UEFI Driver Development Training Using EDK I LAB

Ruth Li
UEFI Development
Intel

Agenda

- Tools
- Download and install EDK
- Overview of EDK
- Usage:
  1. Creating directory
  2. Adding files
  3. INF files
  4. Updating master makefile
  5. Custom protocols
  6. Building the driver

Tools

- Required:
  - Microsoft Visual Studio 2003* (version 7.1)
  - Or Microsoft Visual Studio 2005* (version 8.0)

    NOTE: Config.env file will need to change for:

    USE_VC8 = YES

- Optional:
  - Microsoft Windows XP DDK* (version 3790.1830)
  - MASM* (version 6.15)
  - Intel EFI Byte Code Compiler

*Other names and brands may be claimed as the property of others.

Download and Install EDK

- www.TianoCore.org
  - Required to register for access to projects
  - EDK project - download snapshot
  - Enhanced FAT32 sub-project – download snapshot
  - Unzip EDK (IE c:\EFI\EDK\...)
  - Unzip FAT: EDK\Other\Maintained\Universal\Disk\FileSystem\

Overview of EDK

- Other – Shell Code and FAT
- Foundation – the Framework ‘glue’
- Sample
  - Code
  - Drivers
  - Platforms

EDK Build Tips

Build Tip to use	Works on
DUET	building a development boot environment
NT32	IA-32
X64	Intel® 64
IPF	IA-64
EBC	IA-32 Intel® 64 IA-64

Build NT32 environment

- Open VS.NET command prompt
  - For MSVS 8.0 do VCVARS32
- CD EDK\Sample\Platform\NT32\Build
- Set EDK_source=<location of EDK root>
- For MSVS 8.0 Edit Config.env
  - USE_VC8 = YES
- Nmake
- Wait….
- System.cmd
- Nmake run
- Reset (in EFI shell to close it)

- Leave this command prompt open for the duration of this LAB.

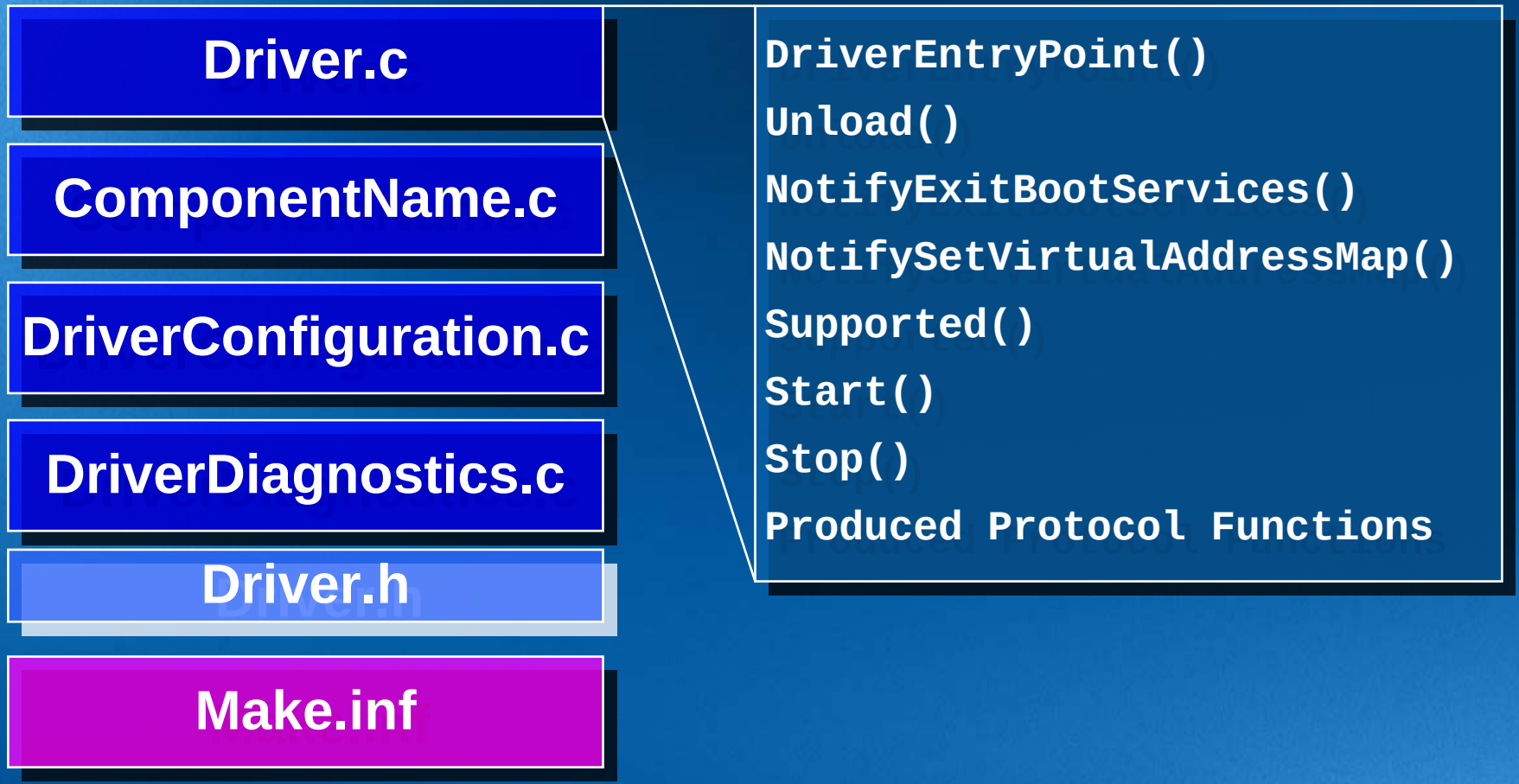
Using the EDK

1. Creating directory
2. Adding files
3. INF files
4. Updating DSC file
5. Custom protocols
6. Building the driver
7. Load the driver

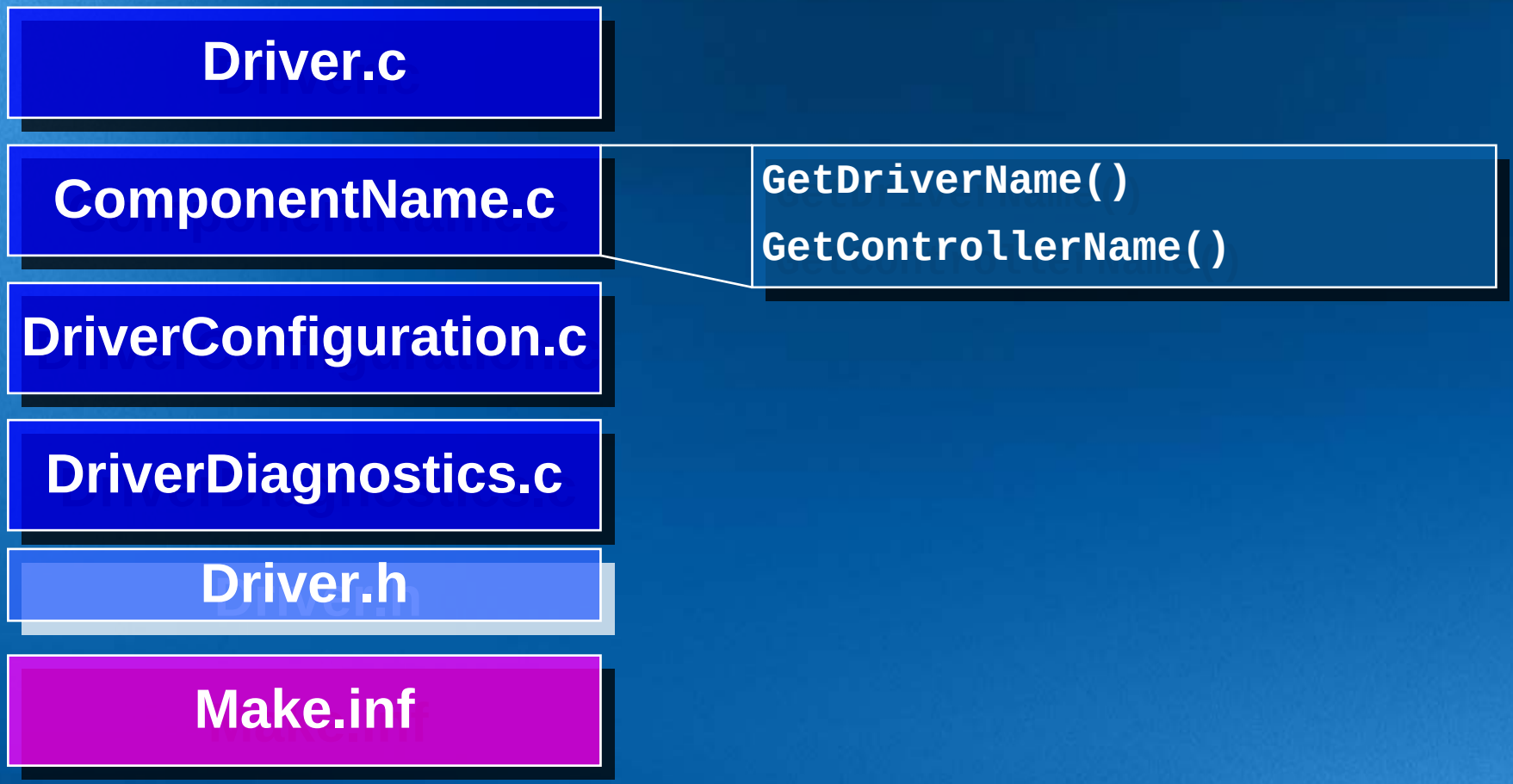
1. Create directories

- Driver Directory
  - Add a directory under Edk\Sample\Bus\PCI
  - Name that directory for your driver
    - SampleDrv in this example
- Protocol Directory
  - Add a directory under EDK\Foundation\Protocol
  - Name that directory for your driver
    - SampleDrv in this example

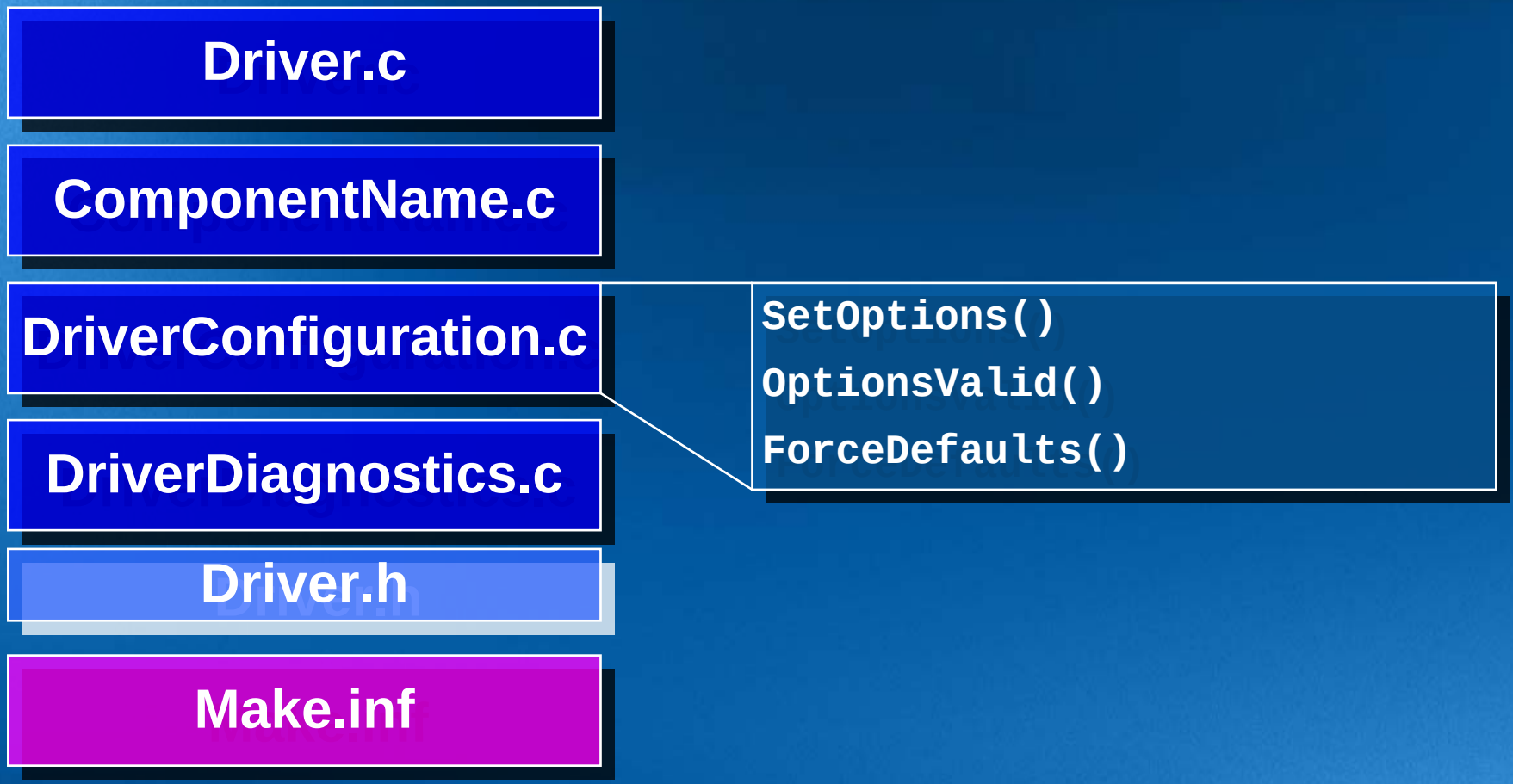
File Recommendations



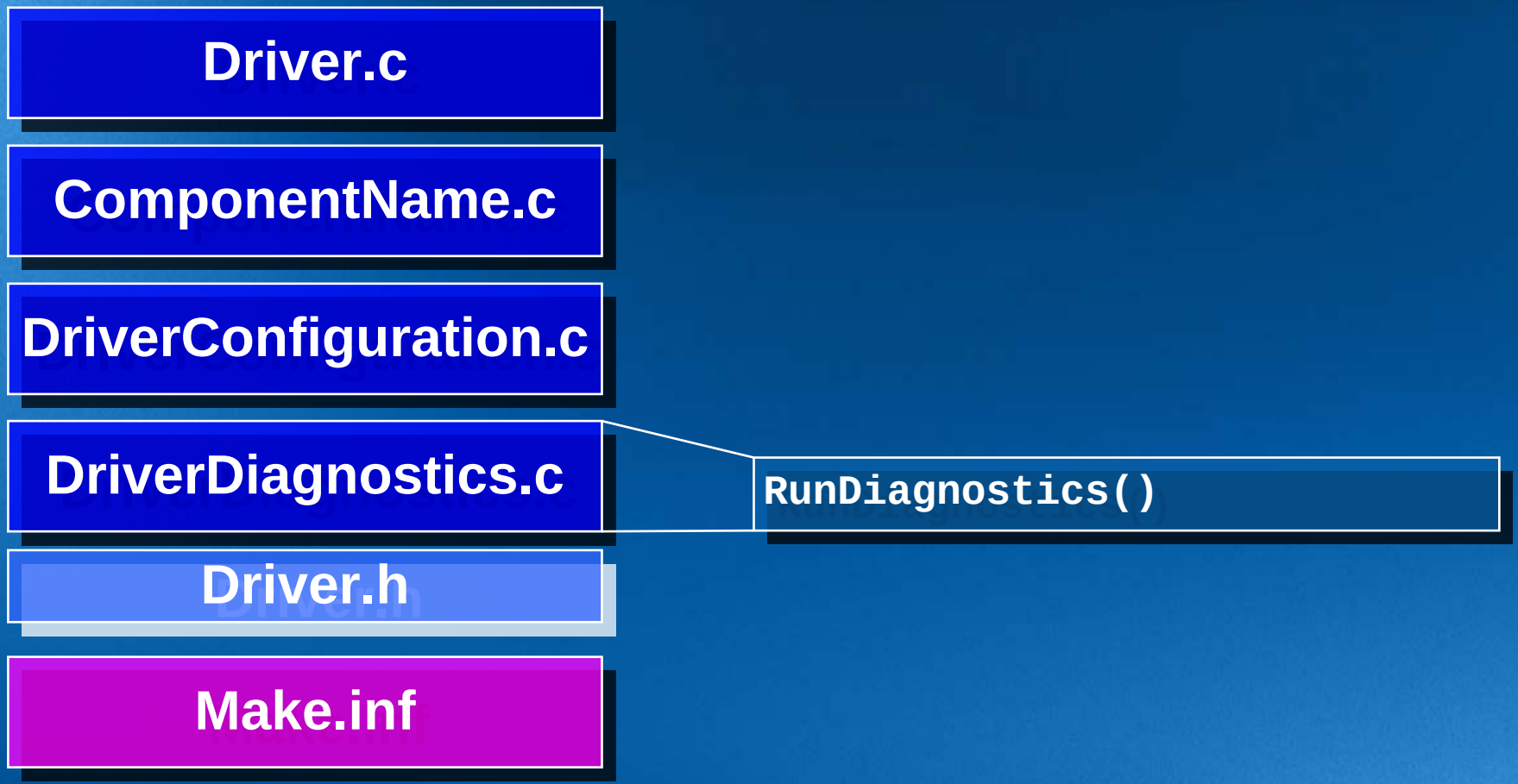
File Recommendations



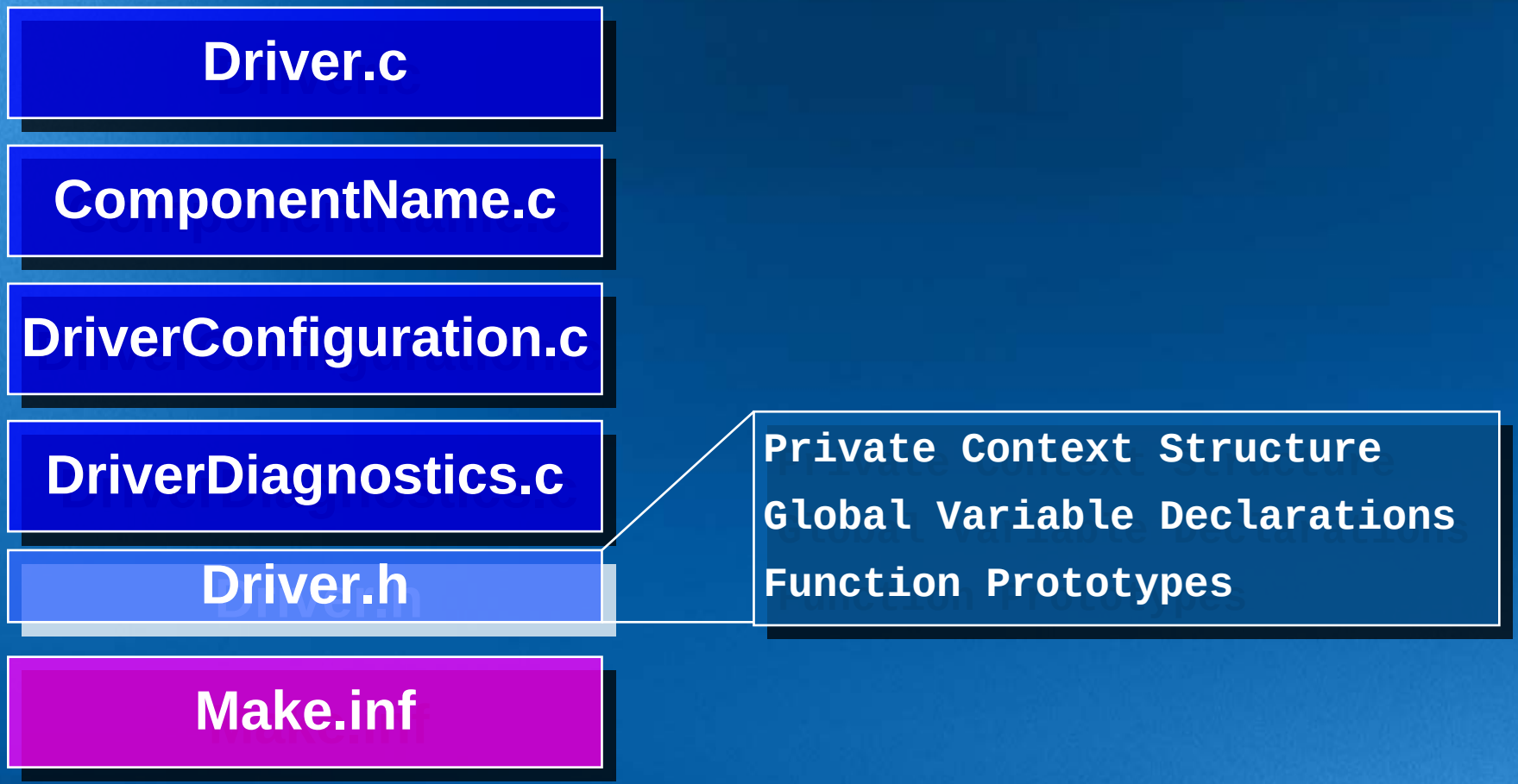
File Recommendations



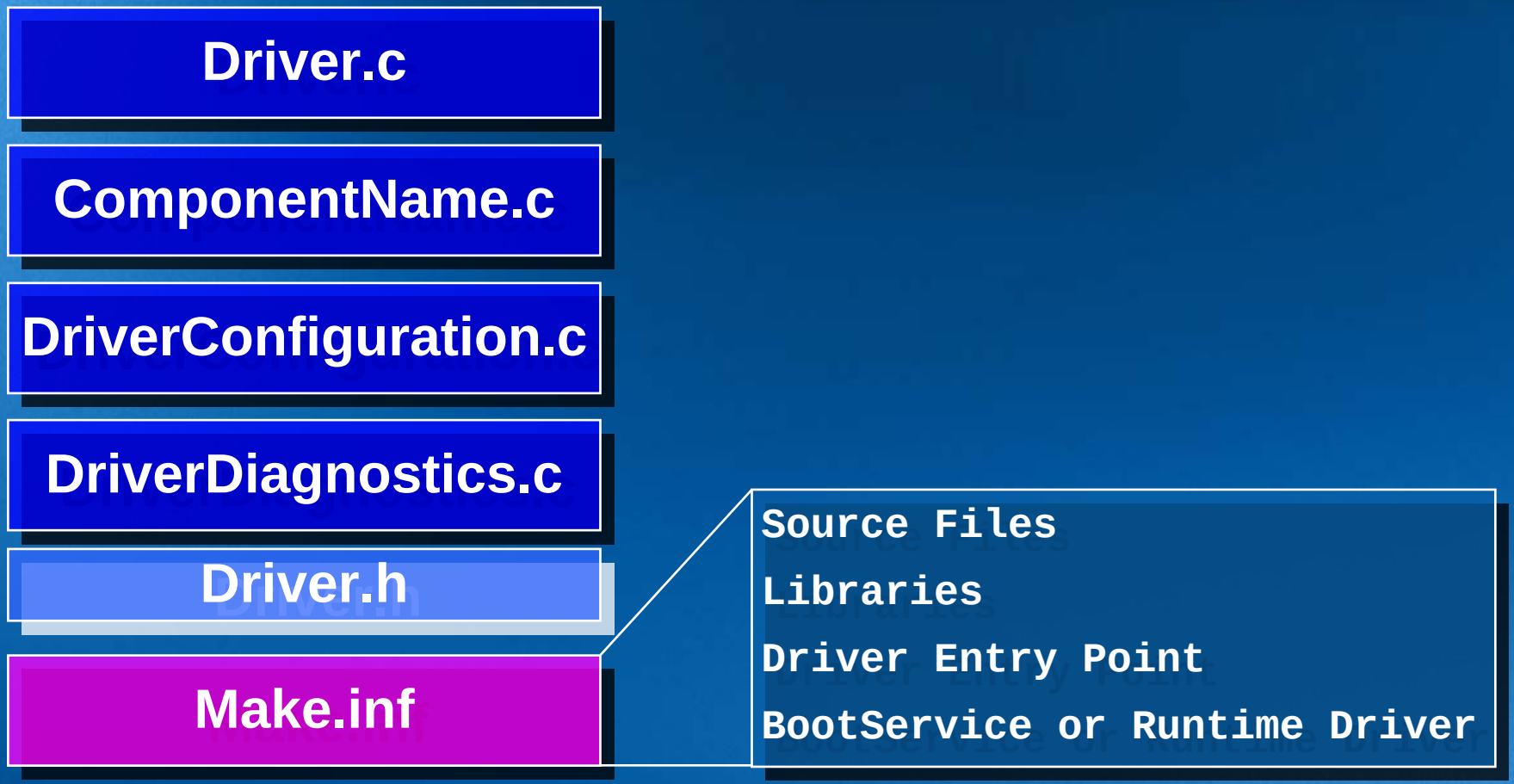
File Recommendations



File Recommendations



File Recommendations



File Recommendations

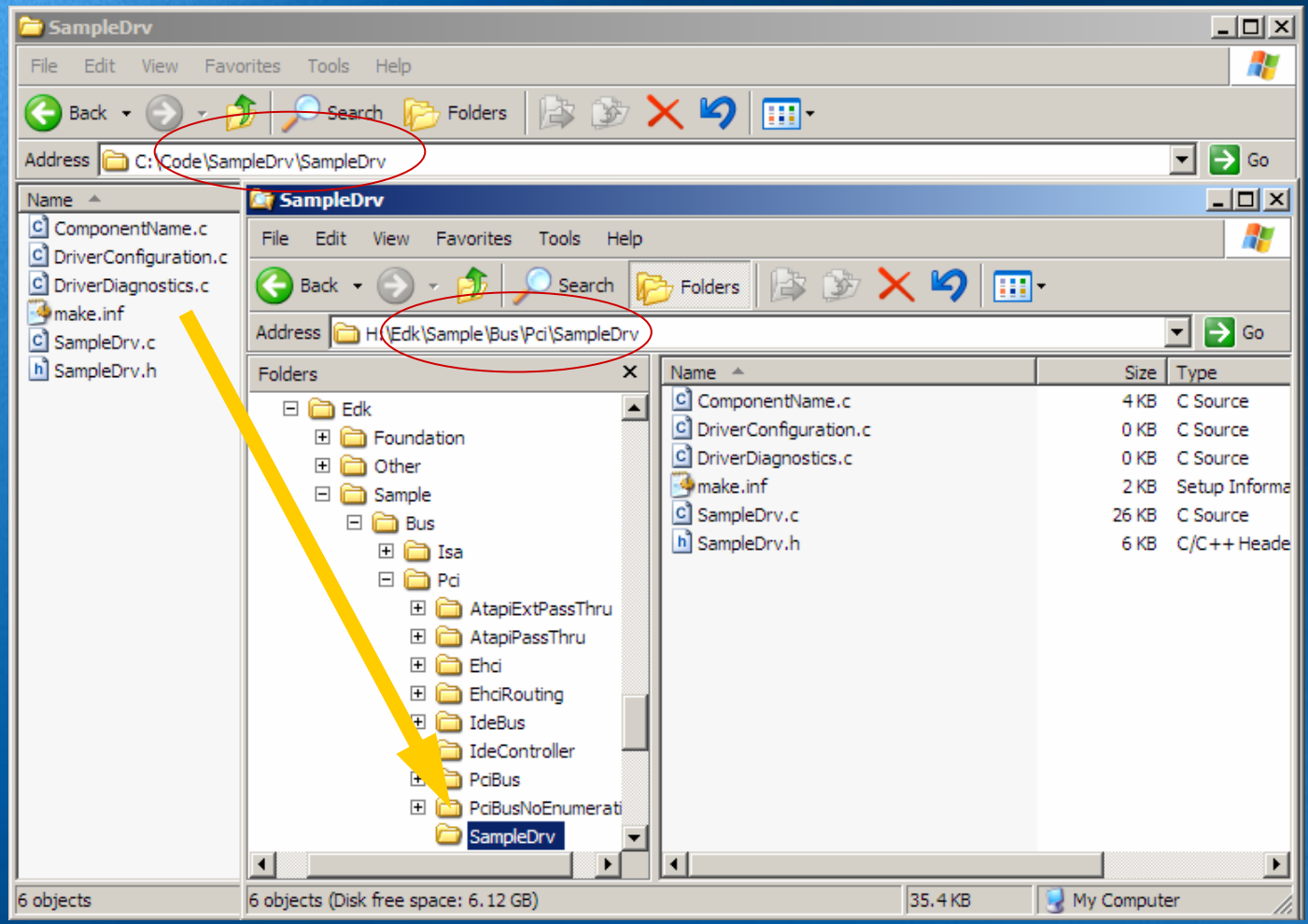
- Driver.c
- ComponentName.c
- DriverConfiguration.c
- DriverDiagnostics.c
- Driver.h
- Make.inf

- Low Complexity Drivers
  - 4 Functions
  - 1 Data Structure Designed
- Medium Complexity Drivers
  - 7 Functions
  - 1 Data Structure Designed
- High Complexity Drivers
  - 14 Functions
  - 1 Data Structure
- Very High Complexity Drivers
  - 20+ Functions
  - 3+ Data Structures Designed

2. Adding files

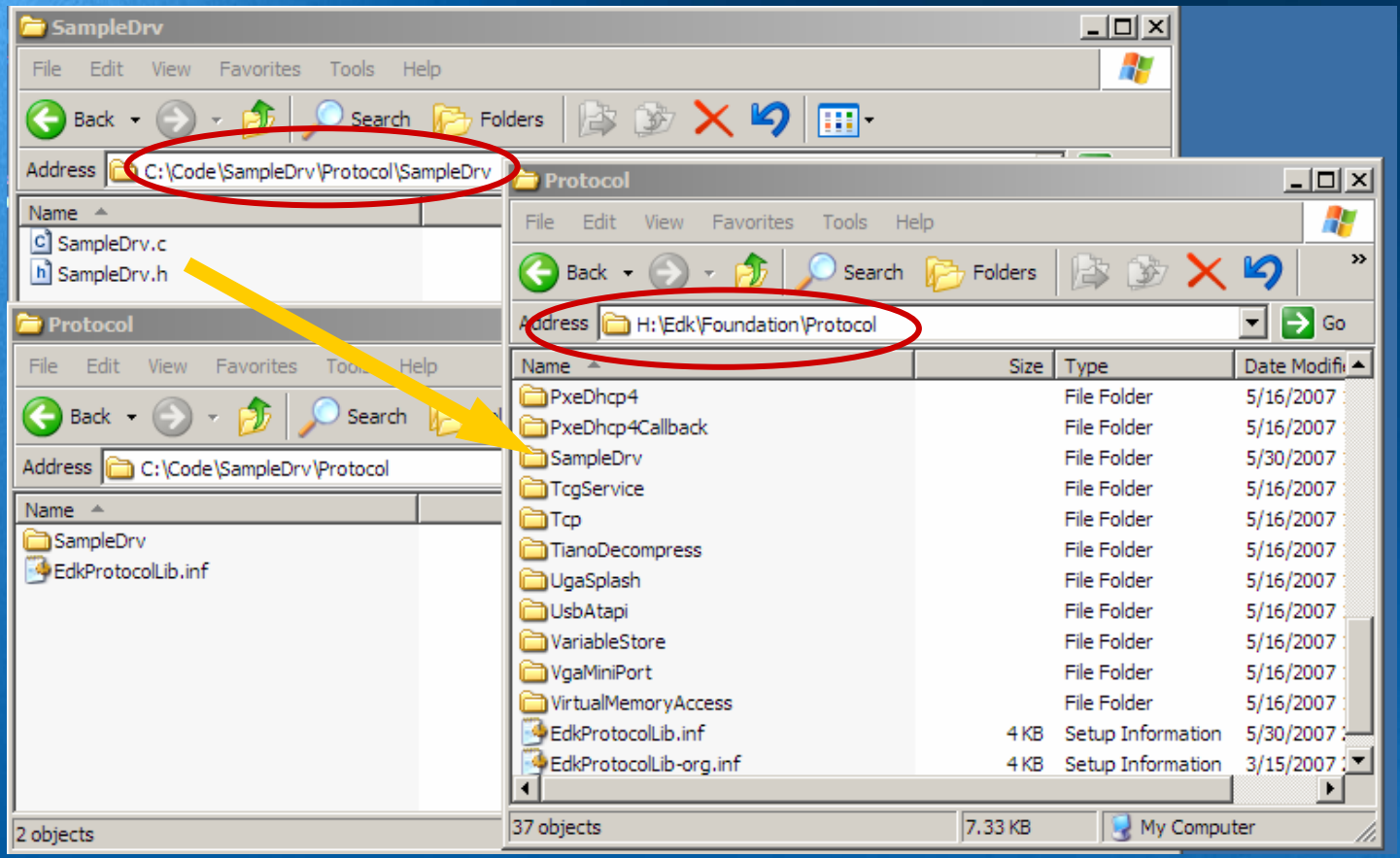
- Add files to the driver directory (step 1)
  - SampleDrv.c, SampleDrv.h, ComponentName.c, Make.inf are part of every driver.
  - DriverDiagnostics.c and DriverConfiguration.c are not required in your driver.
  - Look on the CD \Code\SampleDrv\SampleDrv for the files for the demonstration.
- Add files to the protocol directory
  - SampleDrv.c, SampleDrv.h
  - Look on the CD \Code\SampleDrv\Protocol for the files for the demonstration.

Copy Driver Files



Create the Directory SampleDrv under Sample\Bus\PCI and copy the files

Copy Protocol Files



Create Dir SampelDrv under the EDK\Foundation\Protocol Dir

3. INF files

- Defines the build options for the driver
  - Image Entry Point
  - Required libraries
  - Include directories
- Only 1 resides in the driver's directory
  - Sample driver example has a "make.inf" file

INF Example

- [defines]
- BASE_NAME = BlankDrv
- FILE_GUID = 5bc2ba35-838b-42bc-acb8-ec3c76168fce
- COMPONENT_TYPE = BS_DRIVER

- [sources.common]
- BlankDrv.c
- BlankDrv.h

- [libraries.common]
- EdkGuidLib
- EdkProtocolLib
- EfiProtocolLib
- EfiDriverLib

- [includes.common]
- $(EDK_SOURCE)\Foundation
- $(EDK_SOURCE)\Foundation\Efi
- $(EDK_SOURCE)\Foundation\Framework

- [nmake.common]
- IMAGE_ENTRY_POINT=InitializeSampleDrvDriver

Unique GUID

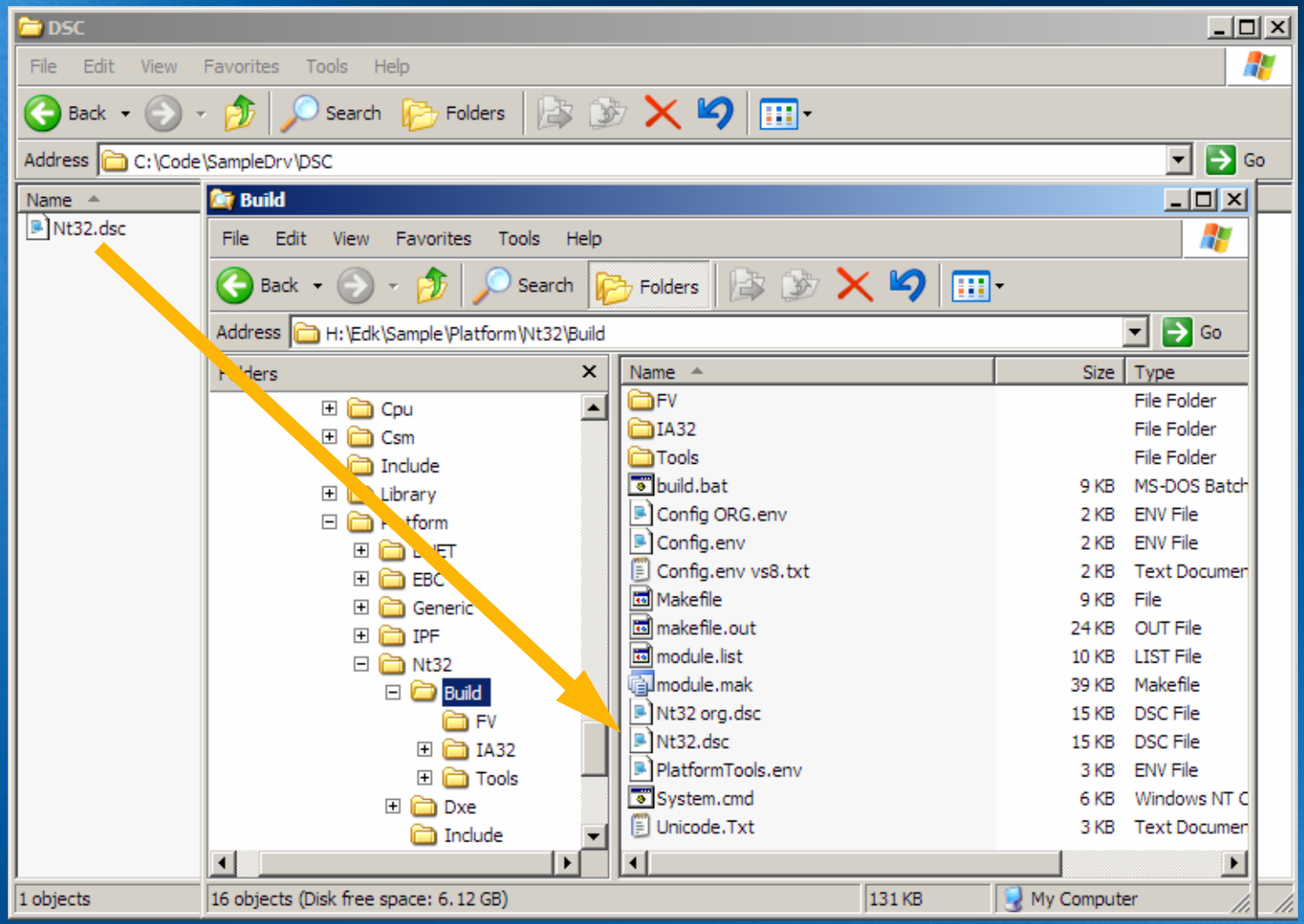
Entry Point

4. Updating DSC file

- Master DSC is updated so that the automated build process knows that the driver exists.
- It will then call into the inf for your driver.

- Look for #DriverTraining in the NT32 file in the Code\SampleDrv\DSC directory.

Nt32.DSC


DSC
File Edit View Favorites Tools Help
Back Search Folders
Address C:\Code\SampleDrv\DSC
Go
Name
Nt32.dsc
Build
File Edit View Favorites Tools Help
Back Search Folders
Address H:\Edk\Sample\Platform\Nt32\Build
Go
Cpu
Csm
Include
Library
EBC
Generic
IPF
Nt32
Build
FV
IA32
Tools
Dxe
Include
Name
Size
Type
FV File Folder
IA32 File Folder
Tools File Folder
build.bat 9 KB MS-DOS Batch
Config ORG.env 2 KB ENV File
Config.env 2 KB ENV File
Config.env vs8.txt 2 KB Text Documer
Makefile 9 KB File
makefile.out 24 KB OUT File
module.list 10 KB LIST File
module.mak 39 KB Makefile
Nt32 org.dsc 15 KB DSC File
Nt32.dsc 15 KB DSC File
PlatformTools.env 3 KB ENV File
System.cmd 6 KB Windows NT C
Unicode.Txt 3 KB Text Documer
1 objects
16 objects (Disk free space: 6.12 GB)
131 KB
My Computer

DSC Example

- Sample\Bus\Usb\UsbMouse\Dxe\UsbMouse.inf
- Sample\Universal\Network\PxeBc\Dxe\BC.inf
- Sample\Universal\Network\PxeDhcp4\Dxe\Dhcp4.inf
- Sample\Universal\Network\Snp32_64\Dxe\SNP.inf

- #DriverTraining
- Sample\Bus\PCI\BlankDrv\make.inf FV=NULL

Remove the “#” comment in the Nt32.DSC file

5. Custom protocols

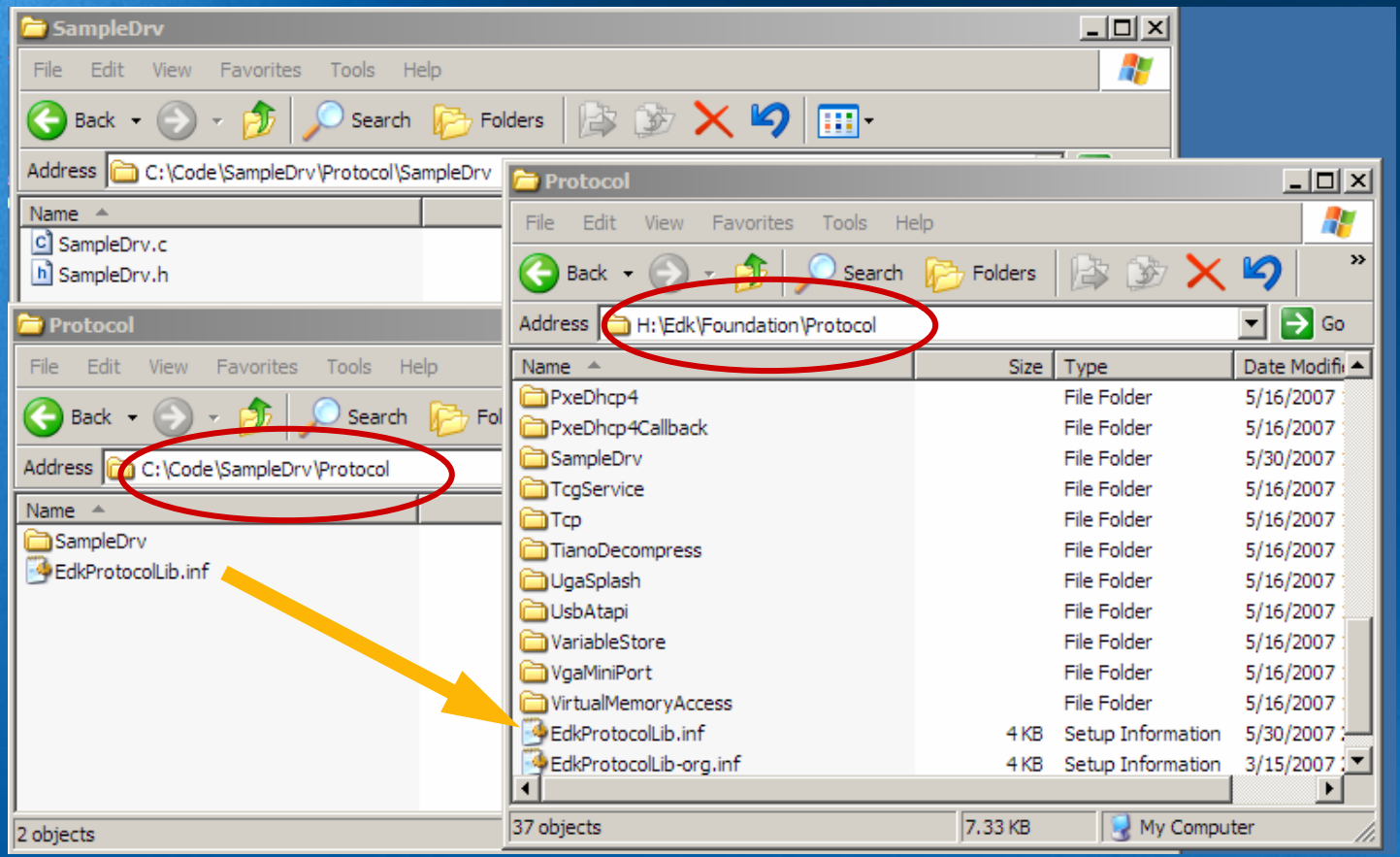
- Add the protocol files to the protocol directory
- Update the EdkProtocolLib.inf with:

  sampleDrv\SampleDrv.c

  sampleDrv\SampleDrv.h

  - Now some image could use (consume or produce) your protocols

EdkProtocolLib.INF File



6. Test building driver

- Go back to VS.NET command prompt
- Nmake

7. Load the driver

- Go back to VS.NET command prompt

```
>Nmake run
>Fsnt0:
>Load Sampledrv.efi
>Dh
```

- The –b option stops at each page break in the EFI shell.
- The SampleDrv should be loaded at the end

Do It Yourself Lab

- How do you see which child process are managed by your driver?
- What if your driver does not manage anything?
- How would you “unload” the driver?
- How would you debug using Visual studio?

- Q/A?


intel®
Leap ahead™

Do IT Yourself LAB

Which child process are managed by your driver

- The EFI shell command “drivers” will show if the driver is managing anything

```
>drivers
```

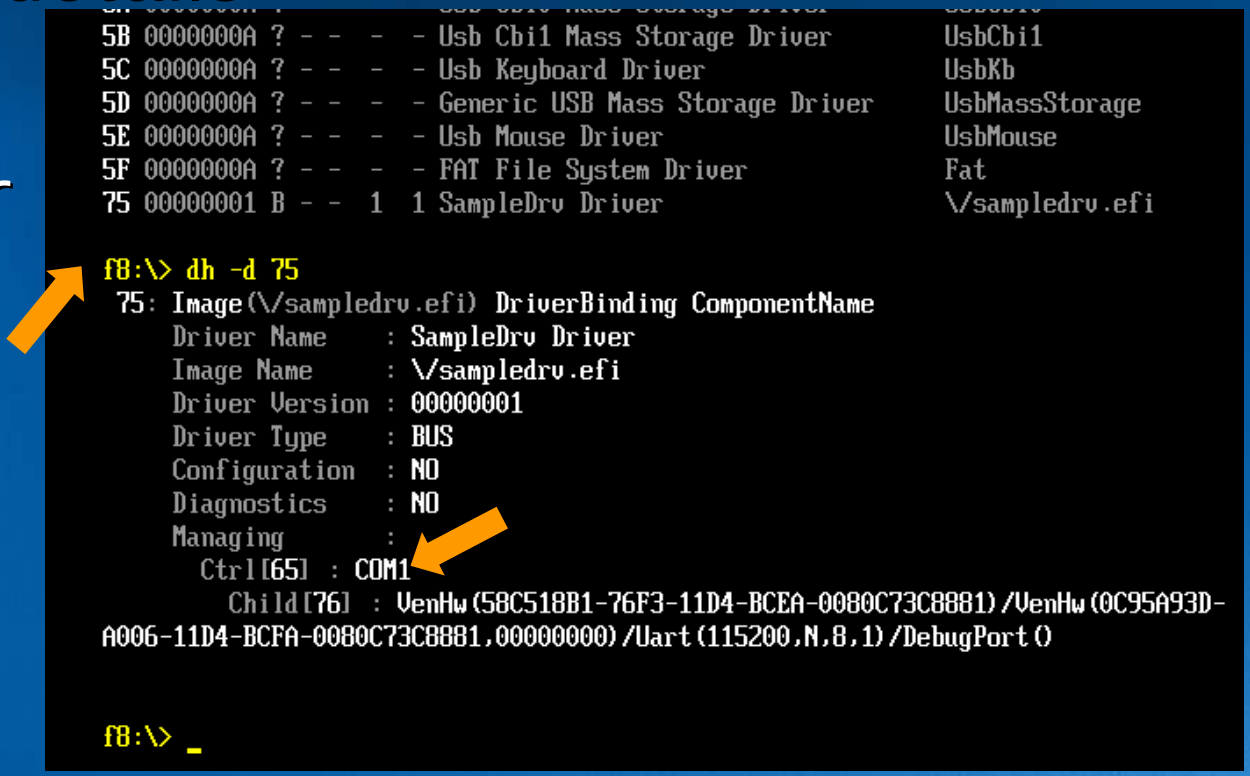
```
53 0000000A B - -  1 15 Windows Bus Driver                 WinNtBusDriver
54 0000000A D - -  2  - Windows GOP Driver                 WinNtGop
55 0000000A ? - -  -  - PCI IDE/ATAPI Bus Driver           IdeBus
56 0000000A ? - -  -  - Usb Uhci Driver                    Uhci
57 00000010 ? - -  -  - <UNKNOWN>                          Undi
58 0000000A ? - -  -  - Usb Bot Mass Storage Driver        UsbBot
59 0000000A ? - -  -  - USB Bus Driver                     UsbBus
5A 0000000A ? - -  -  - Usb Cbi0 Mass Storage Driver       UsbCbi0
5B 0000000A ? - -  -  - Usb Cbi1 Mass Storage Driver       UsbCbi1
5C 0000000A ? - -  -  - Usb Keyboard Driver                UsbKb
5D 0000000A ? - -  -  - Generic USB Mass Storage Driver    UsbMassStorage
5E 0000000A ? - -  -  - Usb Mouse Driver                   UsbMouse
5F 0000000A ? - -  -  - FAT File System Driver             Fat
75 00000001 B - -  1  1 SampleDrv Driver                   \sampledrv.efi

fs0:\> _
```

This tells us the driver is managing a child process

Which child process are managed by your driver

- The “dh” command with “–d #” option will show further details

```
>dh –d 75
Using the
Handle number
in your
emulation
```



Notice that the COM1 resource is being managed by the SampleDrv

Which child process are managed by your driver

- Why would your driver not manage anything?

  >dh –d 74

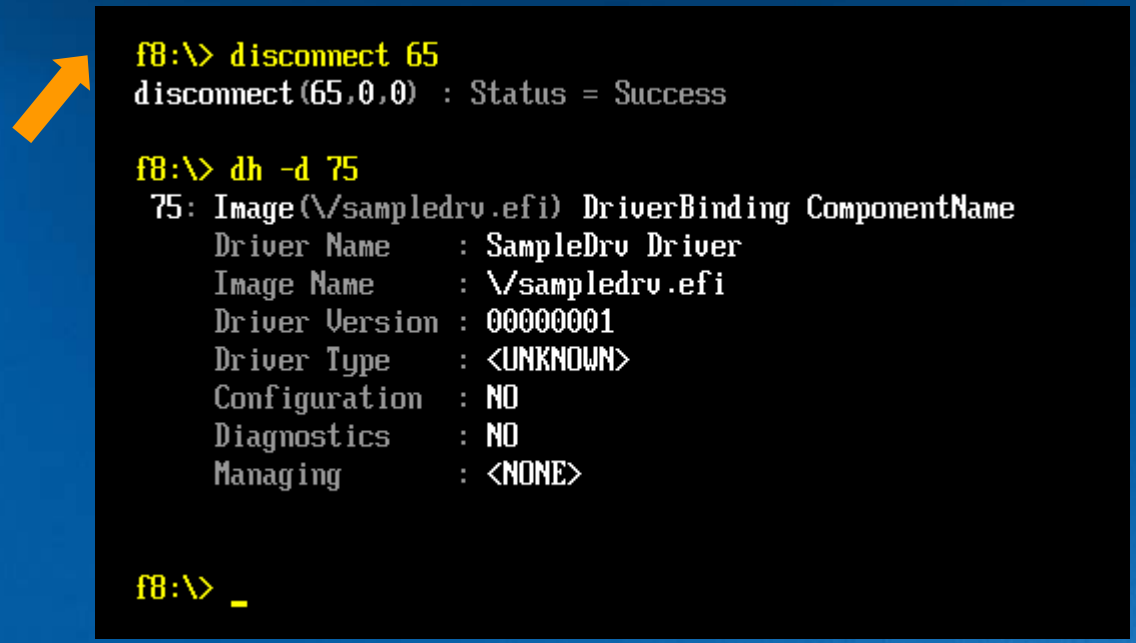
```
fs0:\> dh -d 74
 74: Image(\sampledrv.efi) DriverBinding ComponentName
     Driver Name     : SampleDrv Driver
     Image Name      : \sampledrv.efi
     Driver Version  : 00000001
     Driver Type     : <UNKNOWN>
     Configuration   : NO
     Diagnostics     : NO
     Managing        : <NONE>

fs0:\> _
```

- If something else has taken the resource then the driver will not manage
  - Example – If Hyper-terminal is on COM1

How would you “unload” the driver

- The “disconnect” command will remove the child processes managed by the driver

```
>disconnect 76
>disconnect 65
>dh -d 75
```

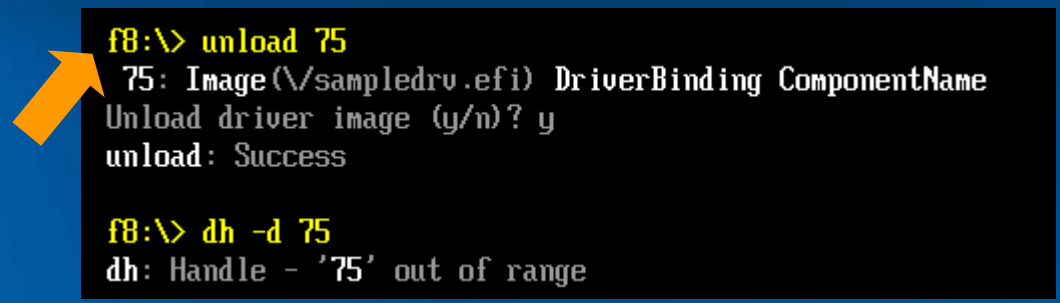


- Notice that there is nothing managed by the Driver after the disconnect.

How would you “unload” the driver

- Use the EFI Shell “unload” command to unload the driver from the driver handle data base.

```
>unload 75
```



- Notice that the handle has been removed

How to Debug using Visual Studio

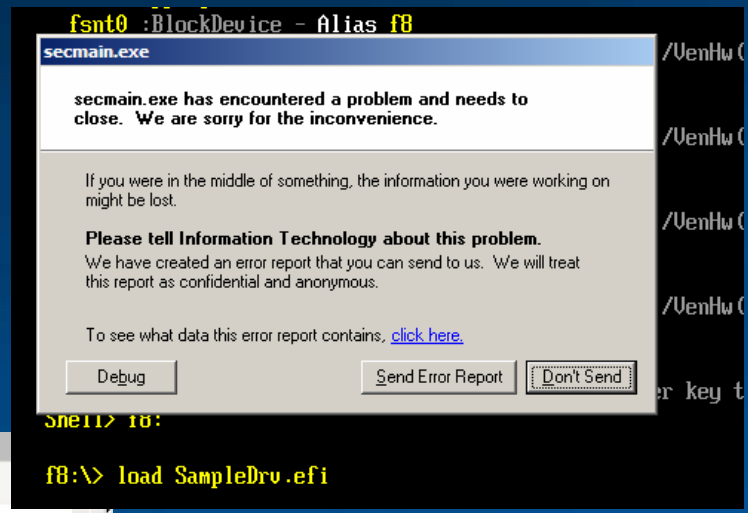
- Edit the file “SampleDrv.c” in the Edk\Sample\Bus\Pci\SampleDrv directory and uncomment the “EFI_BREAKPOINT();” Macro (line 86)
- Go back to VS.NET command prompt

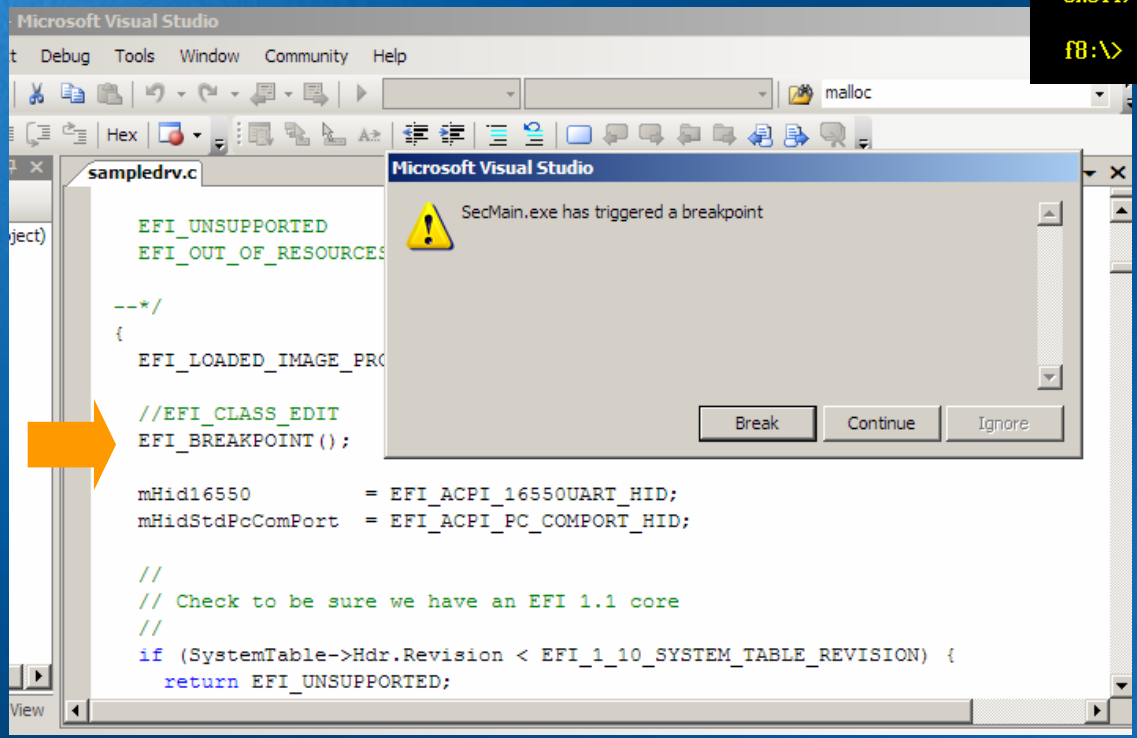
```
>Nmake
>Nmake run
>Fsnt0:
>Load SampleDrv.efi
```

- The Visual Studio debug Screen will prompt at the Breakpoint

How to Debug using Visual Studio

- Select “Debug”
- Then “Break”
- Now Visual Studio can be used for debugging





Notice the Driver source code is brought into the Visual Studio

Use “F10” to Step over


intel®
Leap ahead™