

Invoke Simics and Test Shell Ver command

Copy C:\fw\edk2-ws\Build\SimicsOpenBoardPkg\BoardX58Ich10\DEBUG_VS20XX\FV\BOARDX58ICH10.fd To
%USERPROFILE%\AppData\Local\Programs\Simics\simics-qsp-x86-6.0.57\targets\qsp-x86\images

Run the Simics QSP script from Windows Command Prompt :

```
$> .\simics targets/qsp-x86/qsp-modern-core.simics  
simics> run
```

Press “F2” at the logo, then Select “Boot Manger” followed by “EFI Internal Shell”

At the UEFI Shell prompt, type: ver

```
Shell> ver  
UEFI Interactive Shell v2.2 -From ADD_SHELL_STRING Switch  
EDK II  
UEFI v2.70 (EDK II, 0x00010000)  
Shell> _
```

Exit Simics `simics> stop`, `simics> quit`

Knowledge Check from LAB 2

1. How to write a simple native UEFI Application
2. Each module requires a .inf file with a unique GUID (use <http://www.guidgenerator.com/>)
3. The module created will be the base name defined in the .inf file
4. The module's .inf file is required to be included in the platform .dsc file
5. The [Packages] section is required at minimum to include MdePkg/MdePkg.dec
6. When using a Build Switch (-D) on the command line it overrides the value in the .DSC file

Lab 2: If there are build errors ...

See class files for the solution from: Lab_Material_FW\LabSampleCode

1. Copy the FW\LabSampleCode\SampleApp\MyPkg directory to C:\FW\edk2-ws\edk2
2. See class files for the solution . . . FW\LabSampleCode\LabSolutions\LessonB.2
3. Copy the .inf and .c files to C:\FW\edk2-ws\edk2\MyPkg\SampleApp
4. Search sample DSC for reference to SampleApp.inf and add this line to your workspace DSC file
C:\FW\edk2-ws\edk2-platforms\Platform\Intel\SimicsBoardPkg
 \BoardX58Ich10\OpenBoard.dsc (near the bottom)

```
MyPkg/SampleApp/SampleApp.inf
```

Invoke python build script again and check the solution

ADD FUNCTIONALITY

Add Functionality to the Simple UEFI Application :

Next 3 Labs

Lab 3: Print the UEFI System Table

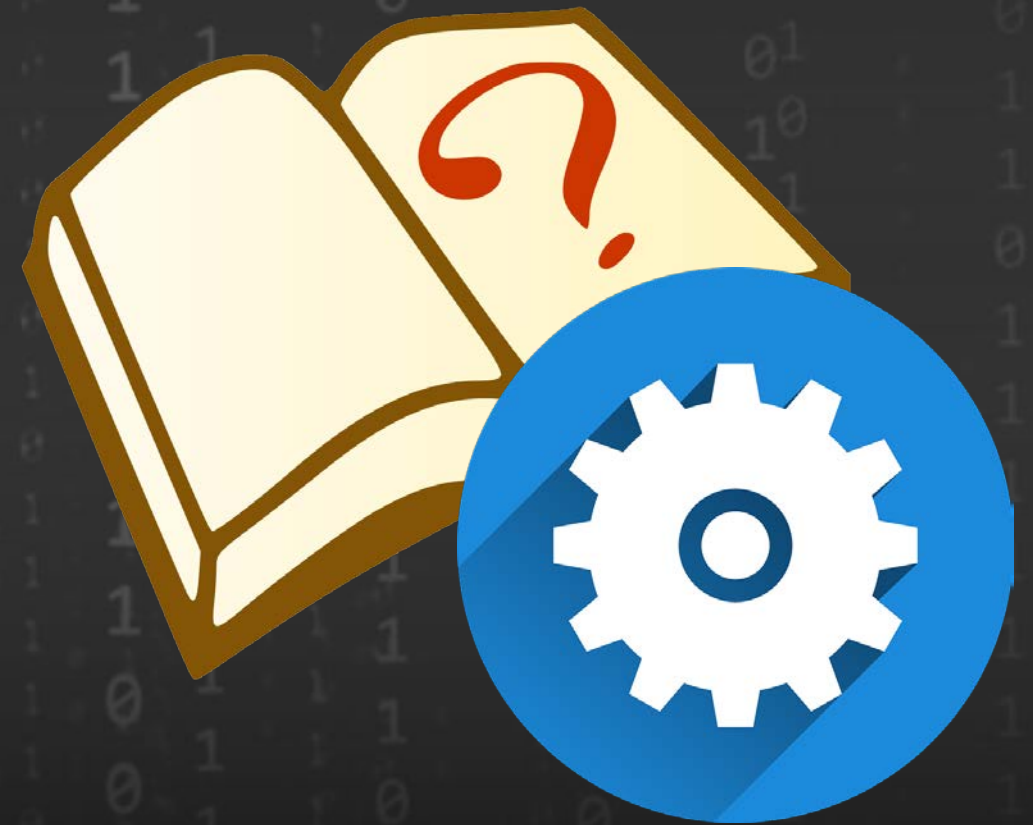
Lab 4: Wait for an Event

Lab 5: Create a Simple Typewriter function & Create a PCD to enable

Solutions in .../FW/LabSampleCode/LabSolutions/LessonB.n

Lab 3: Print the UEFI System Table

Add code to print the hex address of the EFI System Table pointer to the console.



Lab 3 : Add System Table Code

Add code to print to the console the hex address of the system table pointer

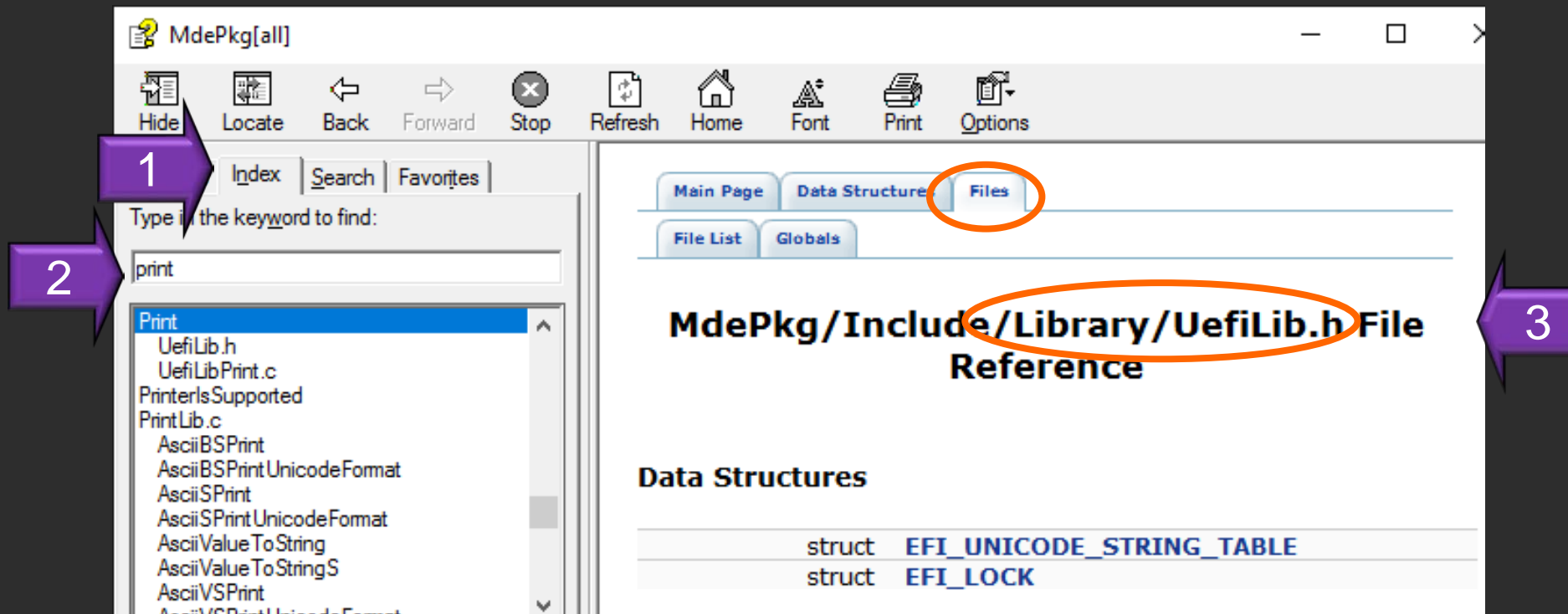
- Where is the “print” function?
- Where does the app get the pointer value?
(compared to **mem** command below)

```
FS1:\> mem
Memory Address 00000000DEFED018 78 Bytes
DEFED018: 49 42 49 20 53 59 53 54-46 00 02 00 78 00 00 00 *IBI SYSTF...x...*
DEFED028: 67 DC 22 34 00 00 00 00-98 8A FD DE 00 00 00 00 *g."4.....*
DEFED038: 00 00 01 00 00 00 00 00-98 DB 60 DE 00 00 00 00 *.....*
DEFED048: C0 20 EC DD 00 00 00 00-98 94 F9 DD 00 00 00 00 *. ....*
DEFED058: 20 D3 0E DE 00 00 00 00-18 CF 60 DE 00 00 00 00 * .....*
DEFED068: 30 24 EC DD 00 00 00 00-98 DB FE DE 00 00 00 00 *0$.....*
DEFED078: 80 3D 32 DF 00 00 00 00-0B 00 00 00 00 00 00 *. =2.....*
DEFED088: 98 DC FE DE 00 00 00 00-
*.....*

Valid EFI Header at Address 00000000DEFED018
-----
System: Table Structure size 00000078 revision 00020046
ConIn (00000000DDEC20C0) ConOut (00000000DE0ED320) StdErr (00000000DDEC2430)
Runtime Services 00000000DEFEDB98
Boot Services 00000000DF323D80
SAL System Table 0000000000000000
ACPI Table 00000000DEFF9000
ACPI 2.0 Table 00000000DEFF9014
MPS Table 0000000000000000
SMBIOS Table 0000000000000000
FS1:\> SampleApp
System Table: 0xDEFED018
```

Lab 3 : Locating the Print() Function

1. Search the MdePkg.chm and find that the Print function by clicking on the “Index” tab
2. Type “Print” and double click
3. Scroll to the top in the right window to see that the print function is in the UefiLib.h file



"MdePkg Document With Libraries.chm" located in ...Lab_Material_FW/FW/Documents

Lab 3 : Modifying .C & .INF Files

```
SampleApp.c - Notepad
File Edit Format View Help

SampleApp.c
#include <Uefi.h>
#include <Library/UefiApplicationEntryPoint.h>
#include <Library/UefiLib.h>

EFI_STATUS
EFIAPI
UefiMain (
    IN EFI_HANDLE          ImageHandle,
    IN EFI_SYSTEM_TABLE    *SystemTable
)
{
    Print(L"System Table: 0x%p\n", SystemTable);
    return EFI_SUCCESS;
}
```

```
SampleApp.inf - Notepad
File Edit Format View Help

SampleApp.inf
[LibraryClasses]
    UefiApplicationEntryPoint
    UefiLib
```

Note: Solution files are in the lab materials directory ...FW\LabSampleCode\LabSolutions\LessonB.3

Lab 3 : Build and Test SampleApp

1. At the VS Command Prompt, Re-Build BoardX58Ich10

```
$> Cd C:\FW\edk2-ws\edk2-platforms\Platform\Intel\  
$> python build_bios.py -p BoardX58Ich10 -t VS20XX
```

2. Copy SampleApp.efi from the build directory to the VHD Disk

```
Copy ..\Build\SimicsOpenBoardPkg\BoardX58Ich10\DEBUG_VS20XX\X64\SampleApp.efi UefiAppLab Disk
```

3. Run the Simics QSP script from Windows Command Prompt :

```
$> .\simics targets/qsp-x86/qsp-modern-core.simics  
simics> run
```

4. At the UEFI Shell prompt

```
Shell> Fs1:  
FS1:\> SampleApp.efi  
System Table: 0x0DEFED018  
FS1:\>
```

5. Exit Simics simics> stop, simics> quit

Verify by using the “mem” command

Lab 4: Waiting for an Event

In this lab, you'll learn how to locate code and .chm files to help write EFI code for waiting for an event



Lab 4 : Add Wait for Event

Add code to make your application wait for a key press event (WaitForEvent / WaitForKey)

```
Press ESC in 2 seconds to skip startup.nsh or any other key to continue.  
Shell> fs1:  
FS1:\> SampleApp  
System Table: 0xDEDED018  
  
Press any Key to continue :  
_
```

- Where are these functions located?
- What else can you do with the key press?

Locate Functions: WaitForEvent / WaitForKey

- Search MdePkg.chm- "MdePkg Document With Libraries.chm" located in ...
Lab_Material_FW/FW/Documentation
 - Locate WaitForEvent in Boot Services
 - Locate WaitForKey and find (EFI_SIMPLE_TEXT_INPUT_PROTOCOL will be part of ConIn)
- Check the [UEFI Spec](#) for parameters needed:
 - WaitForEvent is referenced via Boot Services pointer, which is referenced via EFI System Table
 - WaitForKey can be referenced through the EFI System Table passed into the application
- **OR** Search the working space for WaitForEvent for an example
- One can be found in [MdePkg/Library/UefiLib/Console.c](#) ~ In 569:

Lab 4 : Update the C File for WaitForKey

Search the work space and find the following [MdePkg/Library/UefiLib/Console.c](#) ~ ln 563:

```

Console.c - Notepad
File Edit Format View Help

UINTN                               EventIndex;

// If we encounter error, continue to read another key in.
//
if (Status != EFI_NOT_READY) {
    continue;
}
gBS->WaitForEvent (1, &gST->ConIn->WaitForKey, &EventIndex);

```

Line 410

Line 563

Add the following to SampleApp.c

```

SampleApp.c - Notepad
File Edit Format View Help

UINTN                               EventIndex;
Print(L"System Table: 0x%p\n",SystemTable);
Print(L"\nPress any Key to continue : \n");
gBS->WaitForEvent (1, &gST->ConIn->WaitForKey, &EventIndex);

```

Copy and Paste

Lab 4: Test Compile

However, this won't compile ... gBS and gST are not defined.

```
Building ... c:\fw\edk2\MdeModulePkg\Application\BootManagerMenuApp\BootManagerMenuApp.inf [IA32]
c:\fw\edk2\SampleApp\SampleApp.c(42) : error C2065: 'gBS' : undeclared identifier
c:\fw\edk2\SampleApp\SampleApp.c(42) : error C2223: left of '->WaitForEvent' must point to struct/union
c:\fw\edk2\SampleApp\SampleApp.c(42) : error C2065: 'gST' : undeclared identifier
c:\fw\edk2\SampleApp\SampleApp.c(42) : error C2223: left of '->ConIn' must point to struct/union
Building ... c:\fw\edk2\MdeModulePkg\Universal\LoadFileOnFv2\LoadFileOnFv2.inf [IA32]
NMAKE : fatal error U1077: '"C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin\cl.exe"' : return code '0x2'
Stop.
Building ... c:\fw\edk2\MdeModulePkg\Universal\PlatformDriOverrideDxe\PlatformDriOverrideDxe.inf [IA32]
]

build...
: error 7000: Failed to execute command
C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin\nmake.exe /nologo tbuild [c:\fw\edk2\Build\NT32IA32\DEBUG_VS2013x86\IA32\SampleApp\SampleApp]
```

Search the MdePkg.chm for “gBS” and “gST” – they are located in UefiBootServicesTableLib.h

Add the boot services lib to SampleApp.c . . .
#include <Library/UefiBootServicesTableLib.h>

(hint: Lesson B.4 has the solution)

Lab 4: Update for gBS & gST

```
SampleApp.c - Notepad
File Edit Format View Help
#include <Uefi.h>
#include <Library/UefiApplicationEntryPoint.h>
#include <Library/UefiLib.h>
#include <Library/UefiBootServicesTableLib.h>
// . . .
EFI_STATUS
EFIAPI
UefiMain (
    IN EFI_HANDLE          ImageHandle,
    IN EFI_SYSTEM_TABLE    *SystemTable
)
{
    UINTN                    EventIndex;
    Print(L"System Table: 0x%p\n", SystemTable);
    Print(L"\nPress any Key to continue :\n");
    gBS->WaitForEvent (1, &gST->ConIn->WaitForKey, &EventIndex);
    return EFI_SUCCESS;
}
```

Lab 4 : Build and Test SampleApp

1. At the VS Command Prompt, Re-Build BoardX58Ich10

```
$> Cd C:\FW\edk2-ws\edk2-platforms\Platform\Intel\  
$> python build_bios.py -p BoardX58Ich10 -t VS20XX
```

2. Copy SampleApp.efi from the build directory to the VHD Disk

```
Copy ..\Build\SimicsOpenBoardPkg\BoardX58Ich10\DEBUG_VS20XX\X64\SampleApp.efi UefiAppLab Disk
```

3. Run the Simics QSP script from Windows Command Prompt :

```
$> .\simics targets/qsp-x86/qsp-modern-core.simics  
simics> run
```

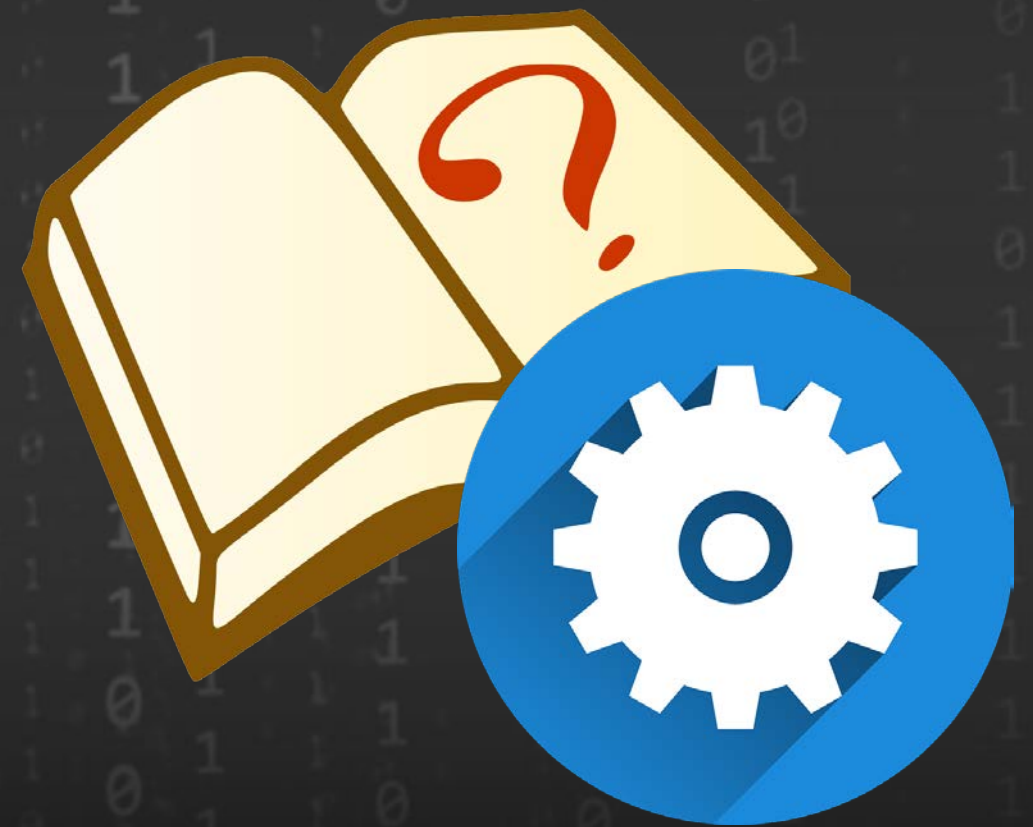
4. At the UEFI Shell prompt

```
Shell> Fs1:  
FS1:\> SampleApp.efi  
System Table: 0x0DEFED018  
Press any key to continue:
```

5. Exit Simics simics> stop, simics> quit

Lab 5: Creating a Simple Typewriter Function

In this lab, you'll learn how to create a simple typewriter function that retrieves the keys you type and subsequently prints each one back to the console



Lab 5 : Typewriter Function

Create a Simple Typewriter Function using the SampleApp from Lab 4

Requirements:

- If the Typewriter Function is enabled
- Retrieve keys entered from keyboard (*Like* Lab 4)
- Print back each key entered to the console
- To exit, press “.” (DOT) and then <Enter>



```
Shell> fs1:  
FS1:\> SampleApp  
System Table: 0xDEFED018
```

```
Press any Key to continue :  
Enter text. Include a dot ('.') in a sentence then <Enter> to exit:  
this is the first line of the typewriter feature function.
```

```
FS1:\> .
```

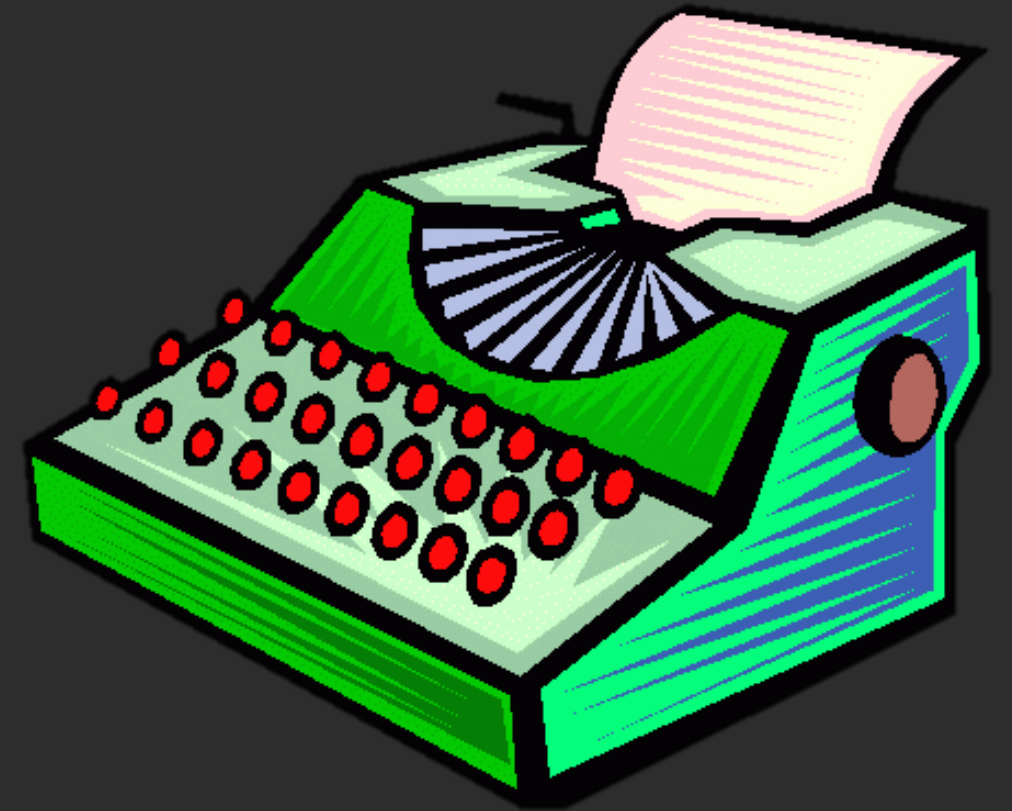


Lab 5 : Typewriter Function

Create a Simple Typewriter Function using the SampleApp from Lab 4

How:

1. Create a Feature Flag PCD to Enable
2. Add a Loop using `WaitForEvent` with `WaitForKey`
3. Use the `ReadKeyStroke` function from `ConIn`
4. Print back each key to console
5. Exit when DOT “.” character is followed by an `<Enter>` key



Lab 5: How Process (Hints)

1. Refer to Lab 1 for How PCDs work and Create a PCD in the Platform DEC File.
2. Use the same procedure as with Lab 4 to find “ReadKeyStroke” in the workspace: [MdePkg/Library/UefiLib/Console.c](#) ~ ln 552

```
Status = gST->ConIn->ReadKeyStroke (gST->ConIn, Key);
```

3. Function ReadKeyStroke uses buffer called EFI_INPUT_KEY ~ ln 393

```
OUT EFI_INPUT_KEY *Key,
```

- TIP: Good Idea to zero out a buffer in your function –
 - Use MdePkg.chm to find ZeroMem function
 - Use ZeroMem on your variable buffer “Key” of type EFI_INPUT_KEY
- 4. Use Boolean flag “ExitLoop” to exit your loop once the user enters a DOT “.” character.

Lab 5: How Process (Hints)

How to Create a Feature Flag PCD

1. Check the MdeModulePkg/MdeModulePkg.dec and search for the “PcdHelloWorldPrintEnable” PCD.
 - Notice that it is in the [PcdsFeatureFlag] section
2. Add a similar section in MyPkg, edk2/MyPkg/MyPkg.dec file and add a flag “PcdTypeWriterFeatureEnable” default, TRUE. (it can be added to the end of the file)
3. Next Update the SampleApp.inf to include: MyPkg/MyPkg.dec, the new PCD, and the PcdLib(hint, see how HelloWorld.inf used the PCD enable)
4. SampleApp.c can now use the new created PCD

Lab 5: Solution: Dec & Inf

MyPkg/MyPkg.dec

```
[PcdsFeatureFlag]
## Indicates if SampleApp Application will enable the Typewriter Feature.
# This PCD is a sample to explain FeatureFlag PCD usage.<BR><BR>
# TRUE - SampleApp Application will enable the Typewriter Feature.<BR>
# FALSE - SampleApp Application will not enable the Typewriter Feature.<BR>
# @Prompt Enable SampleApp enable the Typewriter Feature.
gEfiMyPkgTokenSpaceGuid.PcdTypeWriterFeatureEnable|TRUE|BOOLEAN|0x0001200a
```

SampleApp.inf

```
. . .
[Packages]
MdePkg/MdePkg.dec
MyPkg/MyPkg.dec

[LibraryClasses]
UefiBootServicesTableLib
UefiApplicationEntryPoint
UefiLib
DebugLib
PcdLib

[FeaturePcd]
gEfiMyPkgTokenSpaceGuid.PcdTypeWriterFeatureEnable ## CONSUMES
```

Lab 5: Solution

(hint: Lesson B.5 has the solution)

SampleApp.c - Notepad

File Edit Format View Help

```
#include <Uefi.h>
#include <Library/UefiApplicationEntryPoint.h>
#include <Library/UefiLib.h>
#include <Library/BaseMemoryLib.h>
#include <Library/UefiBootServicesTableLib.h>
#define CHAR_DOT 0x002E // '.' in Unicode

EFI_STATUS
EFIAPI
UefiMain (
    IN EFI_HANDLE      ImageHandle,
    IN EFI_SYSTEM_TABLE *SystemTable
)
{
    UINTN      EventIndex;
    BOOLEAN    ExitLoop;
    EFI_INPUT_KEY Key;

    // Lab 3
    Print(L"System Table: 0xp\n",SystemTable);

    //Lab 4
    Print( L"\nPress any Key to continue : \n");
    gBS->WaitForEvent (1, &gST->ConIn->WaitForKey,EventIndex);
```

```
// Lab 5
gST->ConIn->ReadKeyStroke (gST->ConIn, &Key);
if (FeaturePcdGet(PcdTypeWriterFeatureEnable)) {
    Print(L"Enter text. Include a dot ('.') in a \
        sentence then <Enter> to exit:\n\n");
    ZeroMem (&Key, sizeof (EFI_INPUT_KEY));
    ExitLoop = FALSE;
    do {
        gBS->WaitForEvent (1, &gST->ConIn->WaitForKey,
            &EventIndex);
        gST->ConIn->ReadKeyStroke (gST->ConIn, &Key);
        Print(L"%c", Key.UnicodeChar);
        if (Key.UnicodeChar == CHAR_DOT){
            ExitLoop = TRUE;
        }
    } while (!(Key.UnicodeChar == CHAR_CARRIAGE_RETURN) ||
        !(ExitLoop) );
}
Print(L"\n");
return EFI_SUCCESS;
}
```

Lab 5 : Build and Test SampleApp

1. At the VS Command Prompt, Re-Build BoardX58Ich10

```
$> Cd C:\FW\edk2-ws\edk2-platforms\Platform\Intel\  
$> python build_bios.py -p BoardX58Ich10 -t VS20XX
```

2. Copy SampleApp.efi from the build directory to the VHD Disk

Copy ..\Build\SimicsOpenBoardPkg\BoardX58Ich10\DEBUG_VS20XX\X64\SampleApp.efi UefiAppLab Disk

3. Run the Simics QSP script from Windows Command Prompt :

```
$> .\simics targets/qsp-x86/qsp-modern-core.simics  
simics> run
```

4. Run SampleApp

```
Shell> fs1:  
FS1:\> SampleApp  
System Table: 0xDEDED018  
  
Press any Key to continue :  
Enter text. Include a dot ('.') in a sentence then <Enter> to exit:  
this is the first line of the typewriter feature function.  
FS1:\> _
```

5. Exit Simics `simics> stop`, `simics> quit`

Lab 5 : Bug Reports

You received the following bug reports on your SampleApp UEFI Application

1. If an “Enter” Key is pressed before the “.” character, there is no line feed to go to a new line like a real typewriter would do.
2. Since the DOT “.” is used in sentences a different character other than a “.” is required to exit the Typewriter function. One suggestion was to use the “ESC” Character Instead.
3. Some customers would like the application without the typewriter feature enabled.

How would you add these fixes to the SampleApp UEFI application?

Lab 5.1 : Bug Fixes

(hint: Lesson B.5. has the solution)

Add a Line Feed Character after a “Enter” key.

1. Update SampleApp.c with this fix (hint, print a CHAR_LINEFEED when a Carriage return is entered)

2. At the VS Command Prompt, Re-Build BoardX58Ich10

```
$> Cd C:\FW\edk2-ws\edk2-platforms\Platform\Intel\  
$> python build_bios.py -p BoardX58Ich10 -t VS20XX
```

3. Copy SampleApp.efi from the build directory to the VHD Disk

Copy ..\Build\SimicsOpenBoardPkg\BoardX58Ich10\DEBUG_VS20XX\X64\SampleApp.efi UefiAppLab Disk

4. Run the Simics QSP script from Windows Command Prompt :

```
$> .\simics targets/qsp-x86/qsp-modern-core.simics  
simics> run
```

5. Run SampleApp

6. Exit Simics `simics> stop`, `simics> quit`

```
Shell> fs1:  
FS1:\> SampleApp  
System Table: 0xDEDED018  
  
Press any Key to continue :  
Enter text. Include a dot ('.') in a sentence then <Enter> to exit:  
This is the First Line  
This is the second Line.  
  
FS1:\> _
```


Lab 5.2 : Bug Fixes

Use the Scan Code for ESC instead of the DOT Character to Exit the Typewriter Function. (Hint, the ExitLoop flag will no longer be needed, Hint, Search workspace for SCAN_ESC)

1. Update SampleApp.c to use SCAN_ESC to exit the do-loop

2. At the VS Command Prompt, Re-Build BoardX58Ich10

```
$> Cd C:\FW\edk2-ws\edk2-platforms\Platform\Intel\  
$> python build_bios.py -p BoardX58Ich10 -t VS20XX
```

3. Copy SampleApp.efi from the build directory to the VHD Disk

```
Copy ..\Build\SimicsOpenBoardPkg\BoardX58Ich10\DEBUG_VS20XX\X64\SampleApp.efi UefiAppLab Disk
```

4. Run the Simics QSP script from Windows Command Prompt :

```
$> .\simics targets/qsp-x86/qsp-modern-core.simics  
simics> run
```

5. Run SampleApp

6. Exit Simics `simics> stop`, `simics> quit`

```
Shell> fs1:  
FS1:\> sampleApp  
System Table: 0xDEDED018  
  
Press any Key to continue :  
Enter text as in a typewriter then type <ESC> to exit  
This is line 1.  
This is line 2.    Now press "ESC" Key  
FS1:\> _
```

Lab 5.3 : Bug Fixes

Make the PCD PcdTypeWriterFeatureEnable determined by a Build Flag

1. Update OpenBoardPkg.dsc (about line 45)

```
DEFINE ADD_TYPEWRITTER = TRUE
```

2. Update OpenBoardPkgPcd.dsc (at end add [Packages] MyPkg/MyPkg.dec and [PcdsFeatureFlag] Section)

```
[Packages]
  MyPkg/MyPkg.dec

[PcdsFeatureFlag]
!if $(ADD_TYPEWRITTER) == TRUE
  gMyPkgTokenSpaceGuid.PcdTypeWriterFeatureEnable|TRUE
!else
  gMyPkgTokenSpaceGuid.PcdTypeWriterFeatureEnable|FALSE
!endif
```

FALSE

3. At the VS Command Prompt, Re-Build BoardX58Ich10

```
$> Cd C:\FW\edk2-ws\edk2-platforms\Platform\Intel\
$> python build_bios.py -p BoardX58Ich10 -t VS20XX
```

```
FS1:\> sampleapp
System Table: 0xDEDED018

Press any Key to continue :
Typewriter feature not enabled

FS1:\> _
```

4. Copy SampleApp.efi from the build directory to the VHD Disk

Copy ..\Build\SimicsOpenBoardPkg\BoardX58Ich10\DEBUG_VS20XX\X64\SampleApp.efi UefiAppLab Disk

TRUE

5. Run the Simics QSP script from Terminal Command Prompt:

```
$> .\simics targets/qsp-x86/qsp-modern-core.simics
simics> run
```

6. Run SampleApp

```
FS1:\> sampleapp
System Table: 0xDEDED018

Press any Key to continue :
Enter text as in a typewriter then type <ESC> to exit
line 1.
line 2.
Now the ESC key
FS1:\> _
```

7. Exit Simics simics> stop, simics> quit

Bonus Exercise: Open Protocol Example

Write an Application using `argv`, `argc` parameters

- Captures command line parameters using Open Protocol
- Need to open SHELL_INTERFACE_PROTOCOL
- Note: Requires ShellPkg

Build SampleApp and copy to the VHD Drive

Run the application from the shell in Simics

```
Shell>fs1:  
FS1:/> SampleApp test1 test2
```

(hint: ~FW/LabSampleCode/ShellAppSample has the solution)

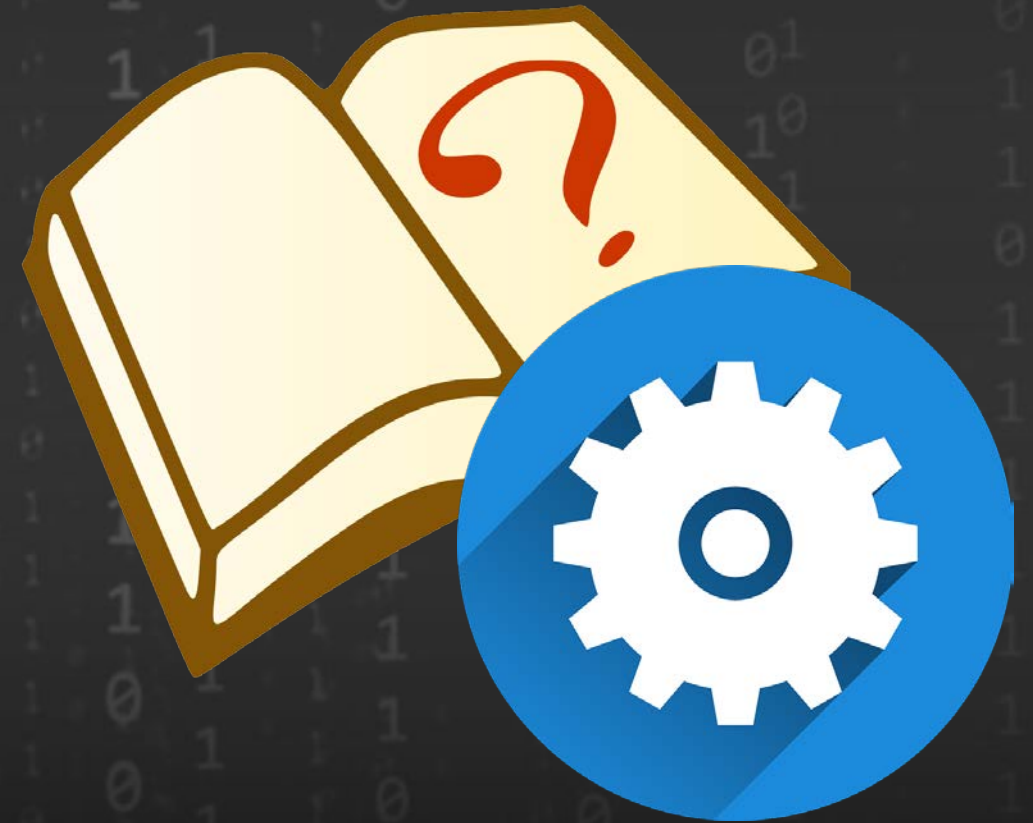
USING EADK

Using EADK with UEFI Application

Labs 6-7 are Optional

Lab 6: Writing UEFI Applications with EADK

In this lab, you'll write an application with the same functionality as SampleApp.c using LibC from the EDK II Application Development Kit (EADK)



Lab 6: With EDK II EADK

Write the same application with the same functionality as SampleApp.c using the LibC from the EADK

```
Shell> fs0:  
FS0:\> SampleCApp  
System Table: 0x631bf90  
  
Press any Key and then <Enter> to continue :  
  
Enter text. Include a dot ('.') in a sentence then <Enter> to exit:  
  
This is a sentence using my UEFI Application using the C library.  
  
FS0:\> _
```

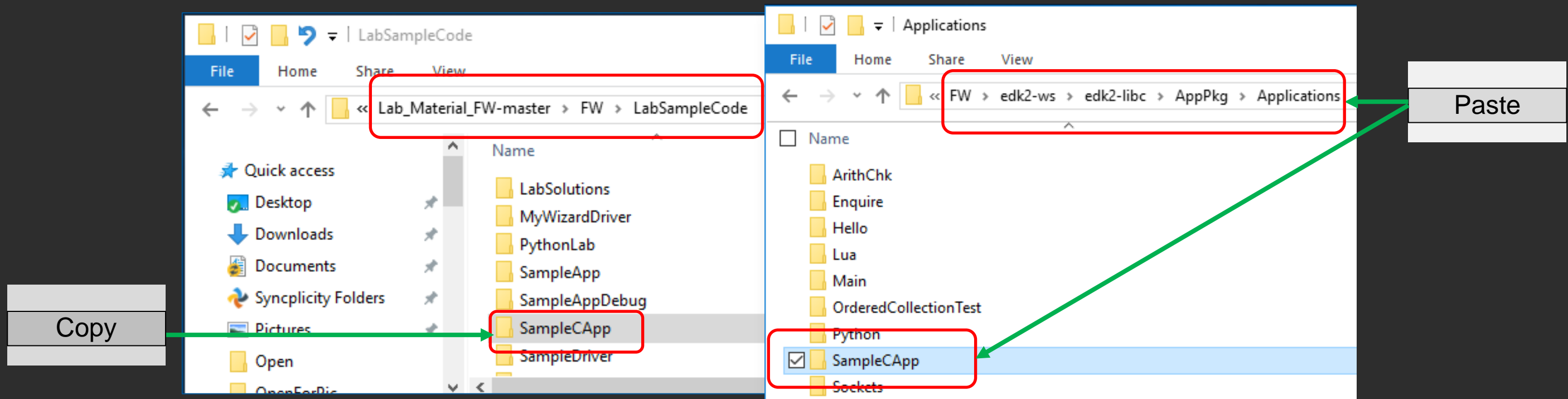
What libraries are needed

What differences are there using the LibC

Lab 6: EDK II using EADK

Start with the packages for EADK from edk2-libc

- /edk2-libc - AppPkg - has directory Applications
- /edk2-libc - StdLib - contains the LibC libraries
- Copy and paste directory ../FW/LabSampleCode/**SampleCApp** to
C:/FW/edk2-ws/edk2-libc/AppPkg/Applications/**SampleCApp**



Lab 6: EDK II using EADK

Check out AppPkg/Applications/SampleCApp
SampleCApp.c and SampleCApp.inf

```
SampleCApp.c - Notepad
File Edit Format View Help

#include <stdio.h>
// . . .
int
main (
    IN int Argc,
    IN char **Argv
)
{
    return 0;
}
```

```
SampleCApp.inf - Notepad
File Edit Format View Help

[Defines]
  INF_VERSION          = 1.25
  BASE_NAME            = SampleCApp
  FILE_GUID            = 4ea9...
  MODULE_TYPE          = UEFI_APPLICATION
  VERSION_STRING       = 0.1
  ENTRY_POINT          = ShellCEntryLib

[Sources]
  SampleCApp.c

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

[LibraryClasses]
  LibC
  LibStdio
```


Lab 6 : Update AppPkg.dsc

Edit the C:/fw/edk2-ws/edk2-libc/AppPkg/AppPkg/AppPkg.dsc and add SampleCApp.inf at the end of the components section

- (hint: search for "#### Sample Applications")
- AppPkg/Applications/SampleCApp/SampleCApp.inf

```
[Components]
#### Sample Applications.
AppPkg/Applications/Hello/Hello.inf           # No LibC includes or functions.
AppPkg/Applications/Main/Main.inf             # Simple invocation. No other LibC function
AppPkg/Applications/Enquire/Enquire.inf       #
AppPkg/Applications/ArithChk/ArithChk.inf     #
AppPkg/Applications/SampleCApp/SampleCApp.inf # LAB 6
```

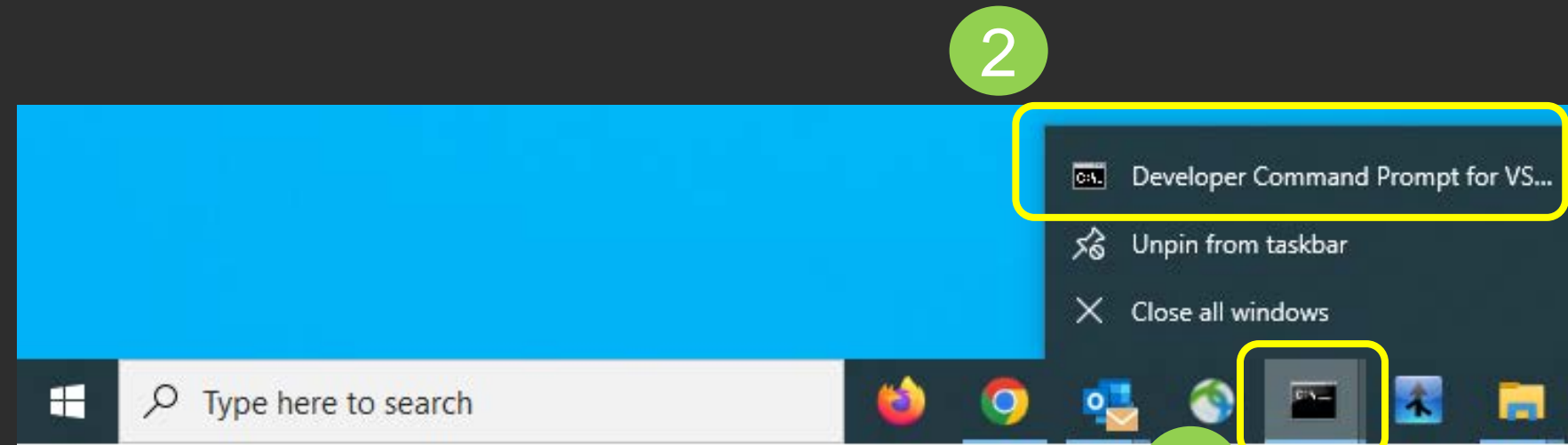
Open Another VS Command Prompt

1. Right Click on the Task tray VS Command prompt Icon to Open another Visual Studio Command Prompt

2. Left Click on “Developer . . .” (this opens another window)

3. Then :

```
$> cd C:\fw\edk2-ws\
$> setenv.bat
$> cd edk2
$> edksetup.bat
```



A screenshot of a Windows Command Prompt window titled 'Developer Command Prompt for VS2015'. The window has a black background with white text. A yellow box labeled '3' is around the command prompt. The text in the window shows the following commands and their outputs:

```
C:\Program Files (x86)\Microsoft Visual Studio 14.0>cd\
C:\>cd fw\edk2-ws
C:\FW\edk2-ws>setenv.bat
C:\FW\edk2-ws>set WORKSPACE=C:\FW\edk2-ws
C:\FW\edk2-ws>set PACKAGES_PATH=C:\FW\edk2-ws\edk2;C:\FW\edk2-ws\edk2-libc
Removing Directory C:\FW\edk2-ws\Conf
Next "CD edk2" then run "EdkSetup"
C:\FW\edk2-ws>cd edk2
C:\FW\edk2-ws\edk2>edksetup
```

Lab 6 :Build and Test SampleCApp

1. Build the AppPkg at the VS Command Prompt

```
$> build -p AppPkg\AppPkg.dsc -m AppPkg\Applications\SampleCApp\SampleCApp.inf
```

2. Copy the built application SampleCApp.efi to the **VHD Drive** (note VS Tool)

```
$> copy C:\fw\edk2-ws\Build\AppPkg\DEBUG_VS20XX\X64\SampleCApp.efi (UefiAppLab Disk)
```

3. Run the Simics QSP script from Windows Command Prompt :

```
$> .\simics targets/qsp-x86/qsp-modern-core.simics
```

```
simics> run
```

4. Run SampleCApp

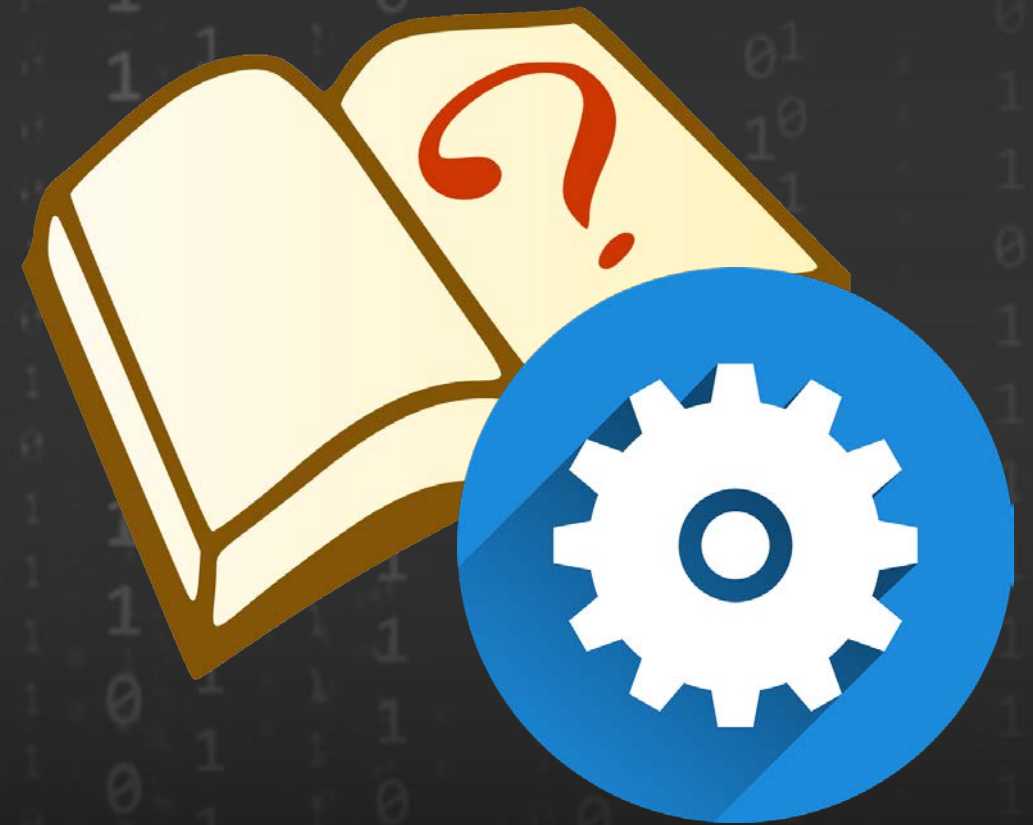
```
Shell> fs1:  
FS1:\> SampleCApp.efi  
FS1:\> _
```

5. Exit Simics `simics> stop`, `simics> quit`

Notice that the program will immediately unload because the main function is empty

Lab 7: Adding Functionality to SampleCApp

In this lab, you'll add functionality to SampleCApp the same as in Lab 5. This lab will use EADK libraries, so the coding style is similar to standard C.



Lab 7: Add Feature PCD

AppPkg/AppPkg.dec

```
[PcdsFeatureFlag]
## Indicates if SampleCApp Application will enable the Typewriter Feature.
# This PCD is a sample to explain FeatureFlag PCD usage.<BR><BR>
# TRUE - SampleCApp Application will enable the Typewriter Feature.<BR>
# FALSE - SampleCApp Application will not enable the Typewriter Feature.<BR>
# @Prompt Enable SampleCApp enable the Typewriter Feature.
gAppPkgTokenSpaceGuid.PcdTypeWriterFeatureEnable|TRUE|BOOLEAN|0x1200a
```

SampleCApp.inf

```
. . .
[Packages]
. . .
[FeaturePcd]
gAppPkgTokenSpaceGuid.PcdTypeWriterFeatureEnable ## CONSUMES
```

Lab 7: Add the same functionality from Lab 5

SampleCApp.c

and

SampleCApp.inf

```
SampleCApp.c - Notepad
File Edit Format View Help

#include <stdio.h>
#include <Library/PcdLib.h>
#include <Library/UefiBootServicesTableLib.h>
// . . .
char c;
printf("System Table: %p \n", gST) ;

puts("Press any Key and then <Enter>
to continue : ");
c=(char)getchar();

if (FeaturePcdGet(PcdTypeWriterFeatureEnable)) {
    puts ("Enter text. Include a dot('.') in a
sentence then <Enter> to exit:");
    do {
        c=(char)getchar();
    } while (c != '.');
}
puts ("\n");

return 0;
}
```

```
SampleCApp.inf - Notepad
File Edit Format View Help

[Defines]
    INF_VERSION          = 1.25
    BASE_NAME            = SampleCApp
    FILE_GUID            = 4ea9...
    MODULE_TYPE          = UEFI_APPLICATION
    VERSION_STRING       = 0.1
    ENTRY_POINT          = ShellCEntryLib

[Sources]
    SampleCApp.c

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

[LibraryClasses]
    LibC
    LibStdio
    UefiBootServicesTableLib
    PcdLib

[FeaturePcd]
    gAppPkgTokenSpaceGuid.PcdTypeWriterFeatureEnable
```

Lab 7: Add the same functionality from Lab 5

SampleCApp.c

and

SampleCApp.inf

```

SampleCApp.c - Notepad
File Edit Format View Help

#include <stdio.h>
#include <Library/PcdLib.h>
#include <Library/UefiBootServicesTableLib.h>
// . . .
char c;
printf("System Table: %p \n", gST) ;

puts("Press any Key and then <Enter>
to continue : ");
c=(char)getchar();

if (FeaturePcdGet(PcdTypeWriterFeatureEnable)) {
    puts ("Enter text. Include a dot('.') in a
sentence then <Enter> to exit:");
    do {
        c=(char)getchar();
    } while (c != '.');
}
puts ("\n");

return 0;
}

```

3

```

SampleCApp.inf - Notepad
File Edit Format View Help

[Defines]
    INF_VERSION           = 1.25
    BASE_NAME             = SampleCApp
    FILE_GUID             = 4ea9...
    MODULE_TYPE           = UEFI_APPLICATION
    VERSION_STRING        = 0.1
    ENTRY_POINT           = ShellCEntryLib

[Sources]
    SampleCApp.c

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

[LibraryClasses]
    LibC
    LibStdio
    UefiBootServicesTableLib
    PcdLib

[FeaturePcd]
    gAppPkgTokenSpaceGuid.PcdTypeWriterFeatureEnable

```


Lab 7: Add the same functionality from Lab 5

SampleCApp.c

and

SampleCApp.inf

```
SampleCApp.c - Notepad
File Edit Format View Help

#include <stdio.h>
#include <Library/PcdLib.h>
#include <Library/UefiBootServicesTableLib.h>
// . . .
char c;
printf("System Table: %p \n", gST) ;

puts("Press any Key and then <Enter>
to continue : ");
c=(char)getchar();

if (FeaturePcdGet(PcdTypeWriterFeatureEnable)) {
    puts ("Enter text. Include a dot('.') in a
sentence then <Enter> to exit:");
    do {
        c=(char)getchar();
    } while (c != '.');
}
puts ("\n");

return 0;
}
```

3

4

```
SampleCApp.inf - Notepad
File Edit Format View Help

[Defines]
    INF_VERSION          = 1.25
    BASE_NAME             = SampleCApp
    FILE_GUID             = 4ea9...
    MODULE_TYPE           = UEFI_APPLICATION
    VERSION_STRING        = 0.1
    ENTRY_POINT           = ShellCEntryLib

[Sources]
    SampleCApp.c

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

[LibraryClasses]
    LibC
    LibStdio
    UefiBootServicesTableLib
    PcdLib

[FeaturePcd]
    gAppPkgTokenSpaceGuid.PcdTypeWriterFeatureEnable
```


Lab 7: Add the same functionally from Lab 5

SampleCApp.c

and

SampleCApp.inf

```

SampleCApp.c - Notepad
File Edit Format View Help

#include <stdio.h>
#include <Library/PcdLib.h>
#include <Library/UefiBootServicesTableLib.h>
// . . .
char c;
printf("System Table: %p \n", gST) ;

puts("Press any Key and then <Enter>
to continue : ");
c=(char)getchar();

if (FeaturePcdGet(PcdTypeWriterFeatureEnable)) {
    puts ("Enter text. Include a dot ('.') in a
sentence then <Enter> to exit:");
    do {
        c=(char)getchar();
    } while (c != '.');
}
puts ("\n");

return 0;
}

```

```

SampleCApp.inf - Notepad
File Edit Format View Help

[Defines]
    INF_VERSION           = 1.25
    BASE_NAME              = SampleCApp
    FILE_GUID              = 4ea9...
    MODULE_TYPE            = UEFI_APPLICATION
    VERSION_STRING         = 0.1
    ENTRY_POINT            = ShellCEntryLib

[Sources]
    SampleCApp.c

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

[LibraryClasses]
    LibC
    LibStdio
    UefiBootServicesTableLib
    PcdLib

[FeaturePcd]
    gAppPkgTokenSpaceGuid.PcdTypeWriterFeatureEnable

```

3

4

5

Lab 7 :Build and Test SampleCApp

1. Build the AppPkg at the VS Command Prompt

```
$> build -p AppPkg\AppPkg.dsc -m AppPkg\Applications\SampleCApp\SampleCApp.inf
```

2. Copy the built application SampleCApp.efi to the **VHD Drive** (note VS Tool)

```
$> copy C:\fw\edk2-ws\Build\AppPkg\DEBUG_VS20XX\X64\SampleCApp.efi (UefiAppLab Disk)
```

3. Run the Simics QSP script from Windows Command Prompt :

```
$> .\simics targets/qsp-x86/qsp-modern-core.simics
```

```
simics> run
```

4. Run SampleCApp

```
Shell> fs1:
FS1:\> SampleCApp.efi
System Table: 0xdefed018
Press any Key and then <Enter> to continue :

Enter text. Include a dot ('.') in a sentence then <Enter> to exit:
this is line 1.

FS1:\> _
```

5 . Exit Simics `simics> stop`, `simics> quit`

Summary

- ★ UEFI Application with PCDs
- ★ Simple UEFI Application
- ★ Add functionality to UEFI Application
- ★ Using EADK with UEFI Application

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.

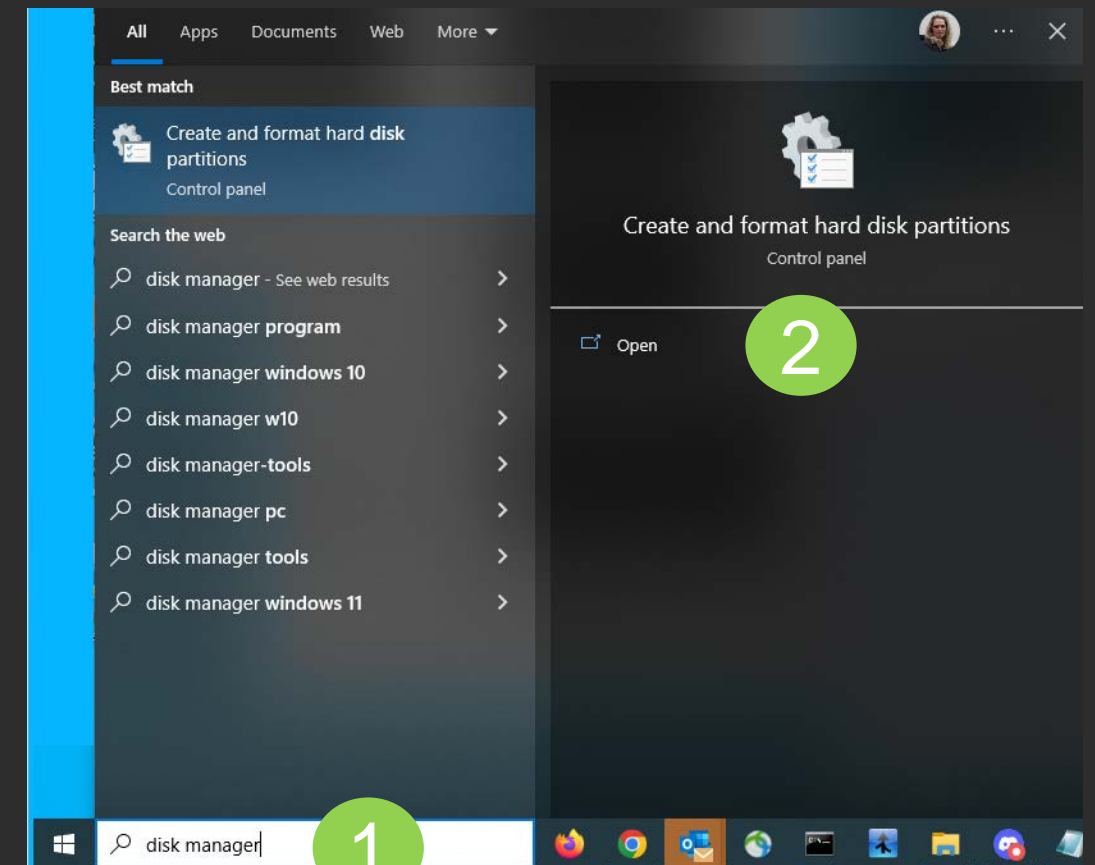
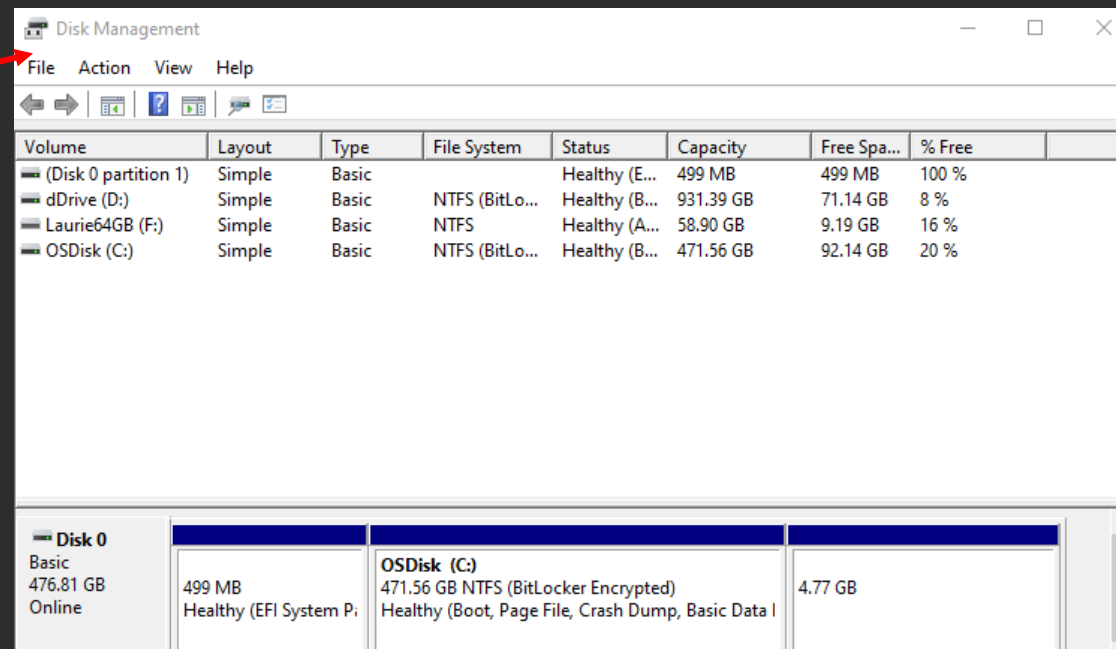
How to Mount a VHD

Open Disk Manager & Attach VHD

Using the Search menu, Lower Left, just right of the  Logo, type:

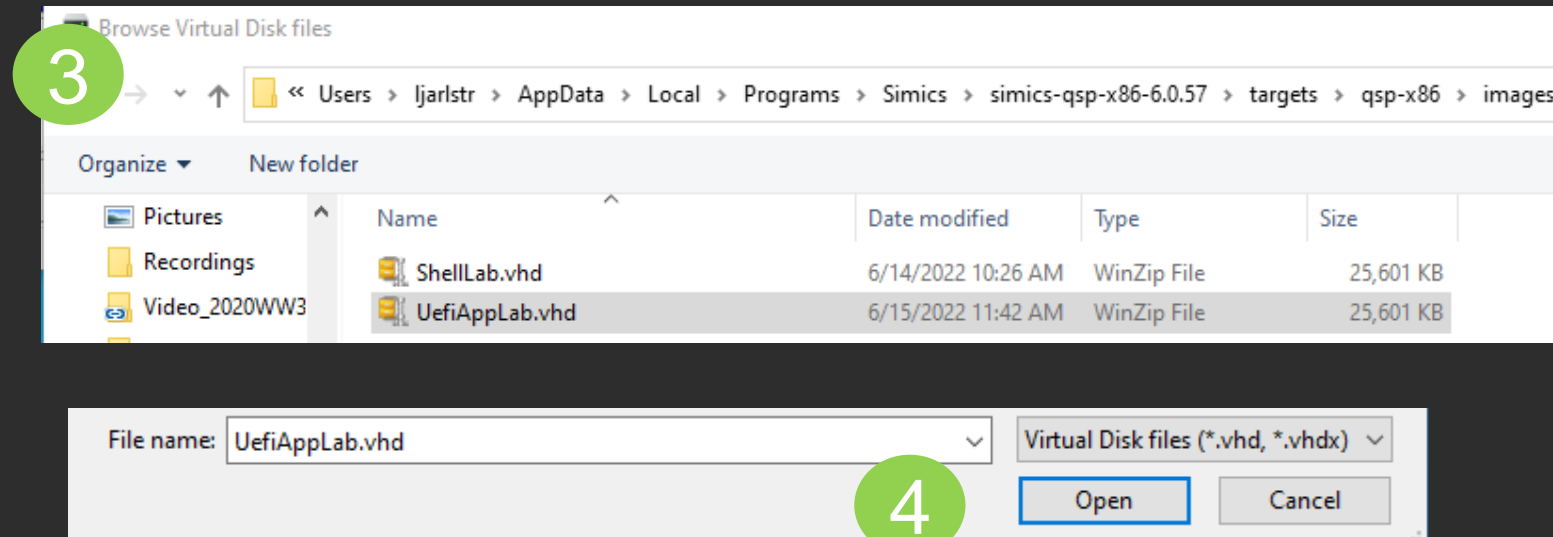
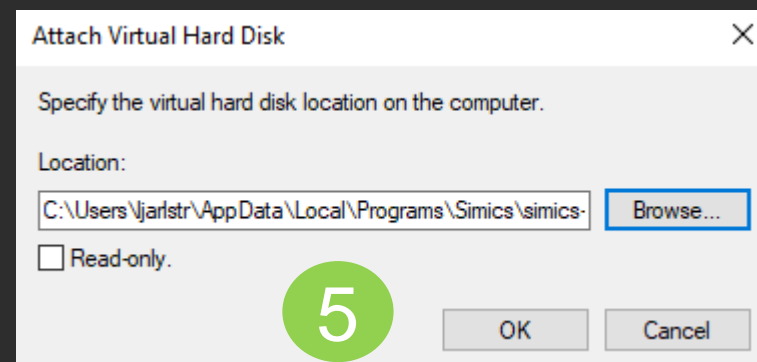
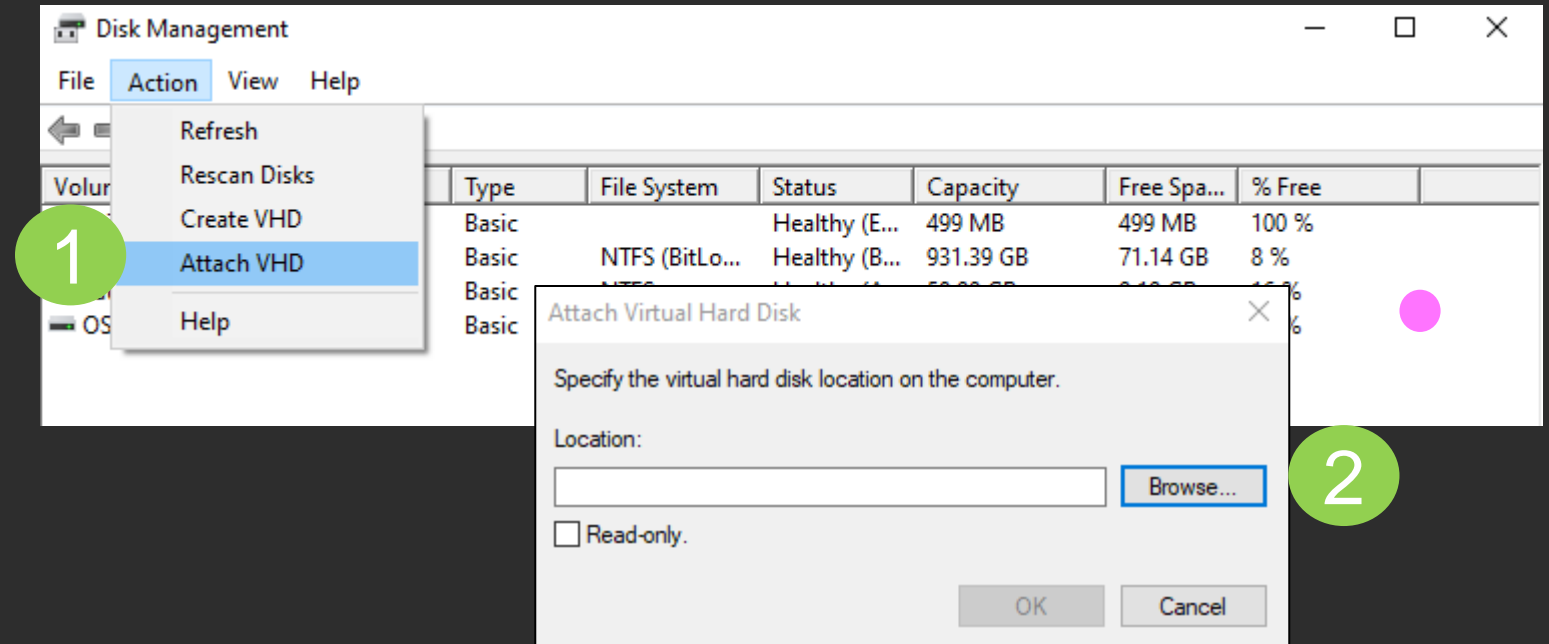
1. "Disk Manager", then open "Create and format hard disk partitions"
2. Select Open (OK on Security menu, if it pops up)

3.



Attach a VHD File

1. Select “Action” then “Attach VHD”** file – This will mount the file as a disk image
2. Select Browse
3. Use File Explorer to Find Desired VHD File
4. Select Open
5. OK



**If “Attach VHD” is greyed out, left click anywhere in the white space in the first window

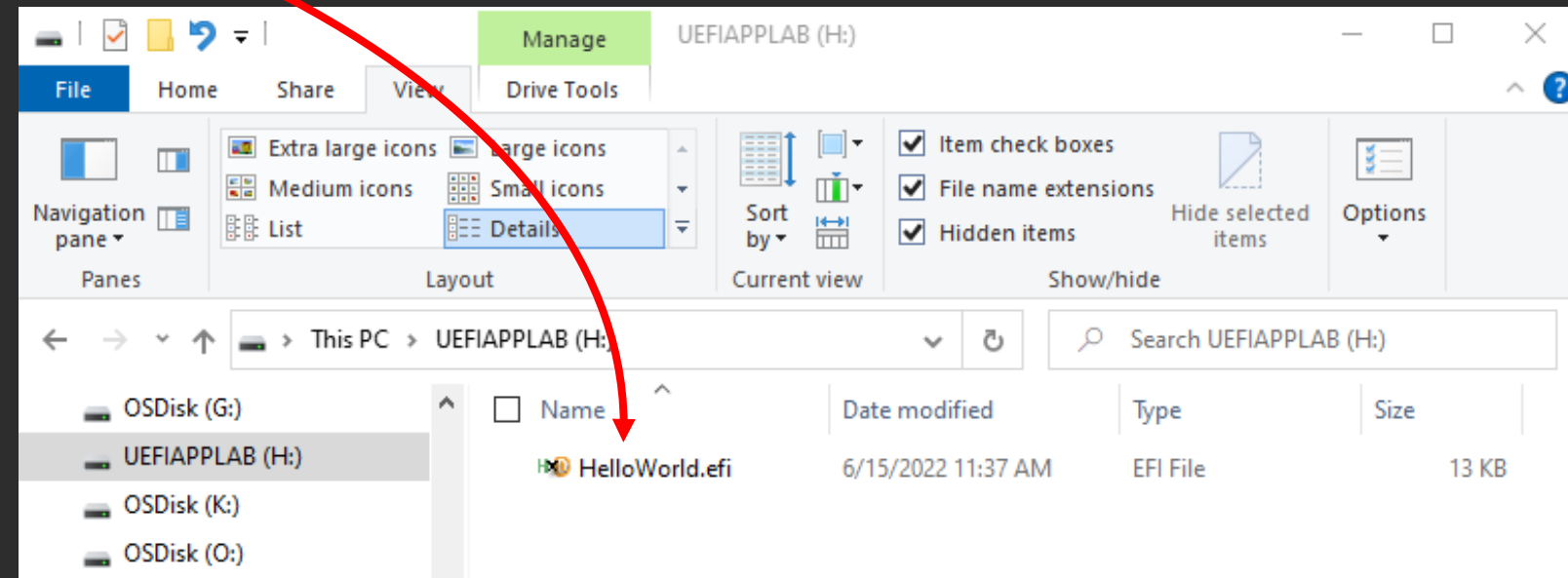
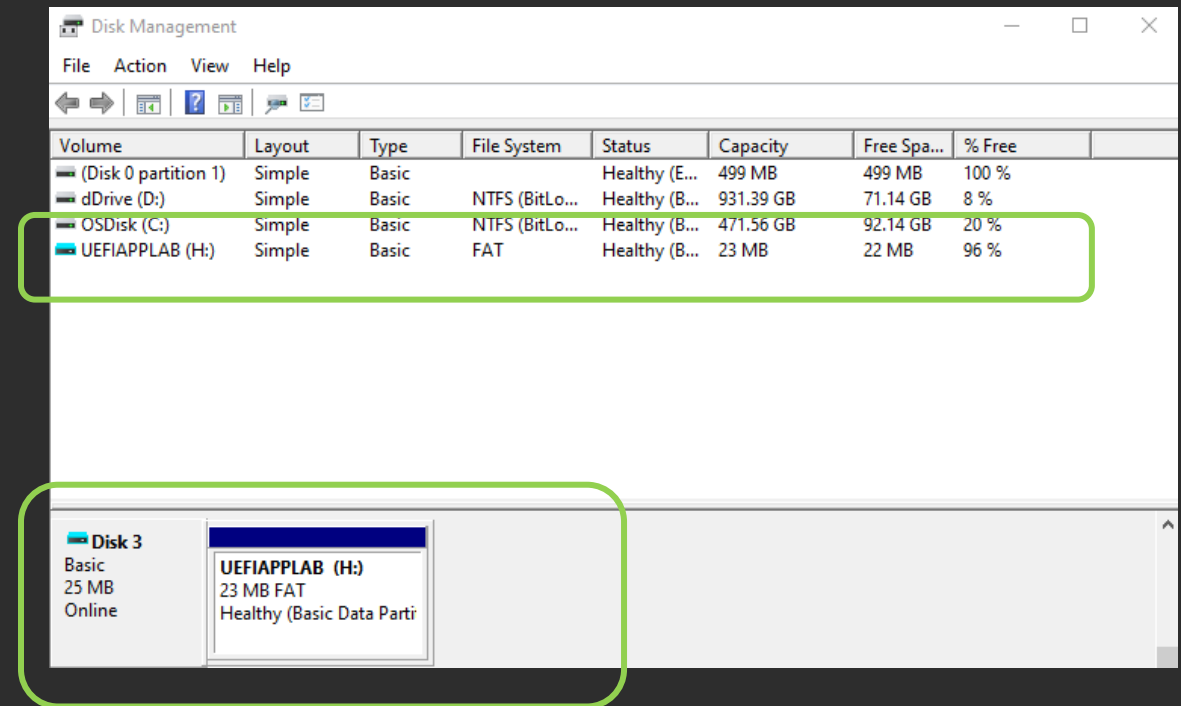
Mounted VHD is Ready

Files from Windows can now be copied to & from the mounted VHD.

Update files Here

Simics Agent for UEFI using Matic

- Note, the VHD method of copying UEFI Applications to use inside the UEFI Shell running Simics works only for coping **TO** the VHD driver.
- If there is data required **FROM** the UEFI Shell running under Simics, It is preferable to use the Simics `simics_agent_efi`



Detach/Dismount the VHD File

Once Finished,

1. Right Click inside left box in Disk Manager
2. Select “Detach VHD”
3. OK

