tianocore

UEFI & EDK II TRAINING

UEFI SHELL LAB w/ WINDOWS EMULATION

See also Lab_Guide.md for Copy & Paste examples in labs

tianocore.org

Lesson Objective

- Run UEFI Shell (Windows Emulation)
- Run UEFI Shell Commands
- Run UEFI Shell Scripts

tianocore

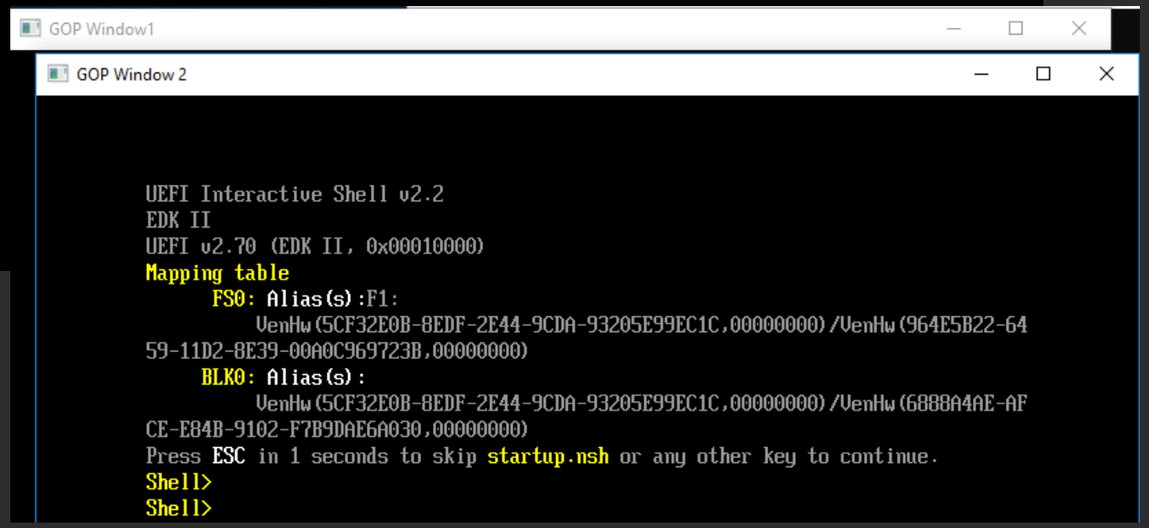
UEFI SHELL LAB WITH WIN EMULATOR

Invoke Win Emulation

First Setup for Building EDK II for EmulatorPkg, See Lab Setup

From the VS command prompt

```
CD C:\FW\edk2-ws
# set up PACKAGES_PATH
$> set WORKSPACE=%CD%
$> set PACKAGES_PATH=%WORKSPACE%\edk2;%WORKSPACE%\edk2-libc
$> cd edk2
$> edksetup Rebuild
$> Build -a X64
$> RunEmulator.bat
```



tianocore

UEFI SHELL COMMANDS

Commands from the Command Line Interface

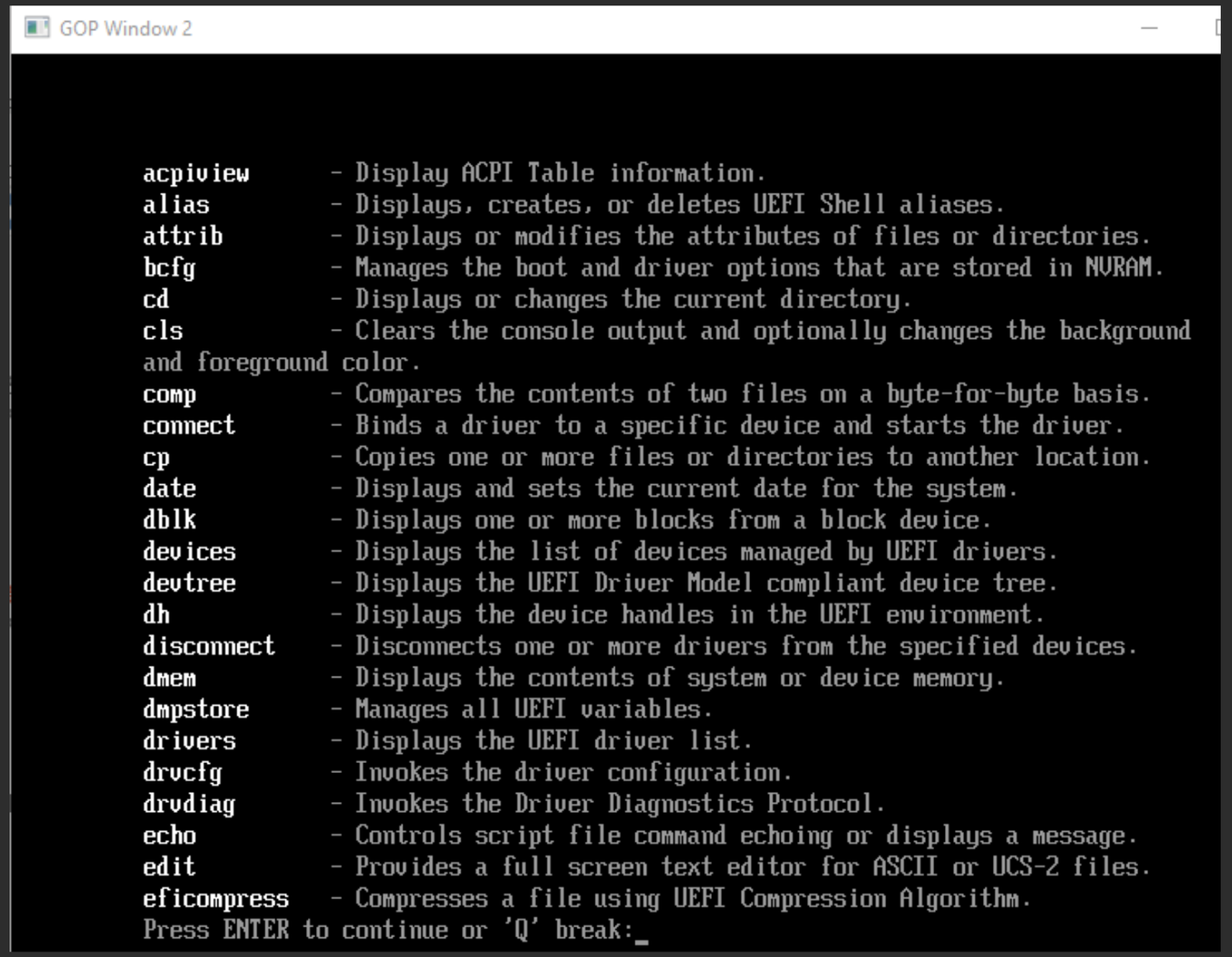
Common Shell Commands for Debugging

```
help
mm
mem
memmap
drivers
devices
devtree
dh
Load
Dmpstore
pci
stall
```

"**-b**" is the command line parameter for breaking after each page.

Shell Help

```
Shell> help -b
```



Shell “memmap”

```
Shell> memmap
```

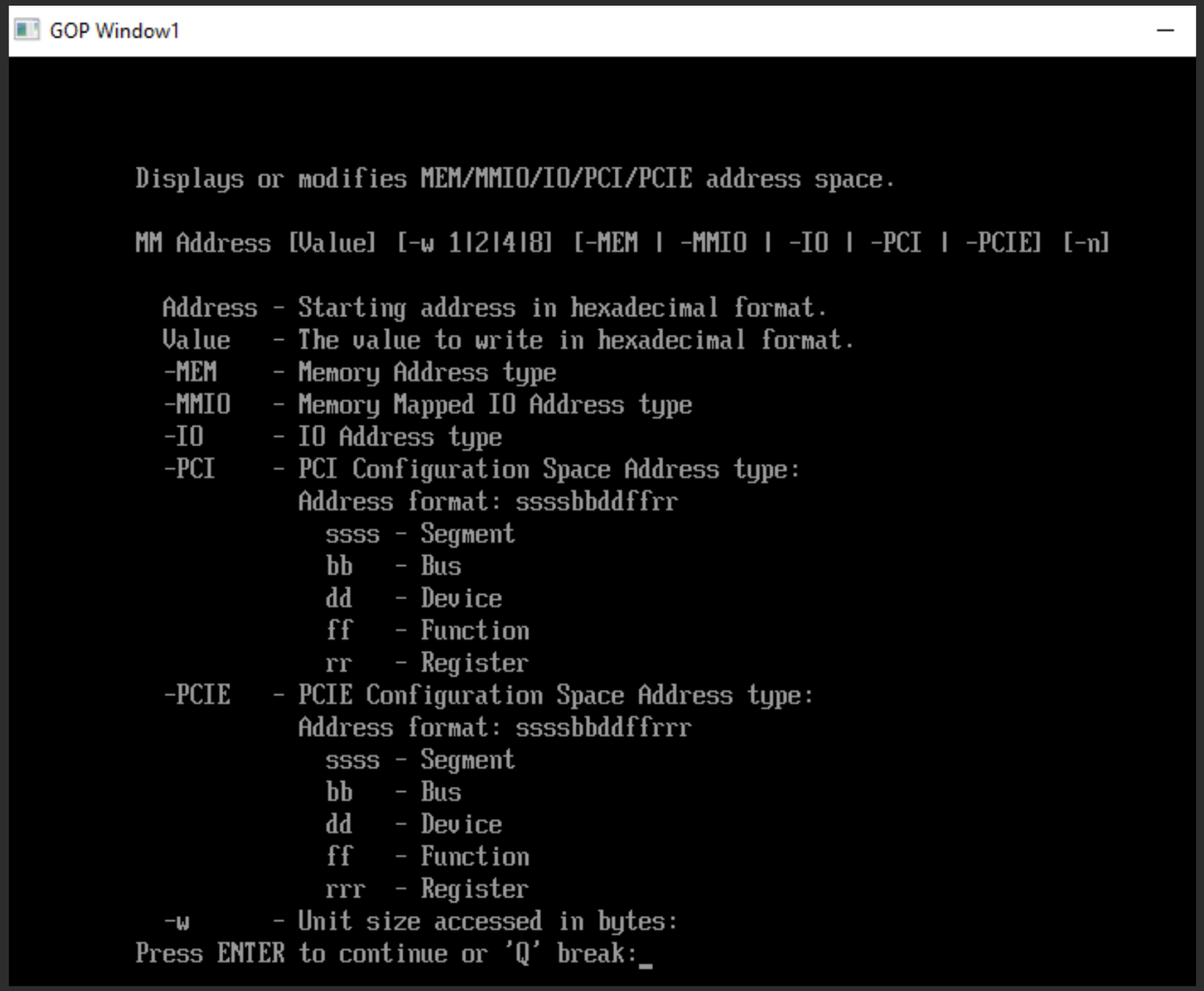
Displays the memory map maintained by the UEFI environment

```
RT_Data    000002B208574000-000002B208574FFF 0000000000000001 800000000000000F
BS_Data    000002B208575000-000002B20859CFFF 0000000000000028 000000000000000F
RT_Data    000002B20859D000-000002B20859FFFF 0000000000000003 800000000000000F
MMIO       000002B200580000-000002B20058BFFF 000000000000000C 8000000000000001

  Reserved  :             0 Pages (0 Bytes)
  LoaderCode:           307 Pages (1,257,472 Bytes)
  LoaderData:             0 Pages (0 Bytes)
  BS_Code   :         1,239 Pages (5,074,944 Bytes)
  BS_Data   :         5,936 Pages (24,313,856 Bytes)
  RT_Code   :            97 Pages (397,312 Bytes)
  RT_Data   :           193 Pages (790,528 Bytes)
  ACPI_Recl :             0 Pages (0 Bytes)
  ACPI_NVS  :             0 Pages (0 Bytes)
  MMIO      :            12 Pages (49,152 Bytes)
  MMIO_Port :             0 Pages (0 Bytes)
  PalCode   :             0 Pages (0 Bytes)
  Available :        24,996 Pages (102,383,616 Bytes)
  Persistent:             0 Pages (0 Bytes)
             --------------

Total Memory:           128 MB (134,217,728 Bytes)
Shell> _
```

Shell “mm”

```
Shell> mm -? -b
```

Help for “mm” command shows options for different types of memory and I/O that can be modified

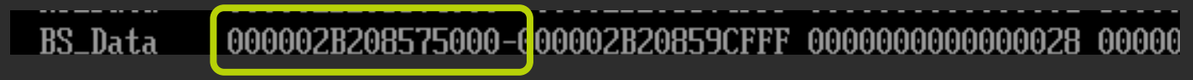


Shell “mm”

```
Shell> mm **
```

```
Shell> mm 2b208575000
MEM  0x000002B208575000 : 0x70 >

MEM  0x000002B208575001 : 0x68 >

MEM  0x000002B208575002 : 0x64 >

MEM  0x000002B208575003 : 0x30 >

MEM  0x000002B208575004 : 0x01 >

MEM  0x000002B208575005 : 0x00 >

MEM  0x000002B208575006 : 0x00 > q

Shell> _
```

**Pick a location from the MemMap command on Previous slide



MM in can display / modify any location

Do **not** try in Win Emulator

```
Shell> mm 0000
```

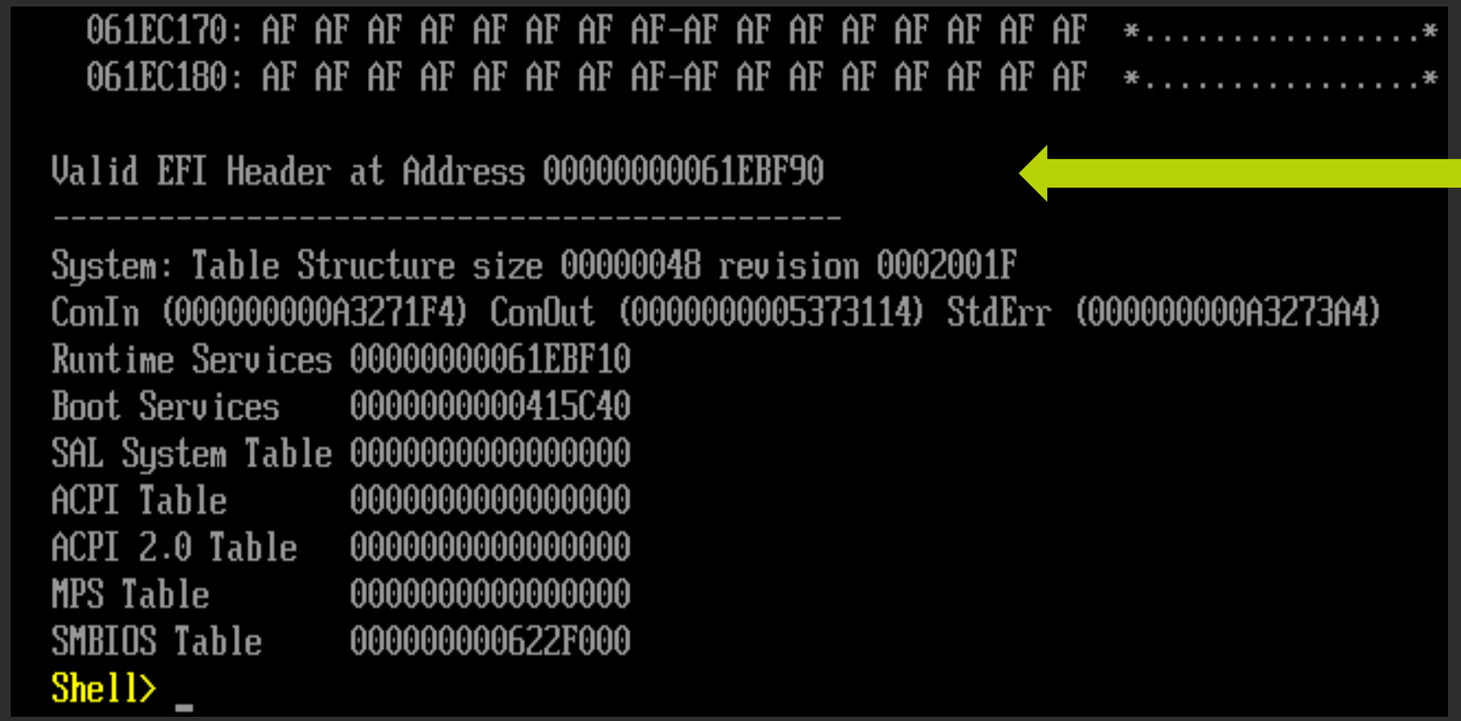
“q” to quit

Shell “mem”

```
Shell> mem
```

Displays the contents of the system or device memory

without arguments, displays the system memory configuration.



UEFI System Table Pointer

Shell “Drivers”

```
Shell> drivers -b
```

```
                T   D
D               Y C I
R               P F A
V  VERSION      E G G #D #C DRIVER NAME                          IMAGE NAME
== ======== = = = == == ================================== ==========
47 0000000A D - -  2  - Platform Console Management Driver  ConPlatformDxe
48 0000000A D - -  2  - Platform Console Management Driver  ConPlatformDxe
49 0000000A B - -  2  2 Console Splitter Driver             ConSplitterDxe
4A 0000000A B - -  2  2 Console Splitter Driver             ConSplitterDxe
4B 0000000A ? - -  -  - Console Splitter Driver             ConSplitterDxe
4C 0000000A B - -  2  2 Console Splitter Driver             ConSplitterDxe
4D 0000000A ? - -  -  - Console Splitter Driver             ConSplitterDxe
51 0000000A D - -  2  - Graphics Console Driver             GraphicsConsoleDxe
52 0000000A B - -  1  1 Serial Terminal Driver              TerminalDxe
53 0000000A D - -  1  - Generic Disk I/O Driver             DiskIoDxe
54 0000000B ? - -  -  - Partition Driver(MBR/GPT/El Torito) PartitionDxe
57 0000000A ? - -  -  - PCI Bus Driver                      PciBusDxe
59 0000000A ? - -  -  - SCSI Bus Driver                     ScsiBus
5A 0000000A ? - -  -  - Scsi Disk Driver                    ScsiDisk
5B 0000000A B - -  1  4 Emu Bus Driver                      EmuBusDriver
5C 0000000A D - -  2  - Emulator GOP Driver                 EmuGopDxe
5D 0000000A D - -  1  - Emu Simple File System Driver       EmuSimpleFileSystem
5E 0000000A D - X  1  - Emu Block I/O Driver                EmuBlockIo
Press ENTER to continue or 'Q' break:_
```

Displays the UEFI driver list.

To get a description of teach section in the list, Use:

```
Shell>
```

Shell “Devices”

```
Shell> devices -b
```

Displays a list of devices that UEFI drivers manage.

```
Shell> devices
     T   D
     Y C I
     P F A
CTRL E G G #P #D #C  Device Name
==== = = = == == === ============================================================
  1C R - -  0  1   5 VenHw(5CF32E0B-8EDF-2E44-9CDA-93205E99EC1C,00000000)
  20 R - -  0  1   1 VenHw(D3987D4B-971A-435F-8CAF-4967EB627241)/Uart(115200,8,N
,1)
  4E D - -  2  0   0 Primary Console Input Device
  4F D - -  2  0   0 Primary Console Output Device
  6F B - -  1  7   2 GOP Window 1
  70 B - -  1  7   2 GOP Window 2
  71 D - -  1  1   0 .
  72 D - X  1  2   0 disk.dmg:FW
  73 D - -  1  1   0 VenHw(5CF32E0B-8EDF-2E44-9CDA-93205E99EC1C,00000000)/VenHw(
FD5FBE54-8C35-B345-8A0F-7AC8A5FD0521,00000000)
  74 D - -  1  0   0 VT-100 Serial Console
Shell> _
```

For the Windows Emulation there is not that many devices

Shell “Devtree”

```
Shell> devtree -b
```

Displays tree of devices currently managed by UEFI drivers.

```
 Ctrl[04] MemoryMapped(0xB,0x1A3F5300000,0x1A3F531FFFF)
 Ctrl[13] MemoryMapped(0xB,0x1A3F4D80000,0x1A3F52FFFFF)
 Ctrl[1C] VenHw(5CF32E0B-8EDF-2E44-9CDA-93205E99EC1C,00000000)
   Ctrl[6F] GOP Window 1
     Ctrl[4E] Primary Console Input Device
     Ctrl[4F] Primary Console Output Device
   Ctrl[70] GOP Window 2
     Ctrl[4E] Primary Console Input Device
     Ctrl[4F] Primary Console Output Device
   Ctrl[71] .
   Ctrl[72] disk.dmg:FW
   Ctrl[73] VenHw(5CF32E0B-8EDF-2E44-9CDA-93205E99EC1C,00000000)/VenHw(FD5FBE54-
8C35-B345-8A0F-7AC8A5FD0521,00000000)
 Ctrl[20] VenHw(D3987D4B-971A-435F-8CAF-4967EB627241)/Uart(115200,8,N,1)
   Ctrl[74] VT-100 Serial Console
 Ctrl[2A] Fv(6D99E806-3D38-42C2-A095-5F4300BFD7DC)/FvFile(462CAA21-7614-4503-836
E-8AB6F4662331)/Enter Setup
 Ctrl[2B] Fv(6D99E806-3D38-42C2-A095-5F4300BFD7DC)/FvFile(EEC25BDC-67F2-4D95-B1D
5-F81B2039D11D)/BootManagerMenuApp
 Ctrl[2C] Fv(6D99E806-3D38-42C2-A095-5F4300BFD7DC)/FvFile(7C04A583-9E3E-4F1C-AD6
5-E05268D0B4D1)/Shell
 Ctrl[6D] VenHw(A04A27F4-DF00-4D42-B552-39511302113D)
 Ctrl[6E] VenHw(B3F56470-6141-4621-8F19-704E577AA9E8)
Press ENTER to continue or 'Q' break:
Shell> _
```

Shell Handle Database - “Dh”

```
Shell> dh -b
```

Dump Handle - Displays the device handles associated with UEFI drivers

```
Shell> dh -b
Handle dump
01: LoadedImage(DxeCore)
02: Decompress
03: FirmwareVolume2 DevicePath(..3D38-42C2-A095-5F4300BFD7DC)) FirmwareVolumeBlo
ck
04: DevicePath(..0x1A3F5300000,0x1A3F531FFFF)) FirmwareVolumeBlock
05: FC1BCDB0-7D31-49AA-936A-A4600D9DD083 EE4E5898-3914-4259-9D6E-DC7BD79403CF
06: ImageDevicePath(..87AB-47F9-A3FE-D50B76D89541)) LoadedImage(PcdDxe)
07: GetPcdInfo GetPcdInfoProtocol Pcd Pcd
08: ImageDevicePath(..A563-4561-B858-D8476F9DEFC4)) LoadedImage(Metronome)
09: MetronomeArch
0A: ImageDevicePath(..A7EB-4730-8C8E-CC466A9ECC3C)) LoadedImage(ReportStatusCode
RouterRuntimeDxe)
0B: SmartCardReader RscHandler
0C: ImageDevicePath(..8985-11DB-8429-0040D02B1835)) LoadedImage(RealTimeClock)
0D: RealTimeClockArch
0E: ImageDevicePath(..37AD-8743-BCF2-DF1A8FF12FAB)) LoadedImage(EmuReset)
0F: ResetArch
10: ImageDevicePath(..43B7-4784-95B1-F4226CB40CEE)) LoadedImage(RuntimeDxe)
11: RuntimeArch
12: ImageDevicePath(..96E8-2A4C-95F4-85248F989753)) LoadedImage(FwBlockService)

13: FirmwareVolume2 DevicePath(..0x1A3F4D80000,0x1A3F52FFFFF)) FirmwareVolumeBlP
ress ENTER to continue or 'Q' break:_
```

Also try `dh -d` with handle number to get more information on that handle.

Shell “Load”

```
Shell> load -?
```

Loads a UEFI driver into memory

```
Shell> load -? -b
Loads a UEFI driver into memory.

LOAD [-nc] file [file...]

  -nc  - Loads the driver, but does not connect the driver.
  File - Specifies a file that contains the image of the UEFI driver (wildcards
are
         permitted).

NOTES:
  1. This command loads a driver into memory. It can load multiple files at
     one time. The file name supports wildcards.
  2. If the -nc flag is not specified, this command attempts to connect the
     driver to a proper device. It might also cause previously loaded drivers
     to be connected to their corresponding devices.
  3. Use the 'UNLOAD' command to unload a driver.

EXAMPLES:
  * To load a driver:
```

tianocore

Shell “dmpstore”

```
Shell> dmpstore –all -b
```

Display the contents of the NVRAM variables

```
Shell> dmpstore -all -b
Variable NV+RT+BS 'EB704011-1402-11D3-8E77-00A0C969723B:MTC' DataSize = 0x04
  00000000: 03 00 00 00                                     *....*
Variable NV+RT+BS 'EFIGlobalVariable:BootOrder' DataSize = 0x0C
  00000000: 05 00 01 00 02 00 03 00-04 00 00 00             *............*
Variable NV+RT+BS 'EFIGlobalVariable:Boot0005' DataSize = 0x68
  00000000: 01 00 00 00 3C 00 55 00-45 00 46 00 49 00 20 00  *....<.U.E.F.I. .*
  00000010: 53 00 68 00 65 00 6C 00-6C 00 00 00 04 07 14 00  *S.h.e.l.l.......*
  00000020: 06 E8 99 6D 38 3D C2 42-A0 95 5F 43 00 BF D7 DC  *...m8=.B.._C....*
  00000030: 04 06 14 00 83 A5 04 7C-3E 9E 1C 4F AD 65 E0 52  *.......|>..O.e.R*
  00000040: 68 D0 B4 D1 04 04 10 00-53 00 68 00 65 00 6C 00  *h.......S.h.e.l.*
  00000050: 6C 00 00 00 7F FF 04 00-4E AC 08 81 11 9F 59 4D  *l.......N.....YM*
  00000060: 85 0E E2 1A 52 2C 59 B2-                         *....R,Y.*
Variable NV+RT+BS 'EFIGlobalVariable:Boot0004' DataSize = 0x9C
  00000000: 01 00 00 00 56 00 55 00-45 00 46 00 49 00 20 00  *....V.U.E.F.I. .*
  00000010: 42 00 6F 00 6F 00 74 00-4D 00 61 00 6E 00 61 00  *B.o.o.t.M.a.n.a.*
  00000020: 67 00 65 00 72 00 4D 00-65 00 6E 00 75 00 41 00  *g.e.r.M.e.n.u.A.*P
ress ENTER to continue or 'Q' break:_
```

Shell “pci”

```
Shell> pci -? -b
```

Display the help for the PCI command

```
Shell> pci -? -b
Displays PCI device list or PCI function configuration space and PCIe extended
configuration space.

PCI [Bus Dev [Func] [-s Seg] [-i [-ec ID]]]

  -s   - Specifies optional segment number (hexadecimal number).
  -i   - Displays interpreted information.
  -ec  - Displays detailed interpretation of specified PCIe extended capability
         ID (hexadecimal number).
  Bus  - Specifies a bus number (hexadecimal number).
  Dev  - Specifies a device number (hexadecimal number).
  Func - Specifies a function number (hexadecimal number).

NOTES:
  1. This command displays a list of all the PCI devices found in the system. It

     also displays the configuration space of a PCI device according to the
     specified bus (Bus), device (Dev), and function (Func) addresses. If the
     function address is not specified, it defaults to 0.
  2. The -i option displays verbose information for the specified PCI
     device. The PCI configuration space for the device is displayed with
     a detailed interpretation.
  3. If no parameters are specified, all PCI devices are listed.
Press ENTER to continue or 'Q' break:_
```

Shell “Stall”

```
Shell> stall 10000000
```

Stalls the operation for a specified number of microseconds

```
Shell> stall 10000000
Shell> _
```

tianocore

UEFI SHELL SCRIPTS

Use Scripting with UEFI Shell

UEFI Shell Scripts

The UEFI Shell can execute commands from a file, which is called a batch script file (**.nsh** files).

Benefits: These files allow users to simplify routine or repetitive tasks.

- Perform basic flow control.
- Allow branching and looping in a script.
- Allow users to control input and output and call other batch programs (known as script nesting).

Writing UEFI Shell Scripts

At the shell prompt

```
Shell> fs0:
FS0:\> edit HelloScript.nsh
```

Type : `echo Hello World`

```
UEFI EDIT helloscript.nsh                    UNICODE
echo Hello World
```

Press “F2”
Enter
Press “F3” to exit

Help Menu - Shell

```
Help

Control Key   Function Key   Command
-----------   ------------   ----------------
Ctrl-G        F1             Go To Line
Ctrl-S        F2             Save File
Ctrl-Q        F3             Exit
Ctrl-F        F4             Search
Ctrl-R        F5             Search/Replace
Ctrl-K        F6             Cut Line
Ctrl-U        F7             Paste Line
Ctrl-O        F8             Open File
Ctrl-T        F9             File Type

Use Ctrl-W to exit this help
```

Hello World Script

In the shell, **type** HelloScript for the following result:

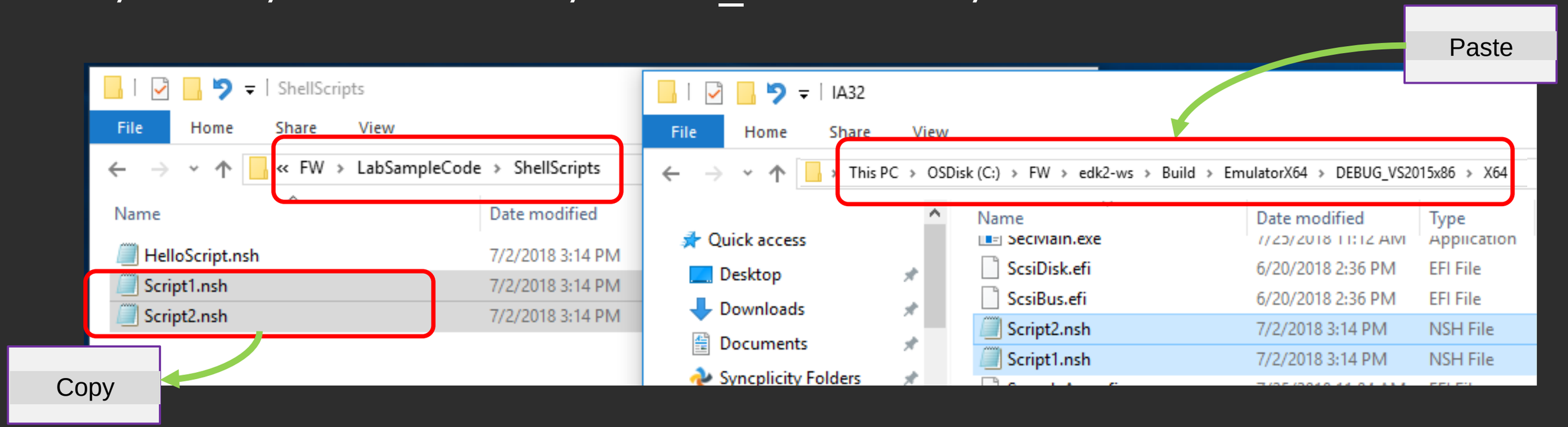
```
FS0:\> helloscript.nsh
FS0:\> echo Hello World
Hello World
FS0:\> _
```

Close the Win emulation, type: "`reset`"

```
FS0:\>
```

UEFI Shell Nested Scripts

Copy the Scripts from the `/FW/LabSampleCode/ShellScripts` to the runtime directory `C:/FW/edk2-ws/Build/EmulatorX64/DEBUG_VS201nx86/X64`



UEFI Shell Script Example

Script1.nsh

```
# Simple UEFI Shell script file
echo -off
script2.nsh
if exist %cwd%Mytime.log then
     type Mytime.log
endif
echo "%HThank you.” “%VByeBye:) %N“
```

Script2.nsh

```
# Show nested scripts
time > Mytime.log
for %a run (3 1 -1)
    echo %a counting down
endfor
```

Run UEFI Shell Scripts

From the VS command Prompt

```
C:\FW\edk2> RunEmulator.bat
```

At the Shell prompt Type

```
Shell> fs0:
FS0:\> Script1

FS0:\> Edit Script1.nsh
```

```
           0 Dir(s)
FS0:\> Script1
FS0:\> script2.nsh
FS0:\> time > Mytime.log
FS0:\> for %a run (3 1 -1)
FS0:\>     echo %a counting down
3 counting down
FS0:\> endfor
FS0:\> for %a run (3 1 -1)
FS0:\>     echo %a counting down
2 counting down
FS0:\> endfor
FS0:\> for %a run (3 1 -1)
FS0:\>     echo %a counting down
1 counting down
FS0:\> endfor
FS0:\> for %a run (3 1 -1)
FS0:\> if exist %cwd%Mytime.log then
FS0:\>       type Mytime.log
20:08:54 (UTC 00:00)

FS0:\> endif
FS0:\> echo "Thank you. ByeBye:) "
Thank you. ByeBye:)
FS0:\> _
```

Run UEFI Shell Scripts

Remove the “#” on the first line

```
UEFI EDIT Script1.nsh
echo  -off
script2.nsh
if exist %cwd%Mytime.log then
     type Mytime.log
endif
echo "%HThank you. %VByeBye:) %N"
```

Press “F2”

Enter

Press “F3” to exit

Type

```
FS0:\> Script1
```

```
FS0:\> Script1
FS0:\> echo  -off
3 counting down
2 counting down
1 counting down
20:19:52 (UTC 00:00)

Thank you. ByeBye:)
FS0:\> _
```

tianocore

UEFI SHELL GLOBAL VARIABLES

Use BCFG and DmpStore

Show the UEFI Boot Variables

At the Shell Prompt:

```
Shell> FS0:

FS0:> BCFG Boot Dump
```

```
  Desc    - UEFI BootManagerMenuApp
  DevPath - Fv(6D99E806-3D38-42C2-A095-5F4300BFD7DC)/FvFile(EEC25BDC-67F2-4D95-B
1D5-F81B2039D11D)
  Optional- N
Option: 02. Variable: Boot0002
  Desc    - UEFI Misc Device
  DevPath - VenHw(5CF32E0B-8EDF-2E44-9CDA-93205E99EC1C,00000000)/VenHw(6888A4AE-
AFCE-E84B-9102-F7B9DAE6A030,00000000)
  Optional- Y
Option: 03. Variable: Boot0003
  Desc    - UEFI Non-Block Boot Device
  DevPath - VenHw(5CF32E0B-8EDF-2E44-9CDA-93205E99EC1C,00000000)/VenHw(964E5B22-
6459-11D2-8E39-00A0C969723B,00000000)
  Optional- Y
Option: 04. Variable: Boot0004
  Desc    - UEFI BootManagerMenuApp
  DevPath - Fv(6D99E806-3D38-42C2-A095-5F4300BFD7DC)/FvFile(EEC25BDC-67F2-4D95-B
1D5-F81B2039D11D)/BootManagerMenuApp
  Optional- Y
Option: 05. Variable: Boot0000
  Desc    - UEFI Enter Setup
  DevPath - Fv(6D99E806-3D38-42C2-A095-5F4300BFD7DC)/FvFile(462CAA21-7614-4503-8
36E-8AB6F4662331)/Enter Setup
  Optional- N
FS0:\> _
```

Use the Dmpstore to Show the Boot Order

At the Shell Prompt:

```
FS0:> Dmpstore BootOrder
```

```
FS0:\> dmpstore bootorder
Variable NV+RT+BS 'EFIGlobalVariable:BootOrder' DataSize = 0x0C
  00000000: 05 00 01 00 02 00 03 00-04 00 00 00             *............*
FS0:\> _
```

Use the BCFG to Move a boot item

Use BCFG to Move the 5th boot item too 1st location.

Then verify using the “dmpstore”

(Hint: use BCFG -? –b for help menu)

The dmpstore output should look like the screen shot

Result

```
FS0:\> dmpstore bootorder
Variable NV+RT+BS 'EFIGlobalVariable:BootOrder' DataSize = 0x0C
  00000000: 00 00 05 00 01 00 02 00-03 00 04 00                *.....
```

Use the BCFG to Add a boot item

From Windows File explorer, Copy the file from the

`%WORKSPACE%\edk2\ShellPkg\OldShell\Shell_FullX64.efi` to the directory `%WORKSPACE%\Build\ . . .\X64`

Use BCFG to Add a 06 entry for a new boot option with `Shell_FullX64.efi`

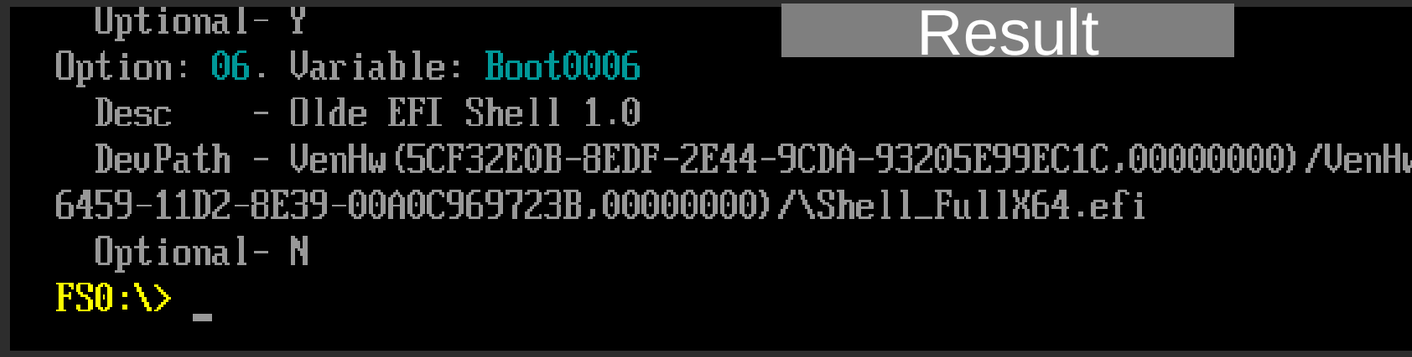
Then verify using the "`BCFG Boot Dump`"

Hint: make sure `Shell_FullX64.efi` is in the FS0: directory by doing:

```
FS0:\> Dir
```

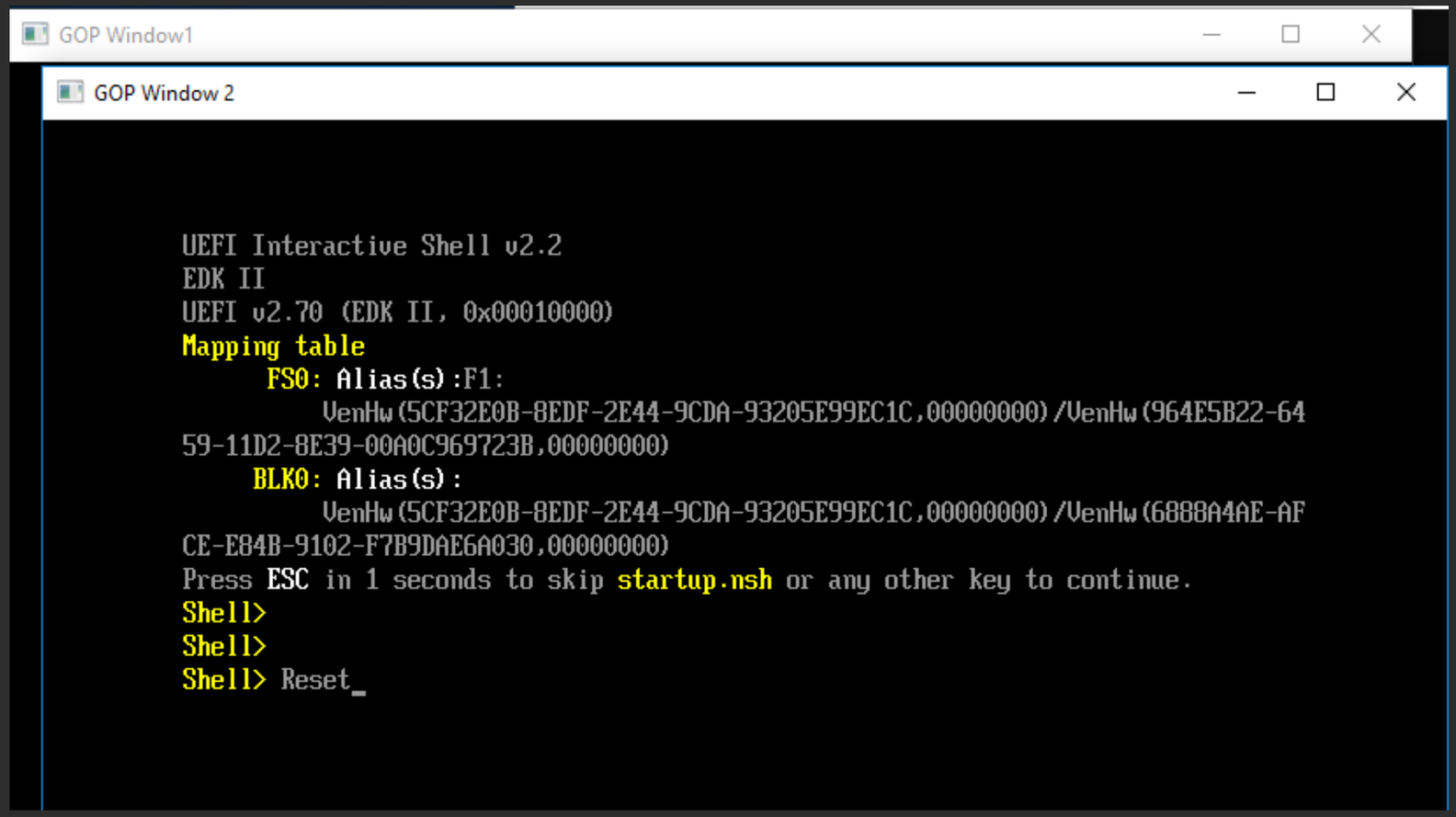
After the `bcfg` add, The output should look like

Result



Emulator at Shell Prompt

Type : “Reset” to exit



Summary

- Run UEFI Shell (Windows Emulation)
- Run UEFI Shell Commands
- Run UEFI Shell Scripts

tianocore

Questions?

Return to Main Training Page

Return to Training Table of contents for next presentation link

tianocore

ACKNOWLEDGEMENTS