tianocore

UEFI & EDK II Training

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

tianocore.org

See also LabGuide 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

EDK II PCD's Purpose and Goals

Review

Documentaton : 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

PCD Syntax

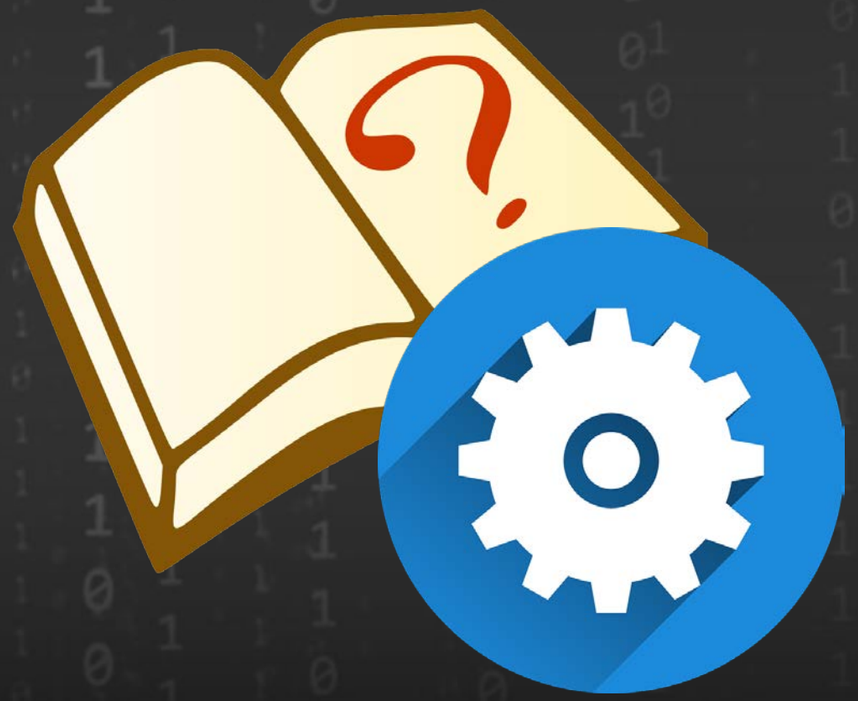
Review

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

.DEC	.INF	.DSC
Define PCD	Reference PCD	Modify PCD
Package	Module	Platform

Lab 1: Writing UEFI Applications with PCDs

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



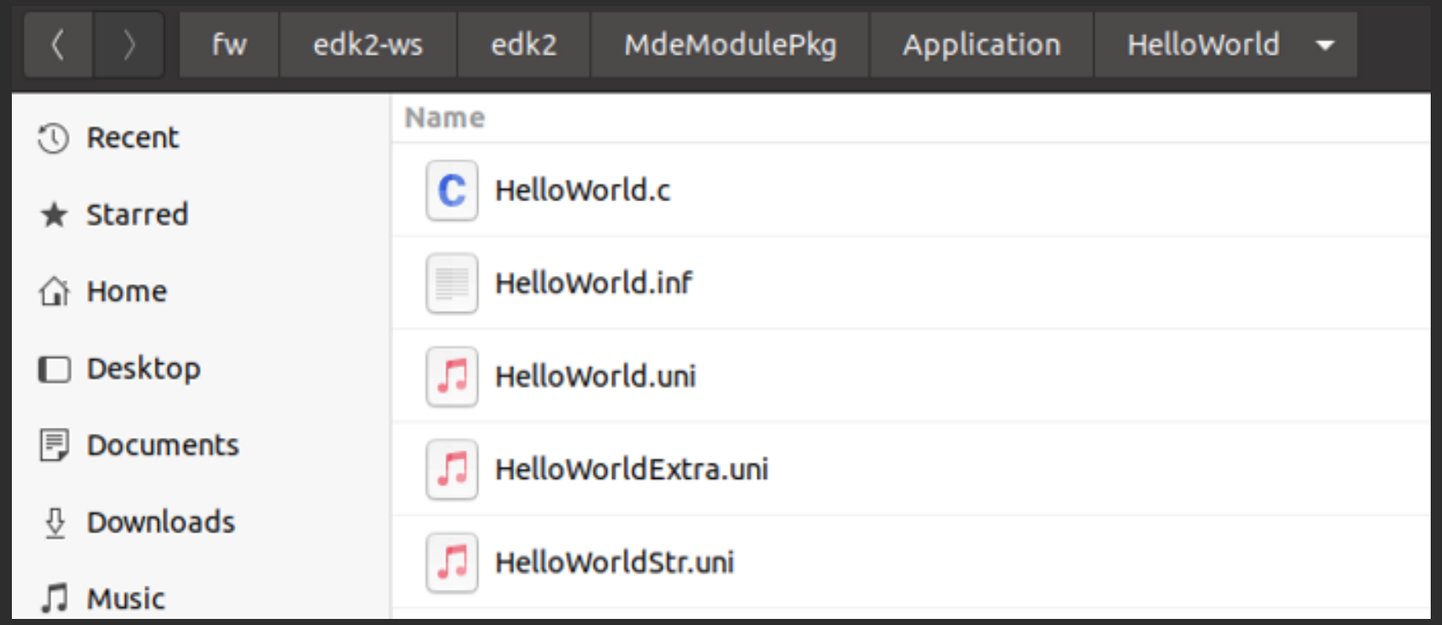
EDK II HelloWorld App Lab

MdeModulePkg/Application/HelloWorld

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

Notice the PCD values

Then Run HelloWorld in Simics



Copy UefiAppLab.vhd file

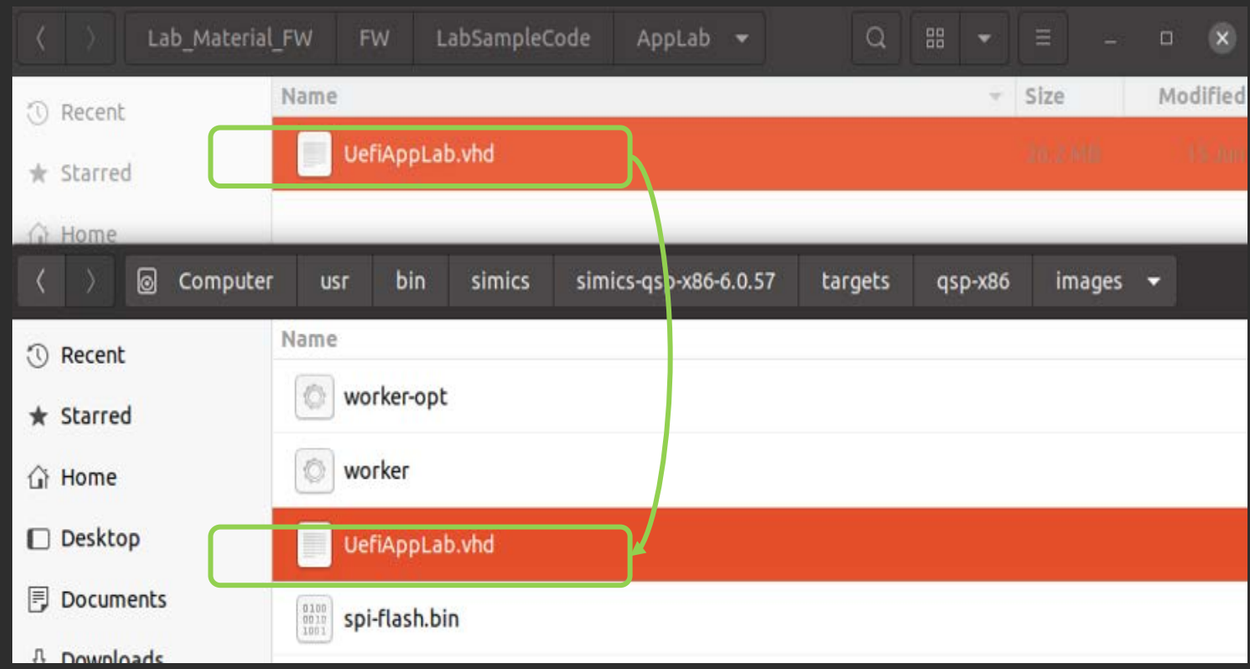
Copy the UefiApplLab.vhd

From:

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

To

<SimicsInstallDir>/simics-qsp-x86-6.0.57/targets/qsp-x86/images



Where *<SimicsInstallDir>* is the directory selected to install Simics, e.g., `Computer/usr/bin/simics`

Update the Simics Script

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

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

<SimicsInstallDir>/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

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

Open another Terminal Prompt in $HOME/fw/edk2-ws
Then CD to edk2 to do edksetup.sh

```
$ cd ~/fw/edk2-ws/edk2
$ . edksetup.sh
```

Then CD to:

```
$ cd ~/fw/edk2-ws/edk2-platforms/Platform/Intel
```

Invoke the Python Build script for Simics OpenBoard QSP

```
$ python build_bios.py -p BoardX58Ich10  -t GCC5
```

Copy

~/fw/edk2-ws/Build/SimicsOpenBoardPkg/BoardX58Ich10/DEBUG_GCC5/FV/BOARDX58ICH10.fd

To

<SimicsInstallDir>/simics-qsp-x86-6.0.57/targets/qsp-x86/images

Invoke Simics & Run HelloWorld App

1. Open a Terminal prompt

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

2. Run the Simics qsp-modern-core 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:/>
```

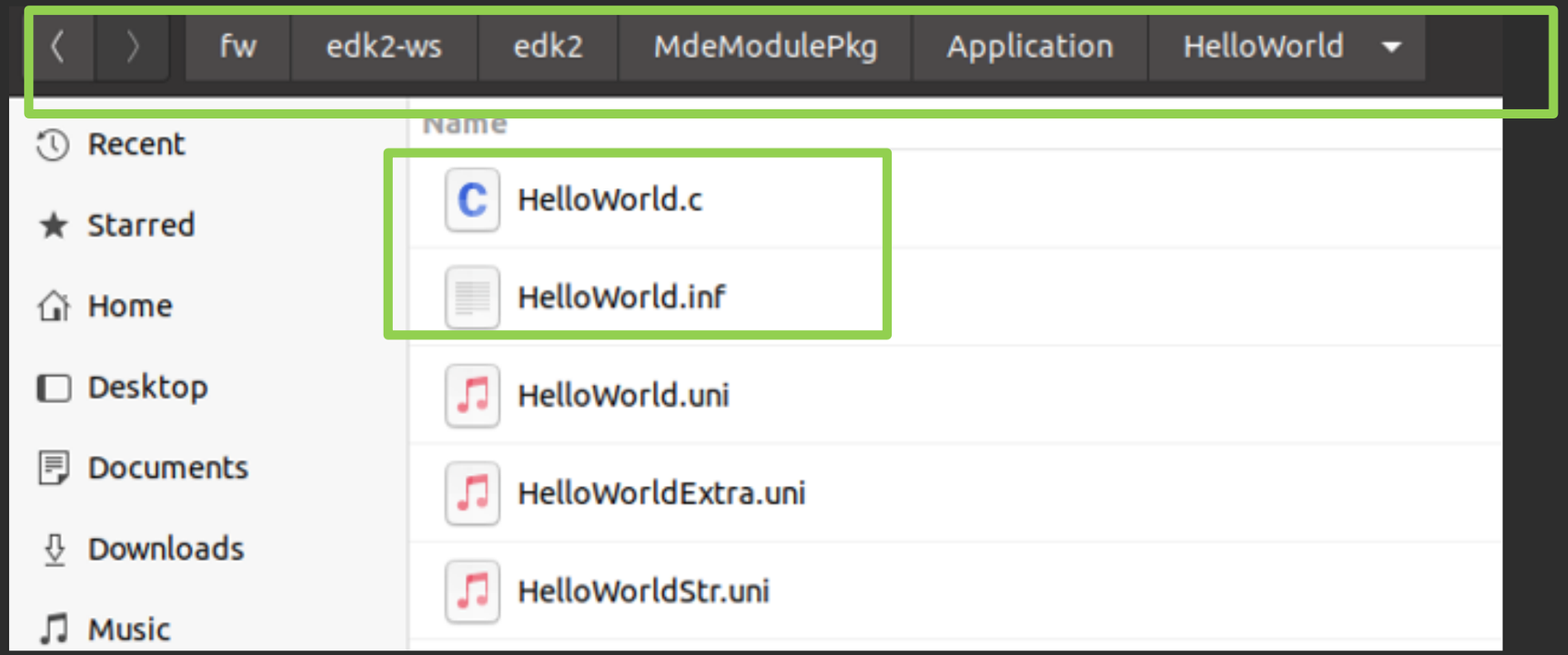
4. Exit Simics

```
simics> stop, simics> quit
```

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

EDK II HelloWorld App Lab

MdeModulePkg/Application/HelloWorld



EDK II HelloWorld App

Source HelloWorld.c

```
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:

~FW/edk2-ws/edk2-platforms/Platform/Intel/SimicsOpenBoardPkg/BoardX58Ich10/OpenBoardPkg.dsc

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

OpenBoardPkg.dsc
~/fw/edk2-ws/edk2-platforms/Platform/Intel/SimicsOpenBoardPkg/BoardX58Ich10/

```
[PcdsFixedAtBuild]
gEfiMdeModulePkgTokenSpaceGuid.PcdHelloWorldPrintTimes|3
```

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

2. Re-Build BoardX58Ich10

Open A Terminal Command Prompt

```
$ Cd ~/FW/edk2-ws/edk2-platforms/Platform/Intel/
$ python build_bios.py -p BoardX58Ich10  -t GCC5
```

Update UefiAppLab.vhd File

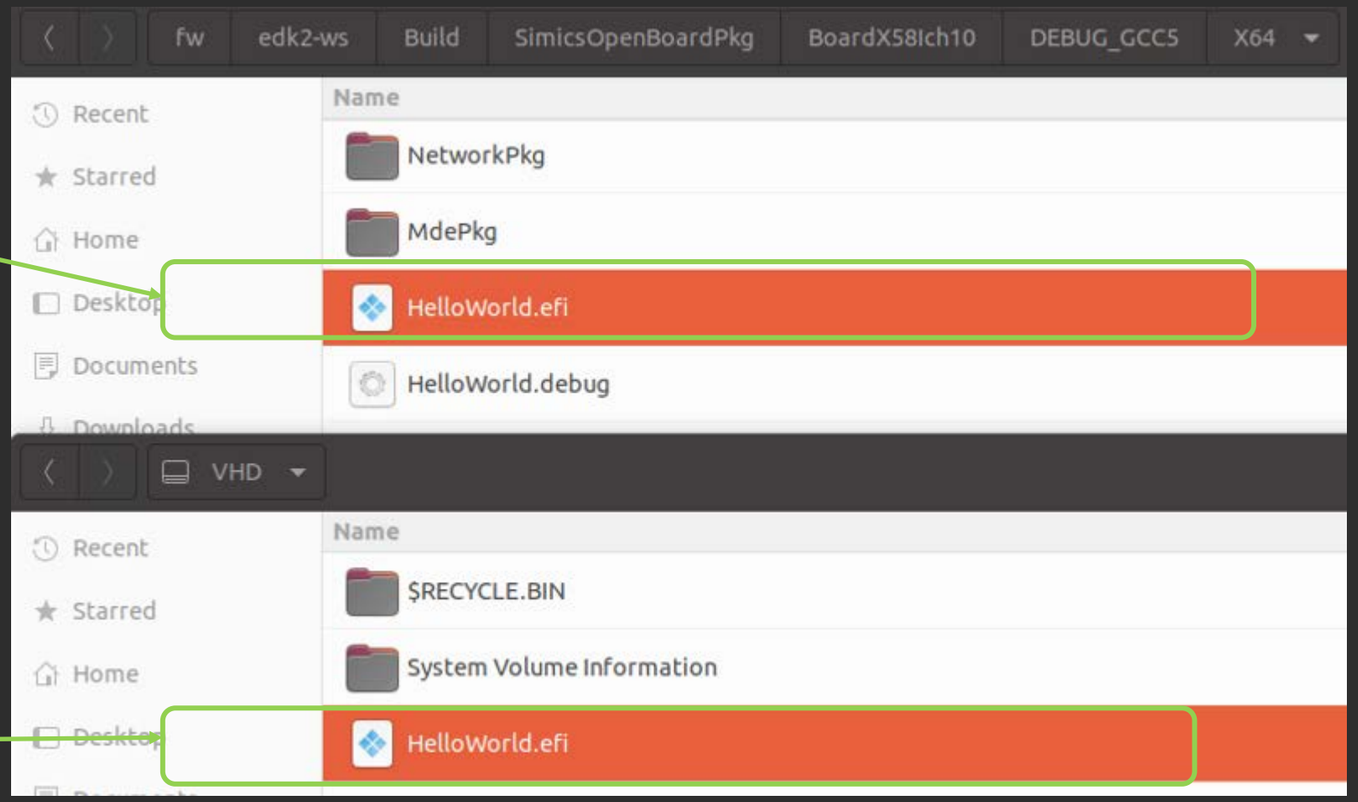
3. Mount the UefiAppLab.vhd using GuestMount: How To Link

4. Copy `HelloWorld.efi`

```
$cp ~/FW/edk2-ws/Build/SimicsOpenBoardPkg/BoardX58Ich10/DEBUG_GCC5/X64/HelloWorld.efi ~/VHD
```

Copy

Paste



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:/>
```

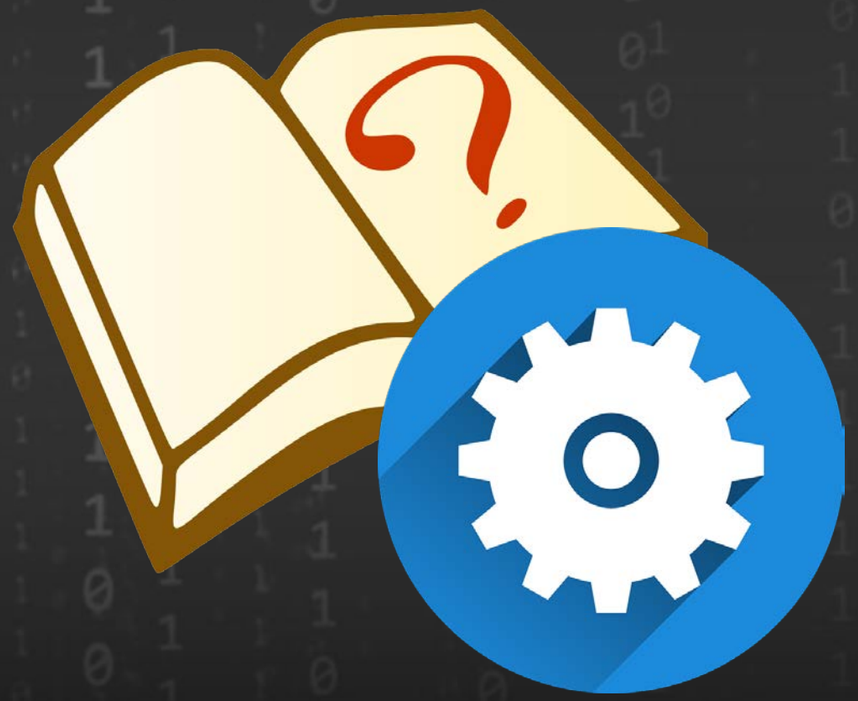
7. Exit Simics `simics> Stop` then `simics> quit`

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

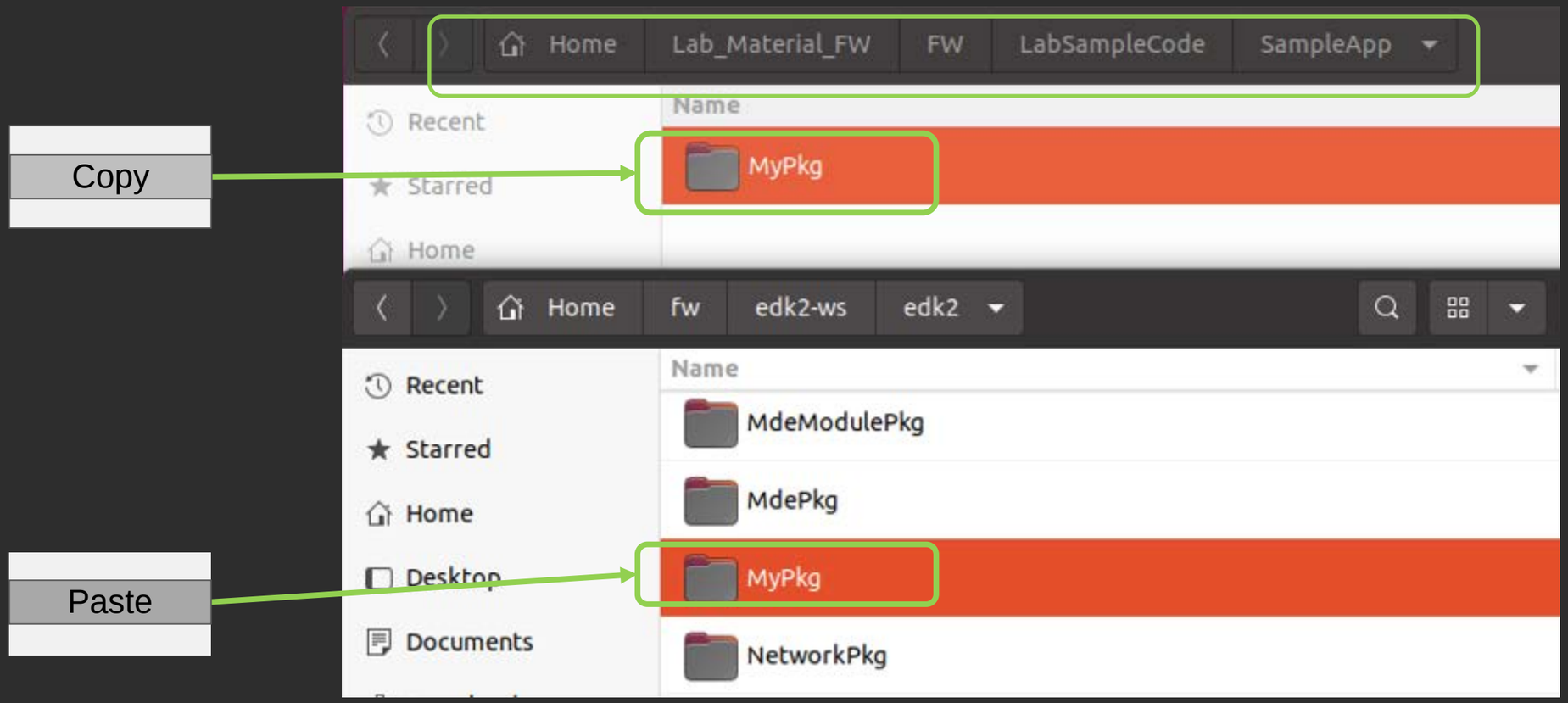
.inf file

```
[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 `~/Lab_Material_FW/LabSampleCode/SampleApp/MyPkg` directory to `~/FW/edk2-ws/edk2`



Edit **`SampleApp.inf`**

- Look in the INF for **"XXXXXXXXXXX"** sections that will need information
- Create Name & GUID, and then fill in the MODULE_TYPE

Lab 2: Sample Application INF file

SampleApp.inf
~/fw/edk2-ws/edk2/MyPkg/SampleApp/

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

[Sources]
  XXXXXXXXX
[Packages]
  #XXXXXXXX

[LibraryClasses]
  #XXXXXXXXXXXXX

[Guids]
 # . . .
```

SampleApp

Get a GUID

UEFI_APPLICATION

SampleApp.c

Get a GUID guidgenerator.com/ or https://www.guidgen.com/

Copy and paste LabGuide.md

Lab 2: Sample Application ‘C’ file

SampleApp.c
~/fw/edk2-ws/edk2/MyPkg/SampleApp/

```
/** @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/OpenBoardPkg.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

Open *OpenBoardPkg.dsc
~/fw/edk2-ws/edk2-platforms/Platform/Intel/SimicsOpenBoar

```
322 gEfiMdeModulePkgTokenSpaceGuid.PcdHelloWorldPrintTimes|3
323 #   Here is where you would put the HelloWorldPrintString PCD
324 # HINT: look at MdeModulePkg.dec for HelloWorldPrintString
325
326
327 [Components.X64]
328 # UEFI / EDK II Training
329 MdeModulePkg/Application/HelloWorld/HelloWorld.inf
330 # Add new modules here
331 MyPkg/SampleApp/SampleApp.inf
```

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

Open SampleApp.inf
~/fw/edk2-ws/edk2/MyPkg/SampleApp

```
29 #  VALID_ARCHITECTURES           = IA32 X64 IPF EBC
30 #
31
32 [Sources]
33   SampleApp.c
34
35 [Packages]
36   MdePkg/MdePkg.dec
37
38 [LibraryClasses]
39   UefiApplicationEntryPoint
```

MyPkg/SampleApp/SampleApp.inf

Open SampleApp.c
~/fw/edk2-ws/edk2/MyPkg/SampleApp

EmulatorPkg.dsc

```
12
13 **/
14
15 #include <Uefi.h>
16 #include <Library/UefiApplicationEntryPoint.h>
17
```

MyPkg/SampleApp/SampleApp.c

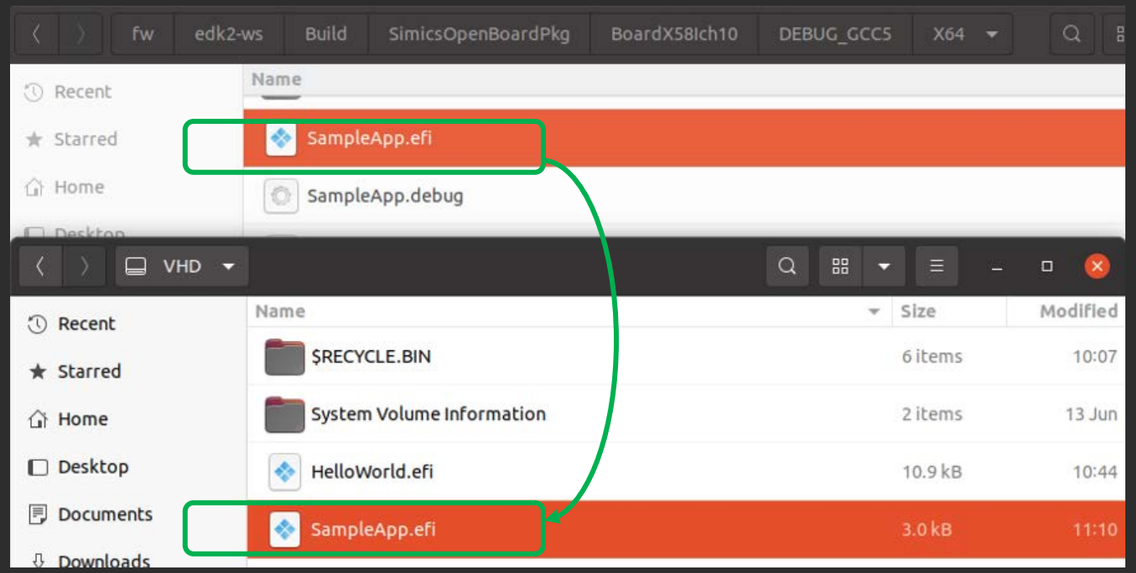
Lab 2 : Will it compile now?

At the Terminal Command Prompt, Build BoardX58Ich10

```
$> cd ~/fw/edk2-ws/edk2-platforms/Platform/Intel/
$> python build_bios.py -p BoardX58Ich10  -t GCC5
```

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

```
$ cp ~/fw/edk2-ws/edk2/Build/SimicsOpenBoardPkg/BoardX58Ich10/DEBUG_GCC5/X64/SampleApp.efi ~/VHD
```



Build Directory

VHD Disk

Invoke Simics & Run SampleApp

1. Run the Simics qsp-modern-core script from Terminal 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

Error on SampleApp.inf

```
EFI_SOURCE       = /home/u-uefi/src/edk2/EdkCompatibilityPkg
EDK_TOOLS_PATH   = /home/u-uefi/src/edk2/BaseTools
CONF_PATH        = /home/u-uefi/src/edk2/Conf


Architecture(s)  = X64
Build target     = DEBUG
Toolchain        = GCC5

Active Platform          = /home/u-uefi/src/edk2/OvmfPkg/OvmfPkgX64.dsc
Flash Image Definition   = /home/u-uefi/src/edk2/OvmfPkg/OvmfPkgX64.fdf

Processing meta-data ..

build.py...
/home/u-uefi/src/edk2/SampleApp/SampleApp.inf(21): error 3000: No value specified
        FILE_GUID                      =


- Failed -
Build end time: 15:20:18, Jun.15 2017
Build total time: 00:00:03

u-uefi@uuefi-TPad:~/src/edk2$
```

```
Processing meta-data .......


build.py...
 : error C0DE: Unknown fatal error when processing [/home/u-uefi/src/edk2/SampleApp/SampleApp.inf]

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

(Python 2.7.12 on linux2) Traceback (most recent call last):
  File "/home/u-uefi/src/edk2/BaseTools/BinWrappers/PosixLike/../../Source/Python/build/build.py", lin
e 2493, in Main
    MyBuild.Launch()
  File "/home/u-uefi/src/edk2/BaseTools/BinWrappers/PosixLike/../../Source/Python/build/build.py", lin
e 2226, in Launch
    self._MultiThreadBuildPlatform()
  File "/home/u-uefi/src/edk2/BaseTools/BinWrappers/PosixLike/../../Source/Python/build/build.py", lin
e 2047, in _MultiThreadBuildPlatform
    Ma.CreateCodeFile(True)
  File "/home/u-uefi/src/edk2/BaseTools/Source/Python/AutoGen/AutoGen.py", line 4213, in CreateCodeFil
e
```

The FILE_GUID was invalid or not updated from “XXX…” to a proper formatted GUID

Possible Build Errors

Error on SampleApp.inf

```
Building ... /home/u-uefi/src/edk2/ShellPkg/Application/Shell/Shell.inf [X64]
Building ... /home/u-uefi/src/edk2/MdeModulePkg/Application/HelloWorld/HelloWorld.inf [X64]
make: Nothing to be done for 'tbuild'.
Building ... /home/u-uefi/src/edk2/SampleApp/SampleApp.inf [X64]
make: Nothing to be done for 'tbuild'.
"gcc" -g -fshort-wchar -fno-builtin -fno-strict-aliasing -Wall -Wno-array-bounds -ffunction-sections -fdata-sect
ions -include AutoGen.h -fno-common -DSTRING_ARRAY_NAME=SampleAppStrings -m64 -fno-stack-protector "-DEFIAPI=__a
ttribute__((ms_abi))" -maccumulate-outgoing-args -mno-red-zone -Wno-address -mcmodel=small -fpie -fno-asynchrono
us-unwind-tables -Wno-address -flto -DUSING_LTO -Os -mno-mmx -mno-sse -D DISABLE_NEW_DEPRECATED_INTERFACES -c -o
 /home/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GCC5/X64/SampleApp/SampleApp/OUTPUT/./SampleApp.obj -I/home/u-uefi/sr
c/edk2/SampleApp -I/home/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GCC5/X64/SampleApp/SampleApp/DEBUG /home/u-uefi/src
/edk2/SampleApp/SampleApp.c
make: Nothing to be done for 'tbuild'.
In file included from <command-line>:0:0:
/home/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GCC5/X64/SampleApp/SampleApp/DEBUG/AutoGen.h:16:18: fatal error: Base.
h: No such file or directory
compilation terminated.
GNUmakefile:329: recipe for target '/home/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GCC5/X64/SampleApp/SampleApp/OUTPU
T/SampleApp.obj' failed
make: *** [/home/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GCC5/X64/SampleApp/SampleApp/OUTPUT/SampleApp.obj] Error 1


build.py...
 : error 7000: Failed to execute command
        make tbuild [/home/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GCC5/X64/SampleApp/SampleApp]
```

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

Possible Build Errors

GCC compiler Error on SampleApp.c

```
make: Nothing to be done for 'tbuild'.
"gcc" -g -fshort-wchar -fno-builtin -fno-strict-aliasing -Wall -Wno-array-bounds -ffunction-sections -fdata-sect
ions -include AutoGen.h -fno-common -DSTRING_ARRAY_NAME=SampleAppStrings -m64 -fno-stack-protector "-DEFIAPI=__a
ttribute__((ms_abi))" -maccumulate-outgoing-args -mno-red-zone -Wno-address -mcmodel=small -fpie -fno-asynchrono
us-unwind-tables -Wno-address -flto -DUSING_LTO -Os -mno-mmx -mno-sse -D DISABLE_NEW_DEPRECATED_INTERFACES -c -o
 /home/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GCC5/X64/SampleApp/SampleApp/OUTPUT/./SampleApp.obj -I/home/u-uefi/sr
c/edk2/SampleApp -I/home/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GCC5/X64/SampleApp/SampleApp/DEBUG -I/home/u-uefi/s
rc/edk2/MdePkg -I/home/u-uefi/src/edk2/MdePkg/Include -I/home/u-uefi/src/edk2/MdePkg/Include/X64 /home/u-uefi/sr
c/edk2/SampleApp/SampleApp.c
/home/u-uefi/src/edk2/SampleApp/SampleApp.c:16:48: fatal error: Library/UefiApplicationsEntryPoint.h: No such fi
le or directory
 #include <Library/UefiApplicationsEntryPoint.h>
                                                ^
compilation terminated.
GNUmakefile:357: recipe for target '/home/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GCC5/X64/SampleApp/SampleApp/OUTPU
T/SampleApp.obj' failed
make: *** [/home/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GCC5/X64/SampleApp/SampleApp/OUTPUT/SampleApp.obj] Error 1


build.py...
 : error 7000: Failed to execute command
        make tbuild [/home/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GCC5/X64/SampleApp/SampleApp]


build.py...
 : error F002: Failed to build module
        /home/u-uefi/src/edk2/SampleApp/SampleApp.inf [X64, GCC5, DEBUG]
```

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

Possible Build Errors

GCC compiler Error on SampleApp.c

```
objcopy --add-gnu-debuglink=/home/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GCC5/X64/SampleApp/SampleApp/DEB
UG/SampleApp.debug /home/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GCC5/X64/SampleApp/SampleApp/DEBUG/Sample
App.dll
objcopy: /home/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GCC5/X64/SampleApp/SampleApp/DEBUG/stSSWk1b: debugl
ink section already exists
cp -f /home/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GCC5/X64/SampleApp/SampleApp/DEBUG/SampleApp.debug /ho
me/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GCC5/X64/SampleApp.debug
"GenFw" -e UEFI_APPLICATION -o /home/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GCC5/X64/SampleApp/SampleApp/
DEBUG/SampleApp.efi /home/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GCC5/X64/SampleApp/SampleApp/DEBUG/Sampl
eApp.dll
GenFw: Elf64Convert.c:440: ScanSections64: Assertion `FALSE' failed.
GenFw: ERROR 3000: Invalid
  Did not find any '.text' section.
Aborted (core dumped)
GNUmakefile:325: recipe for target '/home/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GCC5/X64/SampleApp/Sampl
eApp/DEBUG/SampleApp.efi' failed
make: *** [/home/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GCC5/X64/SampleApp/SampleApp/DEBUG/SampleApp.efi]
 Error 134


build.py...
 : error 7000: Failed to execute command
        make tbuild [/home/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GCC5/X64/SampleApp/SampleApp]


build.py...
 : error F002: Failed to build module
        /home/u-uefi/src/edk2/SampleApp/SampleApp.inf [X64, GCC5, DEBUG]
```

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

Possible Build Errors

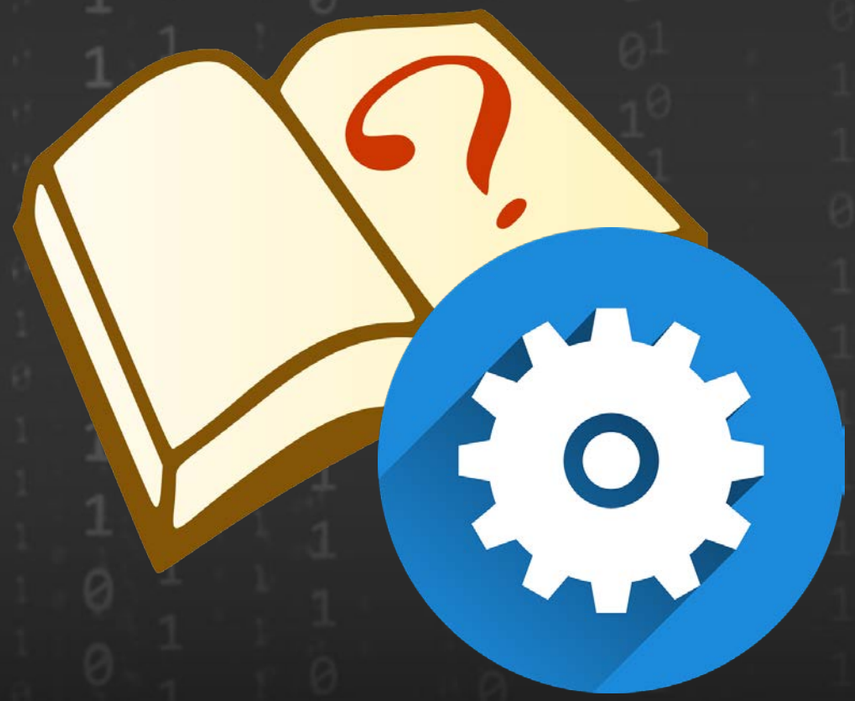
Error at the Shell prompt

```
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 add 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)

Open | OpenBoardPkg.dsc ~/fw/edk2-ws/edk2-platforms/Platform/Intel/SimicsOpenBoardPkg/BoardX58Ich10 | Save

OpenBoardPkg.dsc | OpenBoardPkgBuildOption.dsc

```
262
263   ShellPkg/Application/Shell/Shell.inf {
264     <PcdsFixedAtBuild>
265       gEfiShellPkgTokenSpaceGuid.PcdShellLibAutoInitialize|FALSE
266     <LibraryClasses>
267       NULL|ShellPkg/Library/UefiShellLevel2CommandsLib/UefiShellLevel2CommandsLib.inf
268       NULL|ShellPkg/Library/UefiShellLevel1CommandsLib/UefiShellLevel1CommandsLib.inf
269 !if $(ADD_SHELL_STRING) == TRUE
270         # Training Lib for build switch lab
271       NULL|ShellPkg/Library/UefiShellLevel3CommandsLib_Training_Lib/
    UefiShellLevel3Commands_Training_Lib.inf
272 !else
273         # normal Lib for build switch
274       NULL|ShellPkg/Library/UefiShellLevel3CommandsLib/UefiShellLevel3CommandsLib.inf
275 !endif
276
277       NULL|ShellPkg/Library/UefiShellDriver1CommandsLib/UefiShellDriver1CommandsLib.inf
278       NULL|ShellPkg/Library/UefiShellInstall1CommandsLib/UefiShellInstall1CommandsLib.inf
279       NULL|ShellPkg/Library/UefiShellDebug1CommandsLib/UefiShellDebug1CommandsLib.inf
280       NULL|ShellPkg/Library/UefiShellNetwork1CommandsLib/UefiShellNetwork1CommandsLib.inf
```

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-modern-core script from Terminal 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 Terminal Command Prompt

```
$> Cd ~/fw/edk2-ws/edk2-platforms/Platform/Intel/
$> python build_bios.py -p BoardX58Ich10  -t GCC5
```

```
==========================================
Calling build -n 0 -D ADD_SHELL_STRING --log=Build.log --report-file=BuildReport.log
Build environment: Windows-10-10.0.19041-SP0
```

Lab 2.1: Compiling w/ Build Switch

1

Invoke Simics and Test Shell `Ver` command

Copy ~/fw/edk2-ws/Build/SimicsOpenBoardPkg/BoardX58Ich10/DEBUG_GCC5/FV/BOARDX58ICH10.fd To *<SimicsInstallDir>*/simics-qsp-x86-6.0.57/targets/qsp-x86/images

Run the Simics QSP script from Terminal 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`" -Save
2. Edit the file `~/FW/edk2-ws/edk2-platforms/Platform/Intel/SimicsOpenBoardPkg/BoardX58Ich10/OpenBoardPkgBuildOption.dsc`
   - add the following DEFINE after Line 7 and then 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 Terminal Command Prompt

```
$> cd ~/fw/edk2-ws/edk2-platforms/Platform/Intel/
$> python build_bios.py -p BoardX58Ich10  -t GCC5
```

Lab 2.1: Compiling w/ Build Switch

2

Invoke Simics and Test Shell Ver command

Copy ~/fw/edk2-ws/Build/SimicsOpenBoardPkg/BoardX58Ich10/DEBUG_GCC5/FV/BOARDX58ICH10.fd To *<SimicsInstallDir>*/simics-qsp-x86-6.0.57/targets/qsp-x86/images

Run the Simics QSP script from Terminal 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 ~/Lab_Material_FW/LabSampleCode/SampleApp/MyPkg directory to ~/FW/edk2-ws/edk2
2. See class files for the solution. . .FW/LabSampleCode/LabSolutions/LessonB.2
3. Copy the .inf and .c files to ~/FW/edk2-ws/edk2/MyPkg/SampleApp
4. Search sample DSC for reference to SampleApp.inf and add this line to your workspace DSC file ~/FW/edk2-ws/edk2-platforms/Platform/Intel/SimicsBoardPkg /BoardX58Ich10/OpenBoardPkg.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

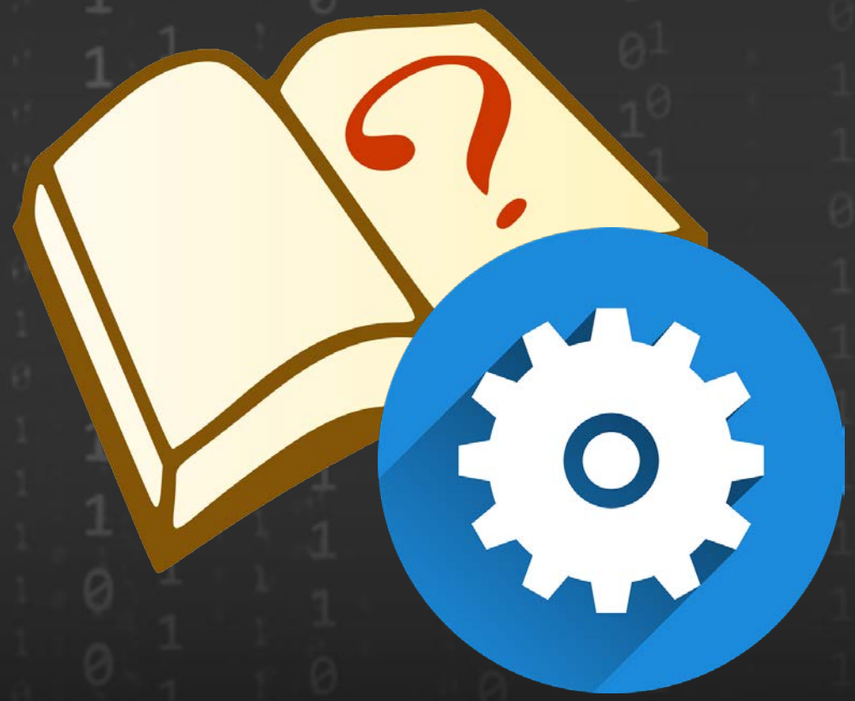
Lab 4: Wait for an Event

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

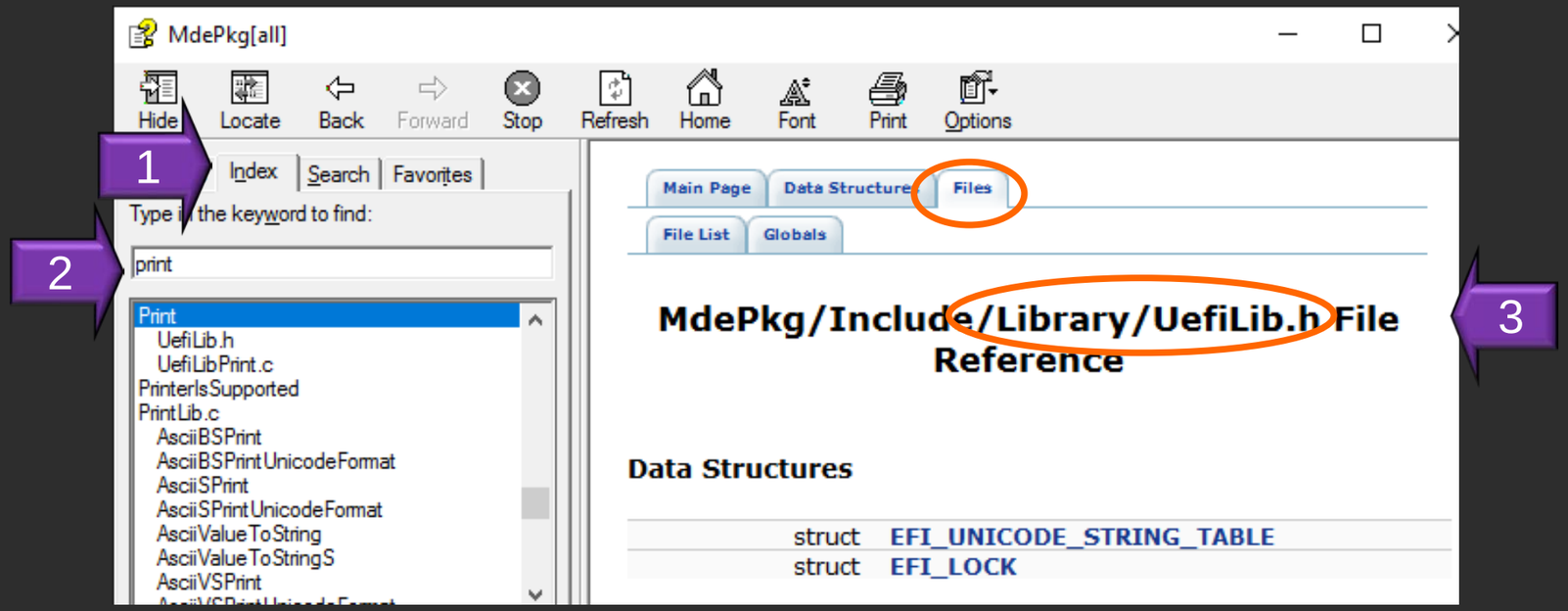
Add code to print to the console the hex address of the system table pointer

- 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 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

Open SampleApp.c ~/fw/edk2-ws/edk2/MyPkg/SampleApp/ Save

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;
}
```

Open SampleApp.inf ~/fw/edk2-ws/edk2/MyPkg/SampleApp/ Save

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 Terminal Command Prompt, Re-Build BoardX58Ich10

```
$> cd ~/fw/edk2-ws/edk2-platforms/Platform/Intel/
$> python build_bios.py -p BoardX58Ich10  -t GCC5
```

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

```
$ cp ../Build/SimicsOpenBoardPkg/BoardX58Ich10/DEBUG_GCC5/X64/SampleApp.efi ~/VHD
```

3. Run the Simics qsp-modern-core script from Terminal 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:/>
```

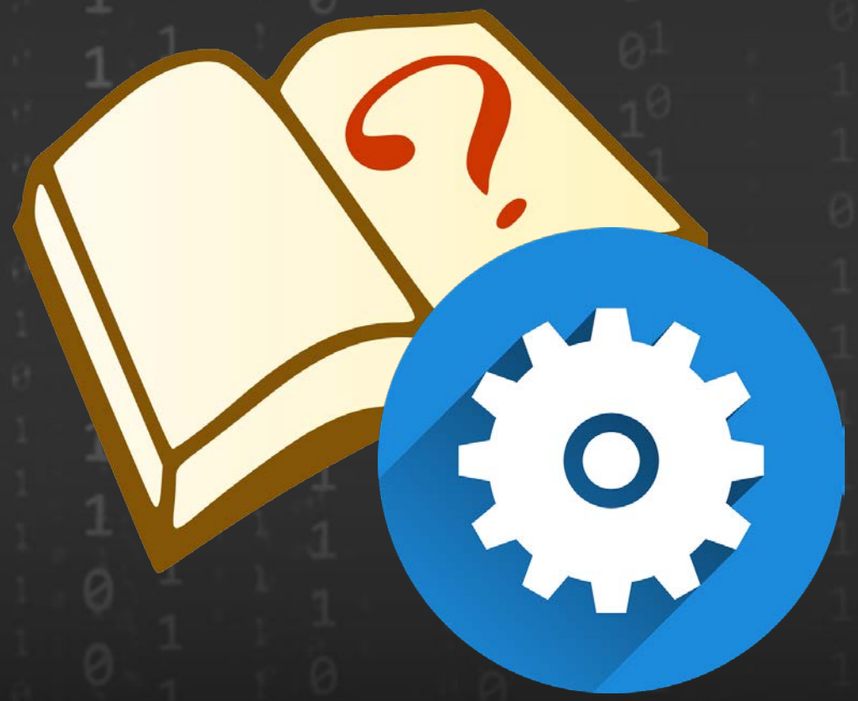
5. Exit Simics

```
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: 0xDEFED018

Press any Key to continue :

_
```

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

Lab 4: HOW?

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 ~ ln 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(~/src/edk2-ws/edk2/MdePkg/Library/UefiLib)-gedit

Save

```
. . .
  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(~/fw/edk2-ws/edk2/MyPkg/SampleApp)-gedit

Save

```
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.

```
/SampleApp.c
/home/u-uefi/src/edk2/SampleApp/SampleApp.c: In function 'UefiMain':
/home/u-uefi/src/edk2/SampleApp/SampleApp.c:42:3: error: 'gBS' undeclared (first
 use in this function)
  gBS->WaitForEvent (1, &gST->ConIn->WaitForKey, &EventIndex);
  ^
/home/u-uefi/src/edk2/SampleApp/SampleApp.c:42:3: note: each undeclared identifi
er is reported only once for each function it appears in
/home/u-uefi/src/edk2/SampleApp/SampleApp.c:42:26: error: 'gST' undeclared (firs
t use in this function)
   gBS->WaitForEvent (1, &gST->ConIn->WaitForKey, &EventIndex);
                         ^
GNUmakefile:376: recipe for target '/home/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GC
C5/X64/SampleApp/SampleApp/OUTPUT/SampleApp.obj' failed
make: *** [/home/u-uefi/src/edk2/Build/OvmfX64/DEBUG_GCC5/X64/SampleApp/SampleAp
p/OUTPUT/SampleApp.obj] Error 1
```

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

Open SampleApp.c ~/src/edk2-ws/edk2/MyPlg/SampleApp Save

```
#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;
}
```

Copy and paste Lab Guide

Lab 4 : Build and Test SampleApp

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

```
$> Cd ~/fw/edk2-ws/edk2-platforms/Platform/Intel/
$> python build_bios.py -p BoardX58Ich10  -t GCC5
```

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

```
$ cp ../Build/SimicsOpenBoardPkg/BoardX58Ich10/DEBUG_GCC5/X64/SampleApp.efi ~/VHD
```

3. Run the Simics qsp-modern-core script from Terminal Command Prompt:

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

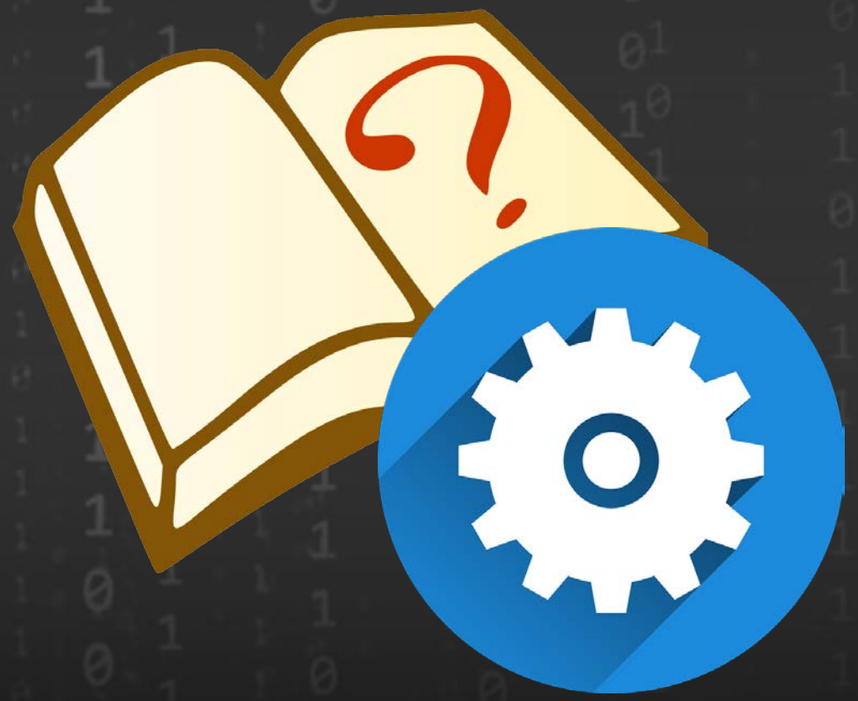
4. At the UEFI Shell prompt

```
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

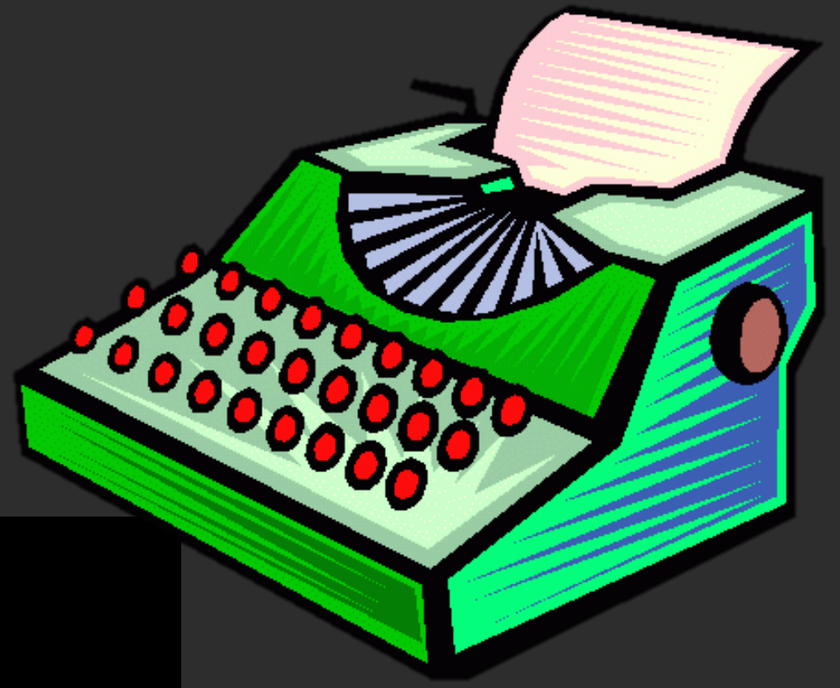


Lab 5 : Typewriter Function

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>



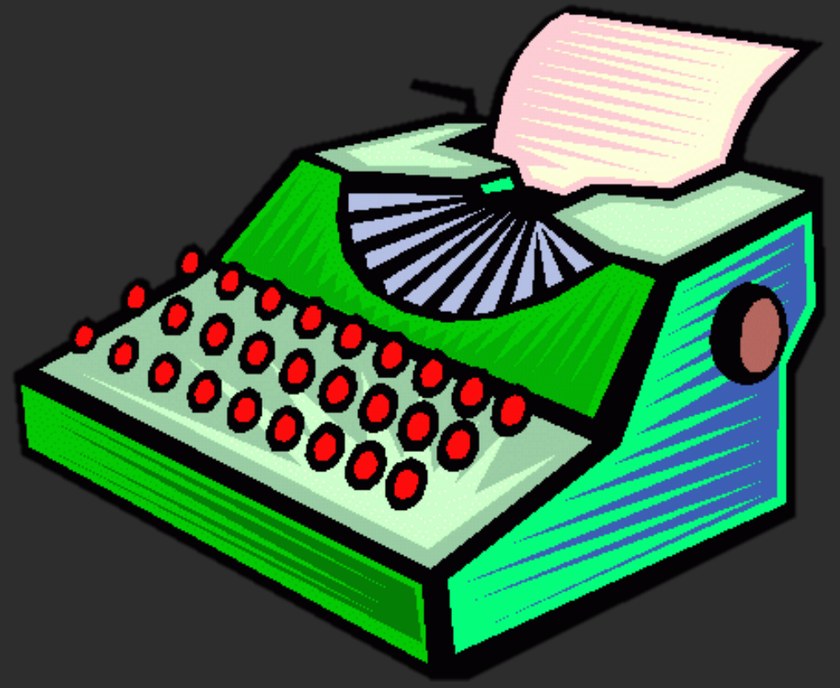
```
Shell> fs1:
FS1:\> SampleApp
System Table: 0xDEFED018

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
```

Copy and paste Lab Guide

Lab 5: Solution

(hint: Lesson B.5 has the solution)

SampleApp.c
~/fw/edk2-ws/edk2/MyPkg/SampleApp/

```
#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;
}
```

Copy and paste Lab Guide

Lab 5 : Build and Test SampleApp

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

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

```
$  cd ~/fw/edk2-ws/edk2-platforms/Platform/Intel/
$  python build_bios.py -p BoardX58Ich10  -t GCC5
```

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

```
$ cp ~/fw/edk2-ws/Build/SimicsOpenBoardPkg/BoardX58Ich10/DEBUG_GCC5/X64/SampleApp.efi ~/VHD
```

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

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

4. Run SampleApp

```
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:\> _
```

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 Terminal Command Prompt, Re-Build BoardX58Ich10

```
$> Cd ~/fw/edk2-ws/edk2-platforms/Platform/Intel/
$> python build_bios.py -p BoardX58Ich10  -t GCC5
```

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

```
$ cp ../Build/SimicsOpenBoardPkg/BoardX58Ich10/DEBUG_GCC5/X64/SampleApp.efi ~/VHD
```

4. Run the Simics QSP script from Terminal 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: 0xDEFED018

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 Terminal Command Prompt, Re-Build BoardX58Ich10

```
$> cd ~/fw/edk2-ws/edk2-platforms/Platform/Intel/
$> python build_bios.py -p BoardX58Ich10  -t GCC5
```

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

```
$ cp ../Build/SimicsOpenBoardPkg/BoardX58Ich10/DEBUG_GCC5/X64/SampleApp.efi ~/VHD
```

4. Run the Simics QSP script from Terminal 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: 0xDEFED018

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
```

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

```
$> Cd ~/fw/edk2-ws/edk2-platforms/Platform/Intel/
$> python build_bios.py -p BoardX58Ich10  -t GCC5
```

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

```
$ cp ../Build/SimicsOpenBoardPkg/BoardX58Ich10/DEBUG_GCC5/X64/SampleApp.efi  ~/VHD
```

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

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

6. Run SampleApp

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

FALSE

```
FS1:\> sampleapp
System Table: 0xDEFED018

Press any Key to continue :
Typewriter feature not enabled

FS1:\> _
```

TRUE

```
FS1:\> sampleapp
System Table: 0xDEFED018

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:\> _
```

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 form the shell in Simics

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

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

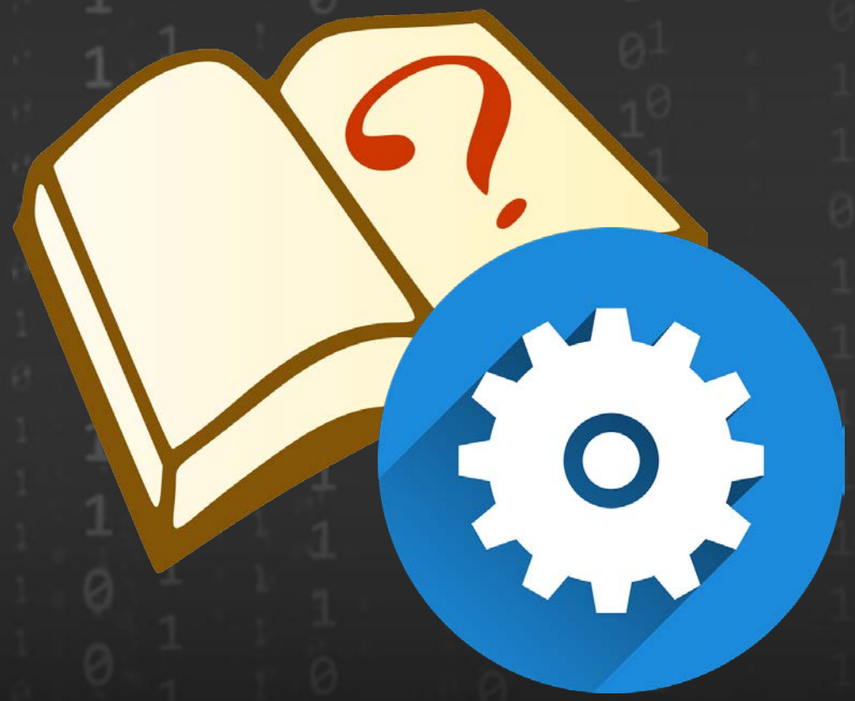
USING EADK

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

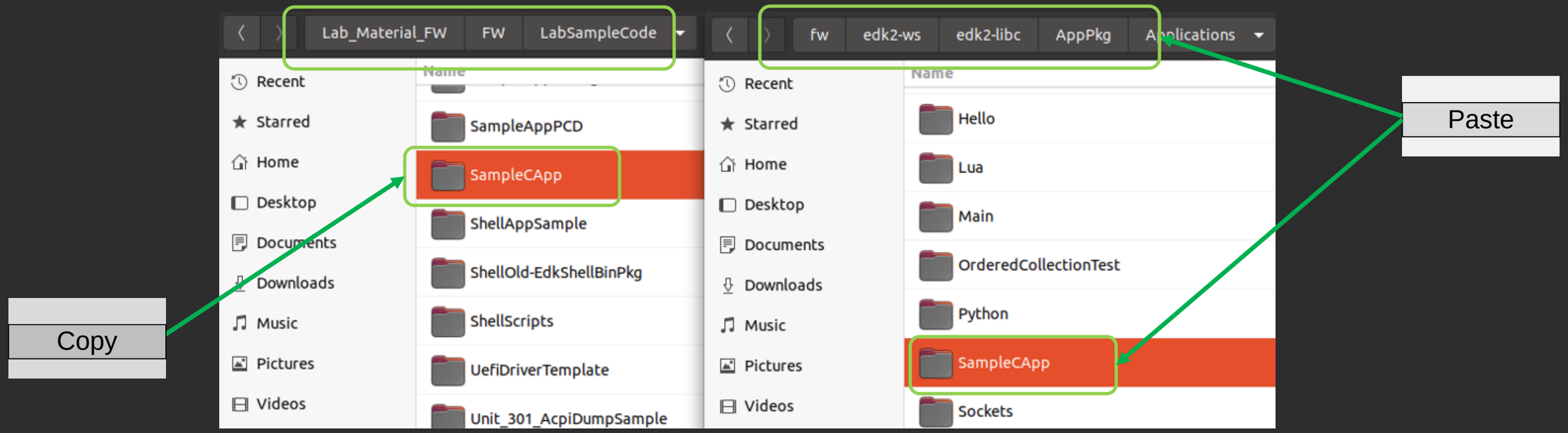
What differences are there using the LibC

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
  `~/fw/edk2-ws/edk2-libc/AppPkg/Applications/SampleCApp`



Lab 6: EDK II using EADK

Check out AppPkg/Applications/SampleCApp

SampleCApp.c and SampleCApp.inf

Open SampleCApp.c ~/fw/edk2-ws/edk2-libc/AppPkg Save

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

Open SampleCApp.inf ~/fw/edk2-ws/edk2-libc/AppPkg Save

```
[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 ~/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
```

Lab 6 :Build and Test SampleCApp

1. Build the AppPkg at a NEW Terminal Prompt

```
$ cd ~/fw/edk2-ws/
$ . setenv.sh
$ cd edk2
$ . edksetup.sh
$> build -p AppPkg/AppPkg.dsc –m AppPkg/Applications/SampleCApp/SampleCApp.inf
```

2. Copy the built application SampleCApp.efi to the VHD Drive

```
$cp ~/fw/edk2-ws/Build/AppPkg/DEBUG_GCC5/X64/SampleCApp.efi ~/VHD
```

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

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

4. Run SampleCApp

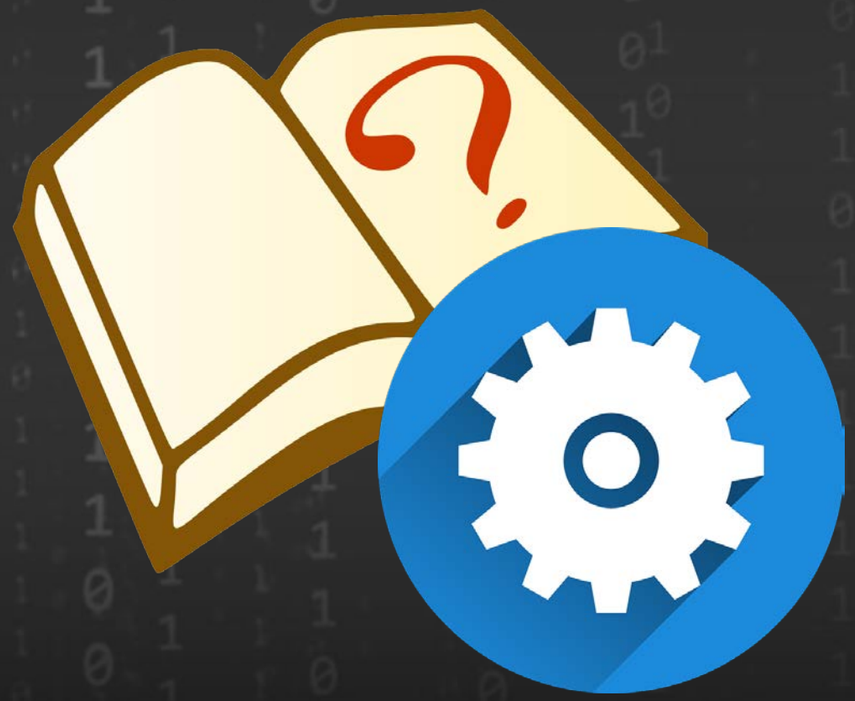
```
Shell> fs1:
FS1:\> SampleCApp.efi
FS1:\> _
```

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
```

Copy and paste LabGuide

Lab 7: Add the same functionally from Lab 5

SampleCApp.c and SampleCApp.inf

SampleCApp.c
~/fw/edk2-ws/edk2/-libc/ SampleCApp/

```c
#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;
 }
```

SampleACpp.inf
~/fw/edk2-ws/edk2-libc/ /SampleCApp/

```
[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
~/fw/edk2-ws/edk2/-libc/ SampleCApp/

```c
#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

SampleACpp.inf
~/fw/edk2-ws/edk2-libc/ /SampleCApp/

```inf
[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
~/fw/edk2-ws/edk2/-libc/ SampleCApp/

```c
#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

SampleACpp.inf
~/fw/edk2-ws/edk2-libc/ /SampleCApp/

```inf
[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
~/fw/edk2-ws/edk2/-libc/ SampleCApp/

```c
#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

5

SampleACpp.inf
~/fw/edk2-ws/edk2-libc/ /SampleCApp/

```
[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 :Build and Test SampleCApp

1. Build the AppPkg at the Terminal 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)

```
$> cp ~/fw/edk2-ws/Build/AppPkg/DEBUG_GCC5/X64/SampleCApp.efi  ~/VHD
```

3. Run the Simics QSP script from Terminal 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

tianocore

Questions?

Return to Main Training Page

Return to Training Table of contents for next presentation link

tianocore

tianocore

www.tianocore.org

ACKNOWLEDGEMENTS

Mounting a VHD File Disk

To Mount:

Open Terminal Prompt in the Simics Install directory directory

e.g., `<InstallDir>/simics-qsp-x86-6.0.57/targets/qsp-x86/images`

(this is the directory where the .VHD files are copied to)

First use virt-list-filesystems to create a file system from the .VHD file.

This will show a partition e.g., /dev/sda1 required for the guestmount command

```
$ sudo virt-list-filesystems UefiAppLab.vhd
```

Use the guestmount command to mount the file.

(Note: may need to create ~/VHD directory with correct permissions)

```
$ sudo guestmount -a UefiAppLab.vhd -m /dev/sda1 -w ~/VHD -o allow_other
```

To Unmount:

```
$ sudo umount /VHD
```

https://www.youtube.com/watch?v=A7OlFwTNWYc

Simics Agent for UEFI using Matic

- Note, the VHD method of copying UEFI Applications to use inside the UEFI Shall 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`