tianocore

UEFI & EDK II TRAINING

UEFI SHELL LAB w/ WINDOWS NT32

tianocore.org

LESSON OBJECTIVE

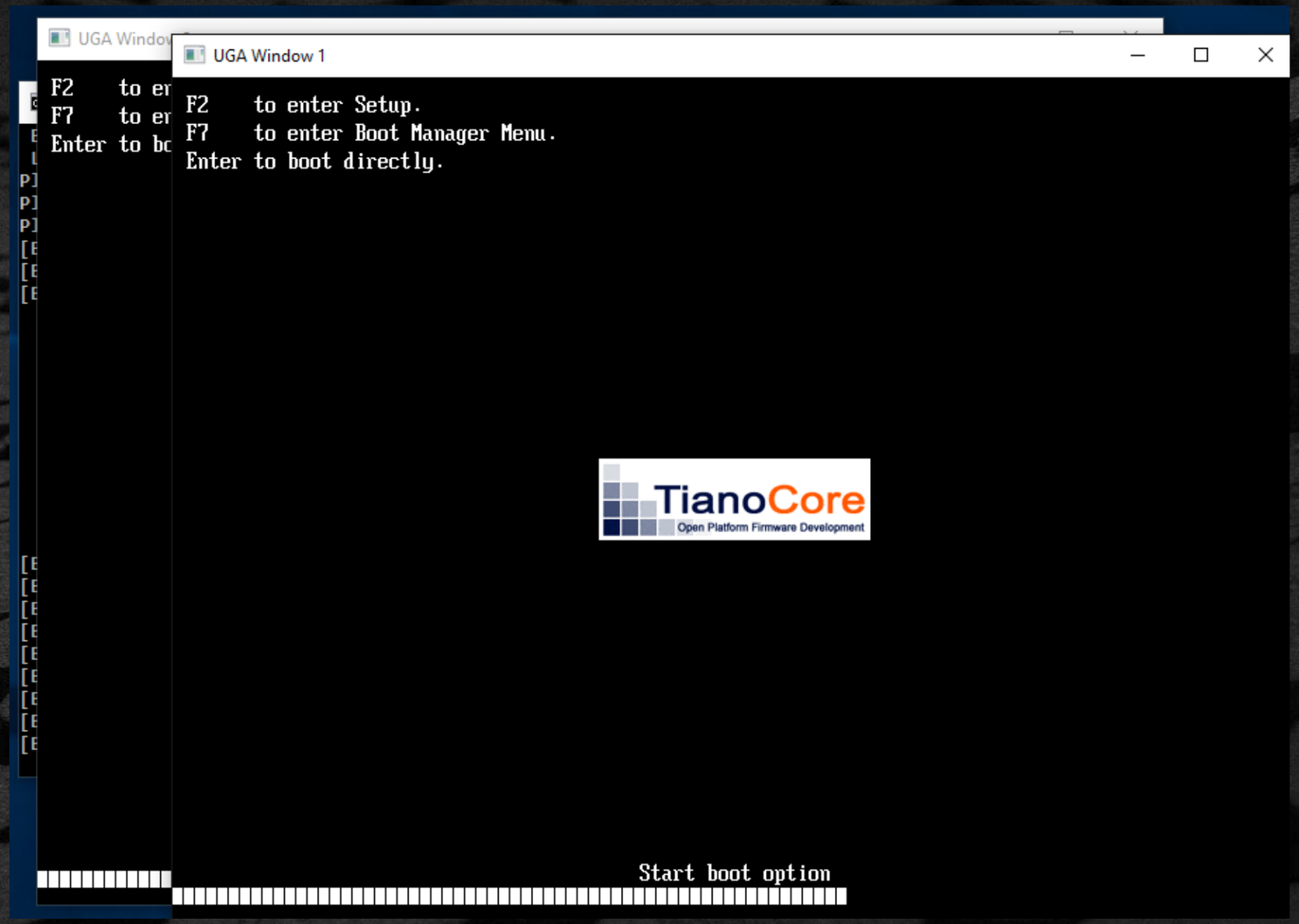
- Run UEFI Shell (Nt32 Emulation)
- Run UEFI Shell Commands
- Run UEFI Shell Scripts

UEFI SHELL LAB WITH NT32

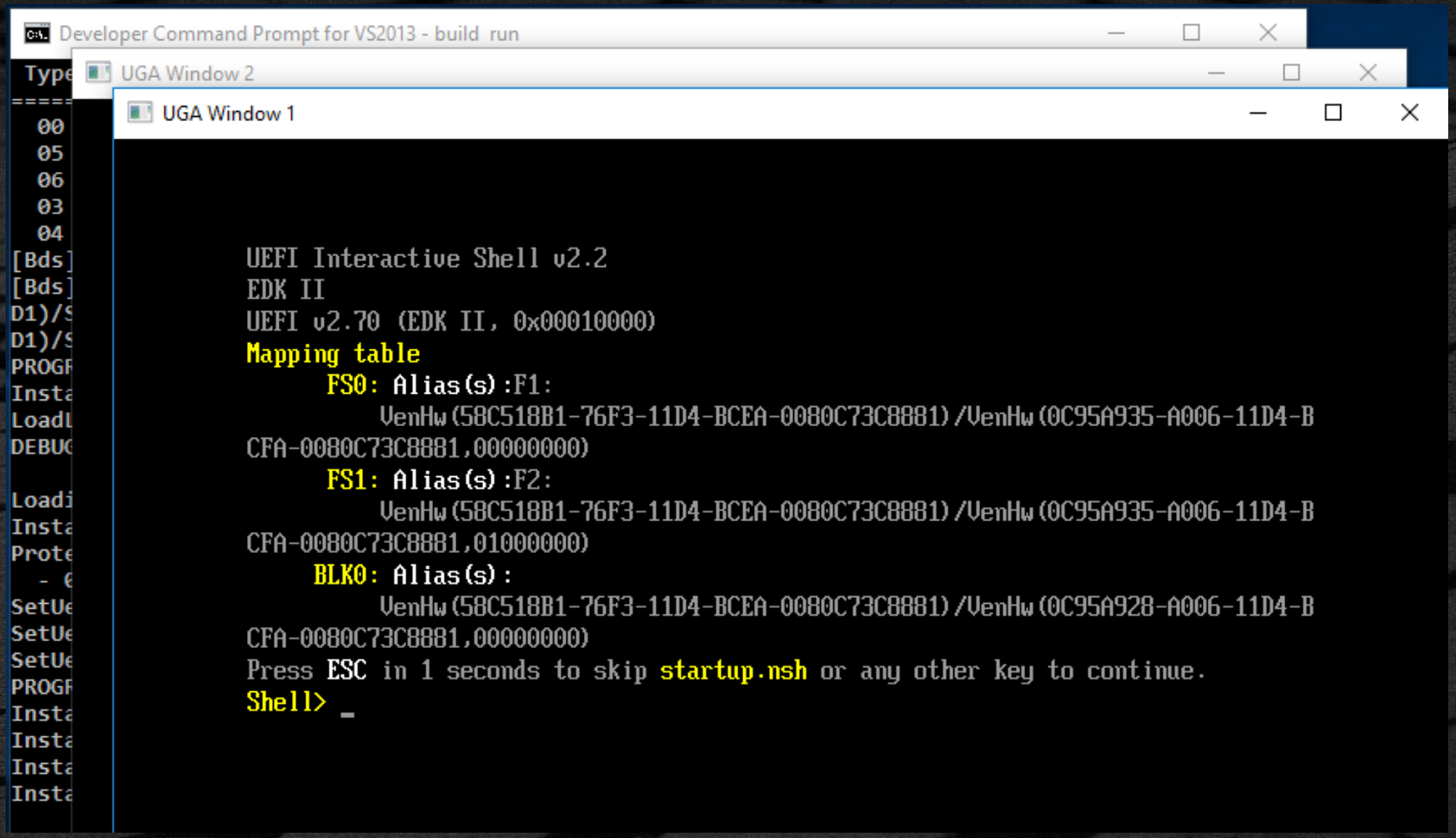
INVOKE NT32 EMULATION

From the VS command prompt

```
CD \FW\edk2
C:\FW\edk2> edksetup
C:\FW\edk2> Build
C:\FW\edk2> Build Run
```



Nt32 boot to UEFI Shell



UEFI SHELL COMMANDS

Commands from the Command Line Interface

COMMON SHELL COMMANDS FOR DEBUGGING

```
help
mm
mem
memmap
drivers
devices
devtree
dh
Load
dmpstore
stall
```

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

SHELL HELP

```
Shell> help -b
```

UGA Window 1

```
alias         - Displays, creates, or deletes UEFI Shell aliases.
attrib        - Displays or modifies the attributes of files or directories.
bcfg          - Manages the boot and driver options that are stored in NVRAM.
cd            - Displays or changes the current directory.
cls           - Clears the console output and optionally changes the background
and foreground color.
comp          - Compares the contents of two files on a byte-for-byte basis.
connect       - Binds a driver to a specific device and starts the driver.
cp            - Copies one or more files or directories to another location.
date          - Displays and sets the current date for the system.
dblk          - Displays one or more blocks from a block device.
devices       - Displays the list of devices managed by UEFI drivers.
devtree       - Displays the UEFI Driver Model compliant device tree.
dh            - Displays the device handles in the UEFI environment.
disconnect    - Disconnects one or more drivers from the specified devices.
dmem          - Displays the contents of system or device memory.
dmpstore      - Manages all UEFI variables.
drivers       - Displays the UEFI driver list.
drvcfg        - Invokes the driver configuration.
drvdiag       - Invokes the Driver Diagnostics Protocol.
echo          - Controls script file command echoing or displays a message.
edit          - Provides a full screen text editor for ASCII or UCS-2 files.
eficompress   - Compresses a file using UEFI Compression Algorithm.
efidecompress - Decompresses a file using UEFI Decompression Algorithm.
Press ENTER to continue or 'Q' break:_
```

SHELL “MM”

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

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

UGA Window 1

```
Displays or modifies MEM/MMIO/IO/PCI/PCIE address space.

MM Address [Value] [-w 1|2|4|8] [-MEM | -MMIO | -IO | -PCI | -PCIE] [-n]

  Address - Starting address in hexadecimal format.
  Value   - The value to write in hexadecimal format.
  -MEM    - Memory Address type
  -MMIO   - Memory Mapped IO Address type
  -IO     - IO Address type
  -PCI    - PCI Configuration Space Address type:
            Address format: ssssbbddffrr
              ssss - Segment
              bb   - Bus
              dd   - Device
              ff   - Function
              rr   - Register
  -PCIE   - PCIE Configuration Space Address type:
            Address format: ssssbbddffrrr
              ssss - Segment
              bb   - Bus
              dd   - Device
              ff   - Function
              rrr  - Register
  -w      - Unit size accessed in bytes:
Press ENTER to continue or 'Q' break:_
```

SHELL “MM”

```
Shell> mm 06bbb000
```

```
Shell> mm 06bbb000
MEM  0x0000000006BBB000 : 0xAF >

MEM  0x0000000006BBB001 : 0xAF >

MEM  0x0000000006BBB002 : 0xAF >

MEM  0x0000000006BBB003 : 0xAF >

MEM  0x0000000006BBB004 : 0xAF >

MEM  0x0000000006BBB005 : 0xAF >

MEM  0x0000000006BBB006 : 0xAF >

MEM  0x0000000006BBB007 : 0xAF > q

Shell> _
```

MM in can display / modify any location

Do **not** try on NT32

```
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.

```
061EC170: AF AF AF AF AF AF AF AF-AF AF AF AF AF AF AF AF  *................*
061EC180: AF AF AF AF AF AF AF AF-AF AF AF AF AF AF AF AF  *................*

Valid EFI Header at Address 00000000061EBF90
--------------------------------------------
System: Table Structure size 00000048 revision 0002001F
ConIn (000000000A3271F4) ConOut (0000000005373114) StdErr (000000000A3273A4)
Runtime Services 00000000061EBF10
Boot Services    0000000000415C40
SAL System Table 0000000000000000
ACPI Table       0000000000000000
ACPI 2.0 Table   0000000000000000
MPS Table        0000000000000000
SMBIOS Table     000000000622F000
Shell> _
```

UEFI System Table Pointer

SHELL “MEMMAP”

```
Shell> memmap
```

Displays the memory map maintained by the UEFI environment

```
Available 00000000061C0000-000000000A1BFFFF 0000000000004000 000000000000000F
MMIO      00000000021A0000-00000000021ABFFF 000000000000000C 8000000000000000
  Reserved  :          4 Pages (16,384)
  LoaderCode:        358 Pages (1,466,368)
  LoaderData:         23 Pages (94,208)
  BS_Code   :        550 Pages (2,252,800)
  BS_Data   :      3,895 Pages (15,953,920)
  RT_Code   :         64 Pages (262,144)
  RT_Data   :         64 Pages (262,144)
  ACPI Recl :          0 Pages (0)
  ACPI NVS  :          0 Pages (0)
  MMIO      :         12 Pages (49,152)
  Available :     27,810 Pages (113,909,760)
Total Memory: 128 MB (134,266,880 Bytes)
Shell> _
```

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
=== ======= = = = == == ================================== ==========
 42 0000000A B N N 1  6                      PCI Bus Driver MemoryMapped(0xB,0x
17A8E000,0x17FBDFFF)/FvFile(93B80004-9FB3-11D4-9A3A-0090273FC14D)
 44 00000011 ? N N 0  0        Block MMIO to Block IO Driver MemoryMapped(0xB,0x
17A8E000,0x17FBDFFF)/FvFile(33CB97AF-6C33-4C42-986B-07581FA366D4)
 45 00000010 ? N N 0  0                  Virtio Block Driver MemoryMapped(0xB,0x
17A8E000,0x17FBDFFF)/FvFile(11D92DFB-3CA9-4F93-BA2E-4780ED3E03B5)
 46 00000010 ? N N 0  0              Virtio SCSI Host Driver MemoryMapped(0xB,0x
17A8E000,0x17FBDFFF)/FvFile(FAB5D4F4-83C0-4AAF-8480-442D11DF6CEA)
 47 0000000A D N N 2  0   Platform Console Management Driver MemoryMapped(0xB,0x
17A8E000,0x17FBDFFF)/FvFile(51CCF399-4FDF-4E55-A45B-E123F84D456A)
 48 0000000A D N N 2  0   Platform Console Management Driver
 49 0000000A B N N 2  2              Console Splitter Driver MemoryMapped(0xB,0x
17A8E000,0x17FBDFFF)/FvFile(408EDCEC-CF6D-477C-A5A8-B4844E3DE281)
 4A 0000000A ? N N 0  0              Console Splitter Driver
 4B 0000000A ? N N 0  0              Console Splitter Driver
 4C 0000000A B N N 2  2              Console Splitter Driver
 4D 0000000A B N N 1  1              Console Splitter Driver
 51 0000000A D N N 1  0              Graphics Console Driver MemoryMapped(0xB,0x
17A8E000,0x17FBDFFF)/FvFile(CCCB0C28-4B24-11D5-9A5A-0090273FC14D)
Press ENTER to continue or 'Q' break:_
```

SHELL “DEVICES”

```
Shell> devices -b
```

Displays a list of devices that UEFI drivers manage.

```
Shell> devices -b
     T   D
     Y C I
     P F A
CTRL E G G #P #D #C  Device Name
==== = = = == == === ==========================================================
  23 R - -  0  1  11 VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)
  51 D - -  2  0   0 Primary Console Input Device
  52 D - -  2  0   0 Primary Console Output Device
  53 D - -  2  0   0 Primary Standard Error Device
  7A B - -  1  7   3 UGA Window 1
  7B B - -  1  7   3 UGA Window 2
  7C D - -  1  0   0 COM1
  7D D - -  1  0   0 COM2
  7E D - -  1  0   0 Bus Driver Console Window
  7F D - -  1  1   0 .
  80 D - -  1  1   0 ..\..\..\..\EdkShellBinPkg\Bin\Ia32\Apps
  81 D - X  1  2   0 Diskfile0
  82 D - -  1  0   0 a:RW;2880;512
  83 D - -  1  0   0 d:RO;307200;2048
  84 D - -  1  0   0 j:RW;262144;512
Shell> _
```

SHELL “DEVTREE”

```
Shell> devtree -b
```

Displays s tree of devices currently managed by UEFI drivers.

```
Ctrl[03] MemoryMapped(0xB,0x800000,0xFFFFFF)
Ctrl[04] MemoryMapped(0xB,0x17A8E000,0x17FBDFFF)
Ctrl[1B] MemoryMapped(0xB,0x17FE0000,0x17FFFFFF)
Ctrl[32] PciRoot(0x0)
  Ctrl[7A] PciRoot(0x0)/Pci(0x0,0x0)
  Ctrl[7B] PciRoot(0x0)/Pci(0x1,0x0)
    Ctrl[84] PciRoot(0x0)/Pci(0x1,0x0)/Serial(0x0)
      Ctrl[8A] PciRoot(0x0)/Pci(0x1,0x0)/Serial(0x0)/Uart(115200,8,N,1)
        Ctrl[8B] PC-ANSI Serial Console
          Ctrl[4E] Primary Console Input Device
          Ctrl[4F] Primary Console Output Device
          Ctrl[50] Primary Standard Error Device
    Ctrl[85] PciRoot(0x0)/Pci(0x1,0x0)/Serial(0x1)
    Ctrl[86] PS/2 Keyboard Device
      Ctrl[4E] Primary Console Input Device
    Ctrl[87] PciRoot(0x0)/Pci(0x1,0x0)/Acpi(PNP0303,0x1)
    Ctrl[88] ISA Floppy Drive #0
    Ctrl[89] ISA Floppy Drive #1
  Ctrl[7C] PCI IDE/ATAPI Controller
    Ctrl[80] QEMU HARDDISK
      Ctrl[82] FAT File System
    Ctrl[81] QEMU DVD-ROM
  Ctrl[7D] PciRoot(0x0)/Pci(0x1,0x3)
  Ctrl[7E] QEMU Video PCI Adapter
Press ENTER to continue or 'Q' break:_
```

SHELL HANDLE DATABASE - “DH”

```
Shell> dh -b
```

Dump Handle - Displays the device handles associated with UEFI drivers

```
Handle dump
01: 4C8A2451-C207-405B-9694-99EA13251341 LoadedImage(DxeCore)
02: Decompress
03: FirmwareVolume2 DevicePath(..3D38-42C2-A095-5F4300BFD7DC)) FirmwareVolumeBlo
ck
04: DevicePath(..ped(0xB,0x2980000,0x299FFFF)) FirmwareVolumeBlock
05: A31280AD-481E-41B6-95E8-127F4C984779 FC1BCDB0-7D31-49AA-936A-A4600D9DD083
06: 4C8A2451-C207-405B-9694-99EA13251341 ImageDevicePath(..AD6B-4F3A-B60B-F59899
003443)) LoadedImage(DevicePathDxe)
07: DevicePathFromText DevicePathToText DevicePathUtilities
08: 4C8A2451-C207-405B-9694-99EA13251341 ImageDevicePath(..87AB-47F9-A3FE-D50B76
D89541)) LoadedImage(PcdDxe)
09: GetPcdInfo GetPcdInfoProtocol Pcd Pcd
0A: 4C8A2451-C207-405B-9694-99EA13251341 ImageDevicePath(..52B5-46CD-99C3-4368AB
BACFFD)) LoadedImage(Metronome)
0B: MetronomeArch
0C: 4C8A2451-C207-405B-9694-99EA13251341 ImageDevicePath(..A7EB-4730-8C8E-CC466A
9ECC3C)) LoadedImage(ReportStatusCodeRouterRuntimeDxe)
0D: SmartCardReader RscHandler
0E: 4C8A2451-C207-405B-9694-99EA13251341 ImageDevicePath(..4E0E-4BE4-B14C-340EB4
AA5891)) LoadedImage(StatusCodeHandlerRuntimeDxe)
0F: 4C8A2451-C207-405B-9694-99EA13251341 ImageDevicePath(..847E-4E5D-AD3F-21CABF
E5E23C)) LoadedImage(WinNtOemHookStatusCodeHandlerDxe)
10: 4C8A2451-C207-405B-9694-99EA13251341 ImageDevicePath(..1644-4EF4-8944-48C4FP
ress ENTER to continue or 'Q' break:_
```

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:
```

Shell “dmpstore”

```
Shell> dmpstore –all -b
```

Display the contents of the NVRAM variables

```
Shell> dmpstore -all -b
Variable NV+BS '4C19049F-4137-4DD3-9C10-8B97A83FFDFA:MemoryTypeInformation' Data
Size = 0x40
  00000000: 0A 00 00 00 2A 00 00 00-09 00 00 00 08 00 00 00  *....*...........*
  00000010: 00 00 00 00 29 00 00 00-06 00 00 00 F2 00 00 00  *....)...........*
  00000020: 05 00 00 00 50 00 00 00-03 00 00 00 57 03 00 00  *....P.......W...*
  00000030: 04 00 00 00 AC 14 00 00-0F 00 00 00 00 00 00 00  *................*
Variable NV+RT+BS 'EFIGlobalVariable:ErrOut' DataSize = 0x49
  00000000: 02 01 0C 00 D0 41 03 0A-00 00 00 00 01 01 06 00  *.....A..........*
  00000010: 00 01 02 01 0C 00 D0 41-01 05 00 00 00 00 03 0E  *.......A........*
  00000020: 13 00 00 00 00 00 00 C2-01 00 00 00 00 00 08 01  *................*
  00000030: 01 03 0A 14 00 53 47 C1-E0 BE F9 D2 11 9A 0C 00  *.....SG.........*
  00000040: 90 27 3F C1 4D 7F FF 04-00                       *.'?.M....*
Variable NV+RT+BS 'EFIGlobalVariable:ConIn' DataSize = 0x7A
  00000000: 02 01 0C 00 D0 41 03 0A-00 00 00 00 01 01 06 00  *.....A..........*
  00000010: 00 01 02 01 0C 00 D0 41-03 03 00 00 00 00 7F 01  *.......A........*P
ress ENTER to continue or 'Q' break:_
```

SHELL “STALL”

```
Shell> stall 10000000
```

Stalls the operation for a specified number of microseconds

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

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 2.0 HelloScript.nsh                    Modified
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:

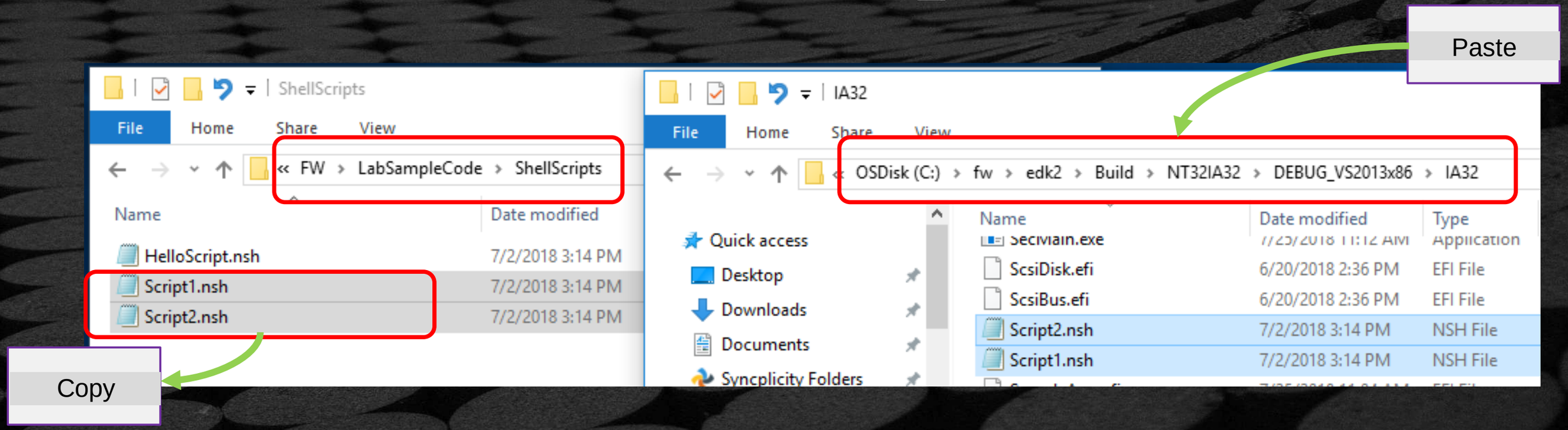
```
FS0:\> HelloScript.nsh
FS0:\> echo "Hello World"
Hello World
FS0:\> _
```

Close the Nt32 emulation, type: "`reset`"

```
FS0:\> reset
```

UEFI SHELL NESTED SCRIPTS

Copy the Scripts from the **`/FW/LabSampleCode/ShellScripts`** to the Nt32 runtime directory
`C:/FW/edk2/Build/NT32IA32/DEBUG_VS201`*`n`*`x86/IA32`



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> Build Run
```

At the Shell prompt Type

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

FS0:\> Edit Script1.nsh
```

```
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:\> _
```

SUMMARY

- Run UEFI Shell (Nt32 Emulation)
- Run UEFI Shell Commands
- Run UEFI Shell Scripts

tianocore

Questions?

tianocore