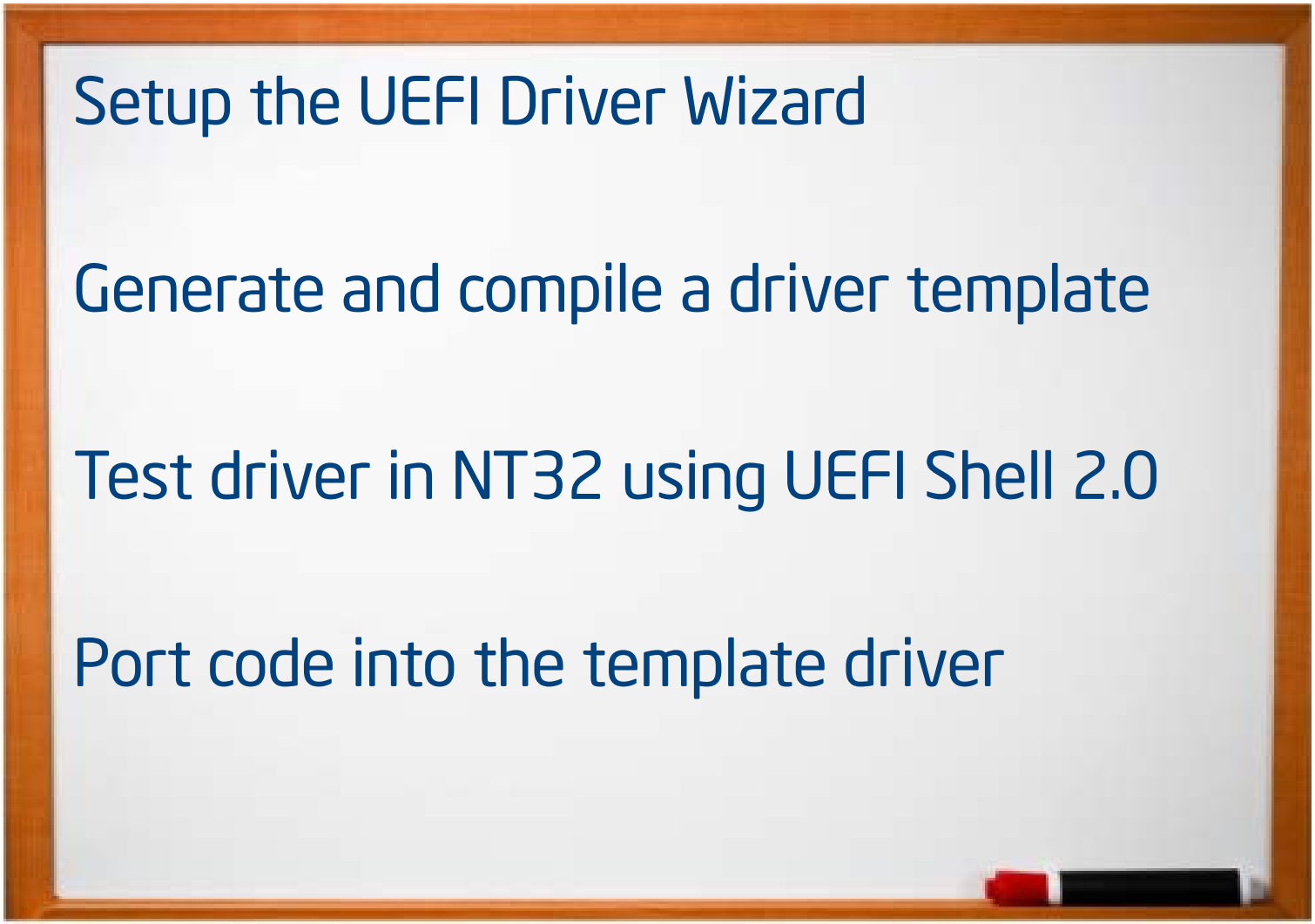
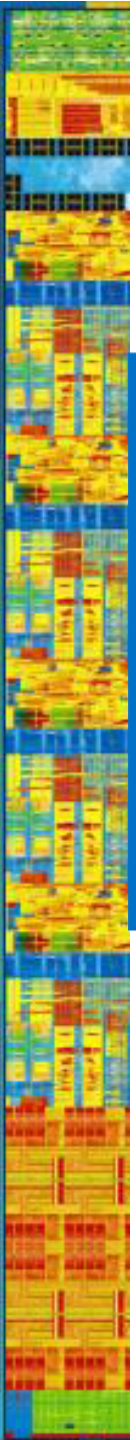
UEFI & EDK II Base Training

Lesson 8
UEFI Driver Wizard Lab

Intel Corporation
Software and Services Group

Lesson 8 Objectives





UEFI Driver Review

Defining a UEFI Driver?

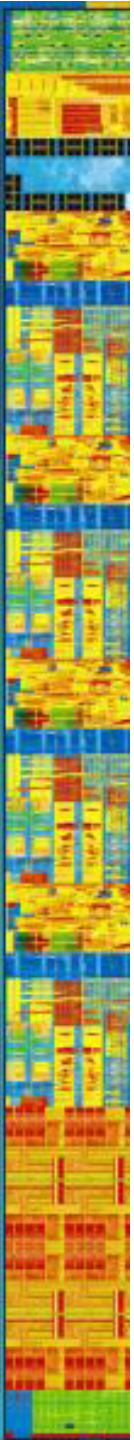
UEFI Loadable Image

May produce/consume protocols

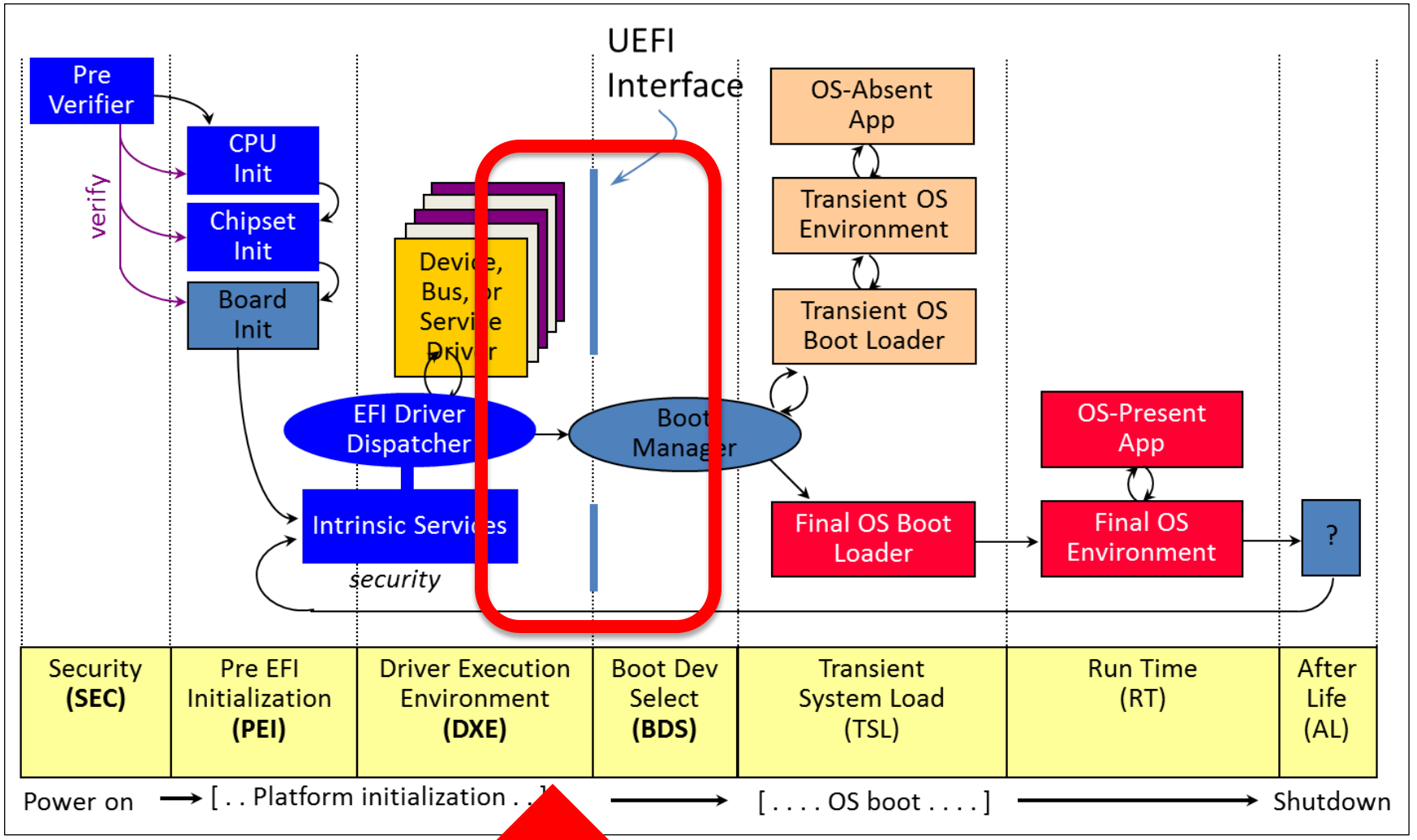
Supports complex bus hierarchies

Driver Binding Protocol matches drivers to devices, adds version management

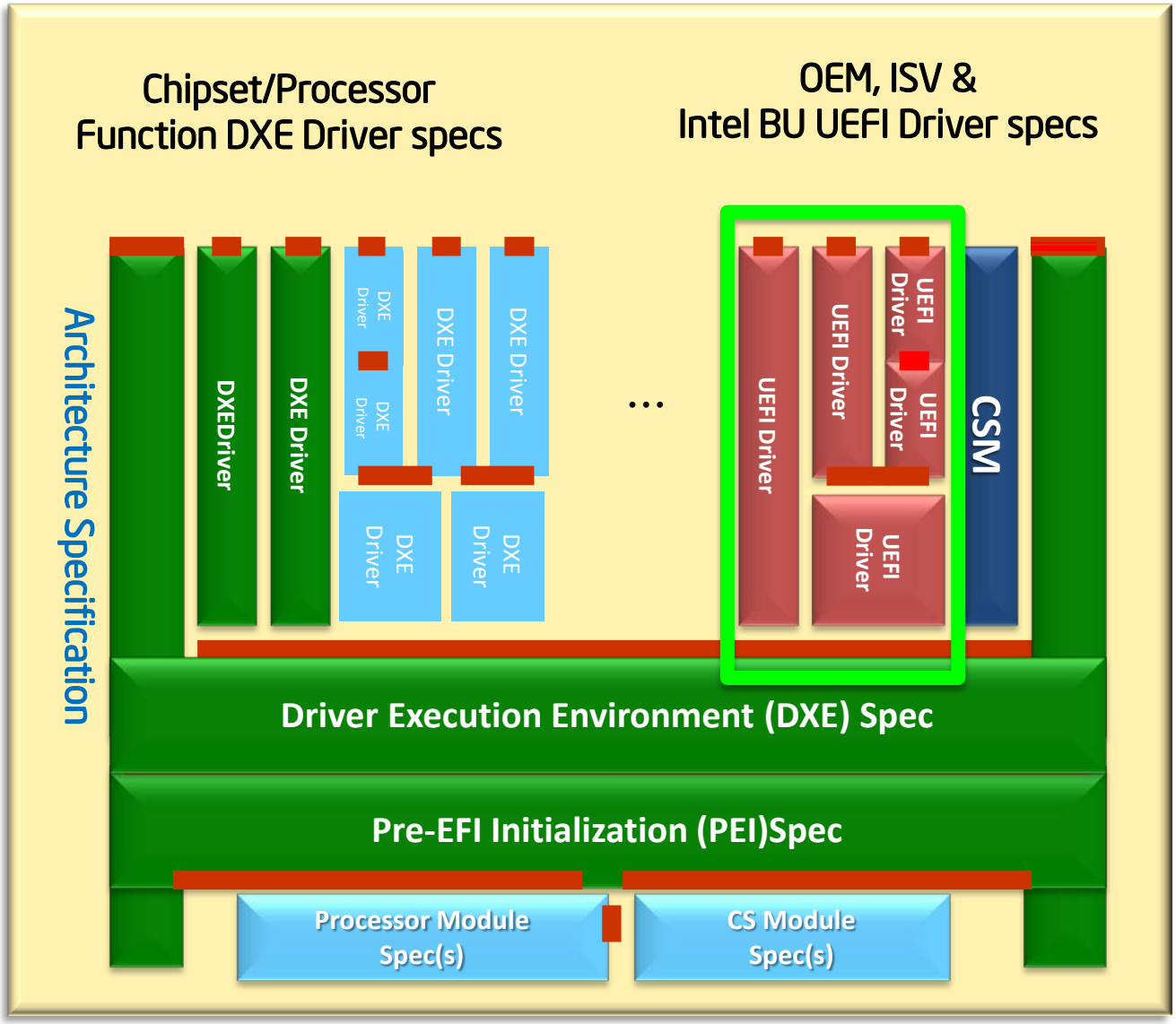
Supports specific hardware, can be unloaded or override an existing driver



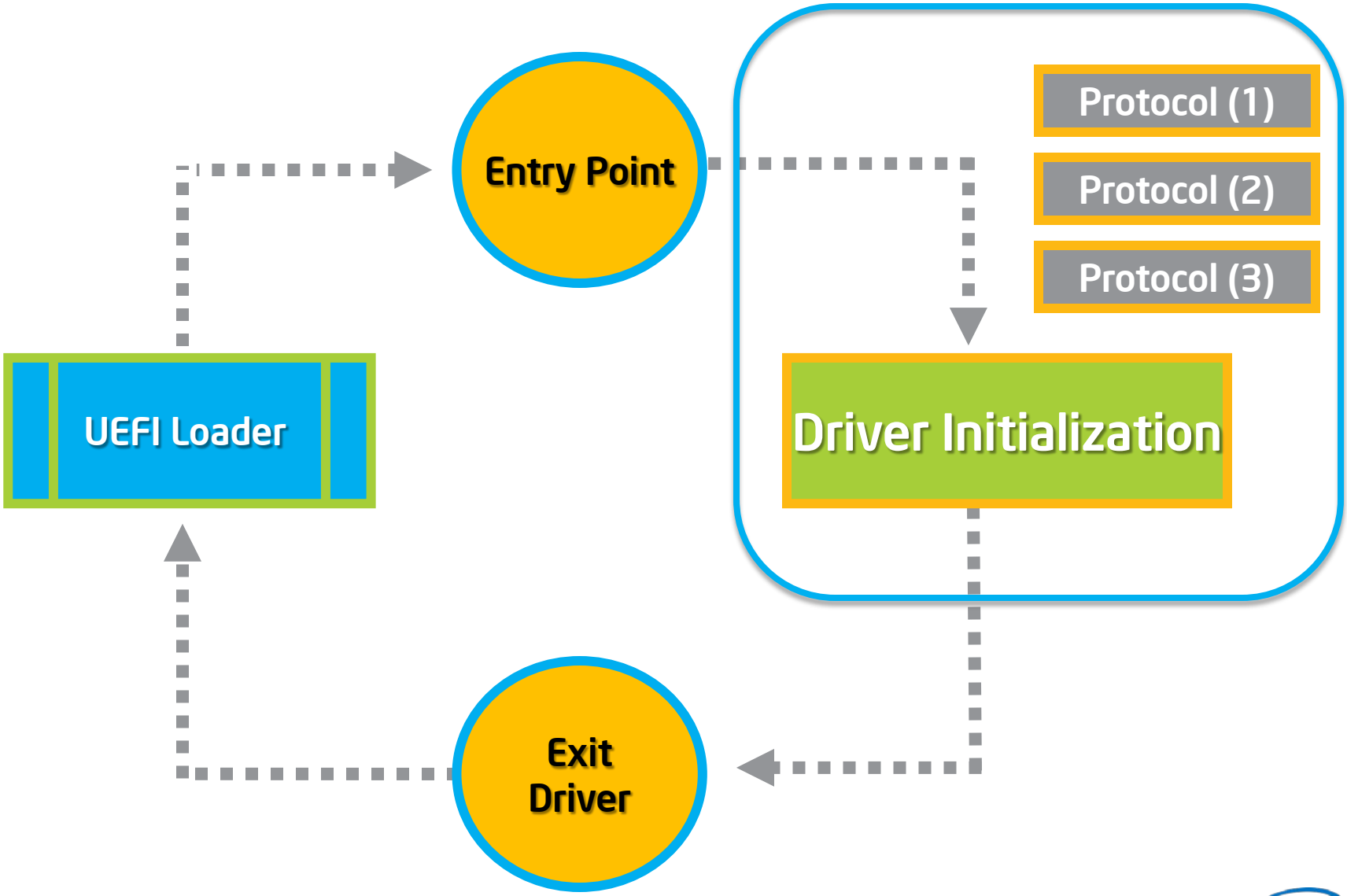
UEFI Drivers - Location



UEFI Drivers - Location



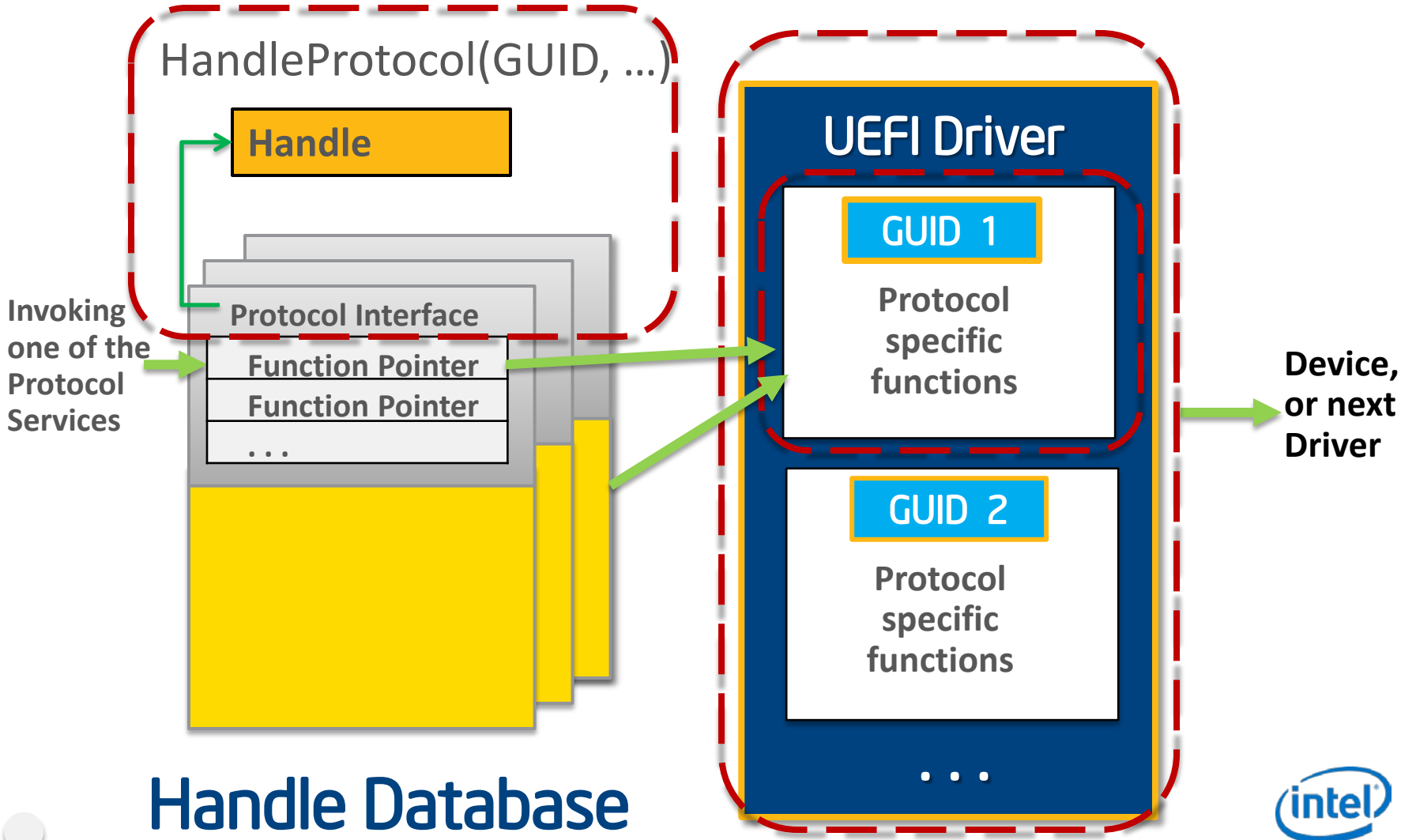
UEFI Drivers Vs. Applications



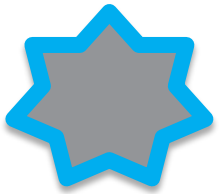
Drivers Produce Protocols

`InstallProtocolInterface`

Construction of a protocol



Driver Binding Protocol



`Supported()`

Determines if a driver supports a controller

`Start()`

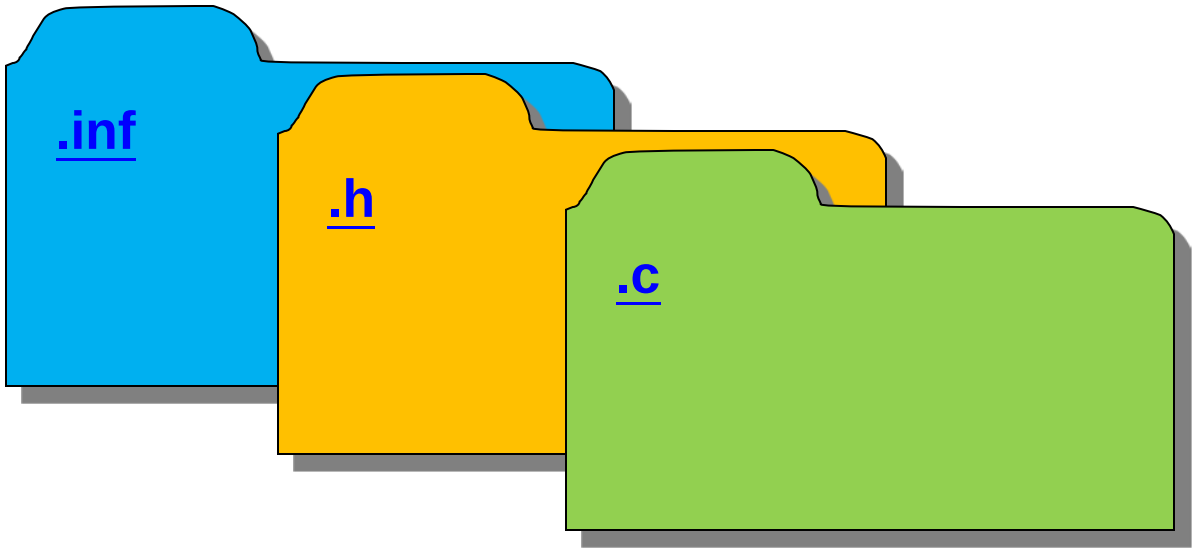
Starts a driver on a controller & Installs Protocols

`Stop()`

Stops a driver from managing a controller

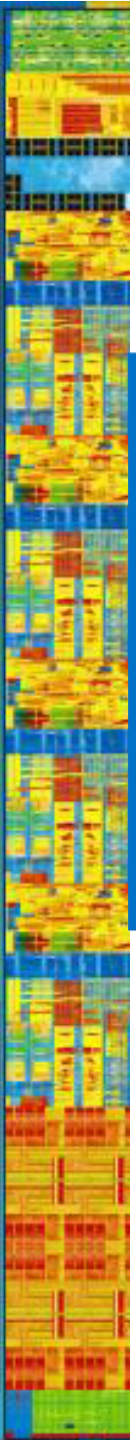
Example of UEFI Driver source code

- C:\FW\edk2\MdeModulePkg\Bus\Scsi\ScsiDiskDxe
  - ScsiDiskDxe.inf
  - ScsiDisk.c
  - ScsiDisk.h

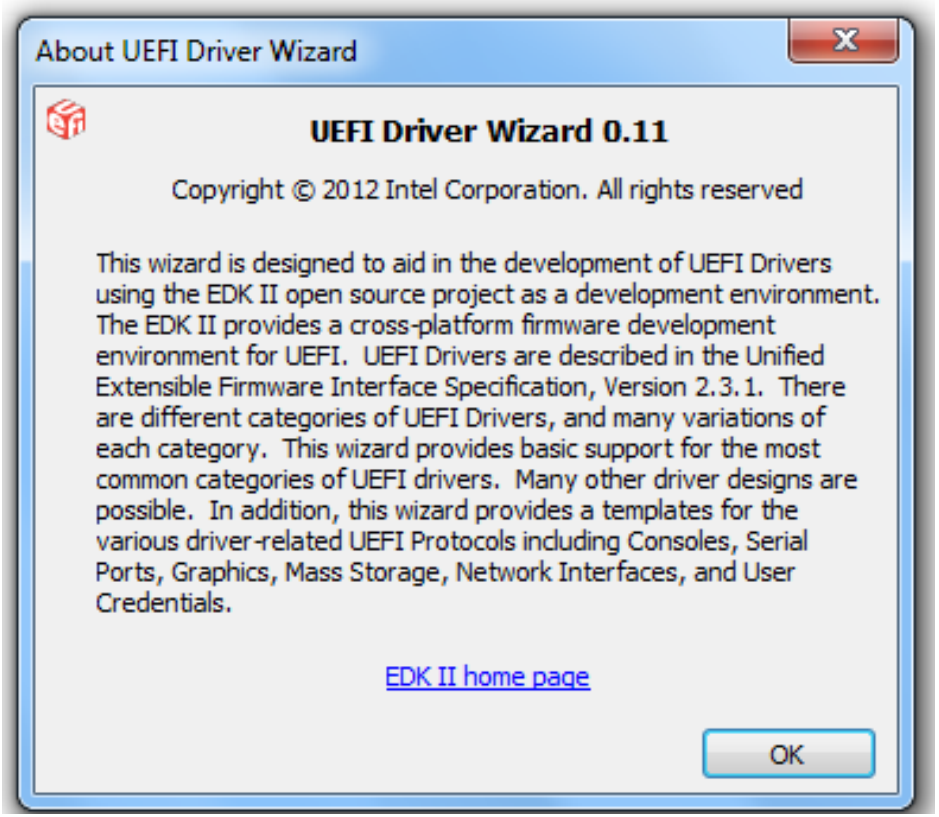


- [.inf] Entry point, Global Protocols
- [.h] Driver's Private Data Structure declaration
- [.c] Review the Supported, Start and Stop functions

Lab 8.1 Creating a UEFI Driver using the UEFI Driver Wizard



UEFI Driver Wizard Overview



- ✓ **Open source tool**
- ✓ Based on *Driver Writer's Guide for UEFI 2.3.1* content
- ✓ **Intel SSG engineers contributed**
- ✓ Located on **www.TianoCore.org**

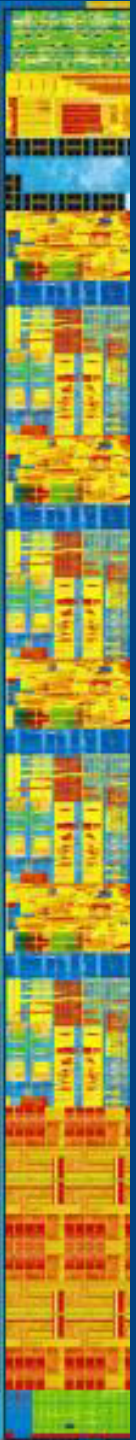
Lab 8.1: Installing the UEFI Driver Wizard

Requirements and Options

Work space must contain BaseTools, MdePkg & MdeModulePkg Packages

UDK2010 IHV release for Driver development on Tianocore.org

Uses previous lab's setup `C:\FW\Edk2`



Driver Functions

- UEFI Device Driver
- UEFI Version 2.3.1 (0x0002001F)
- Unloadable driver
- Support IA32 & x64 CPUs
- Returns component name information
- Test console device
- Option to produce strings & forms for setup

Template File Contents

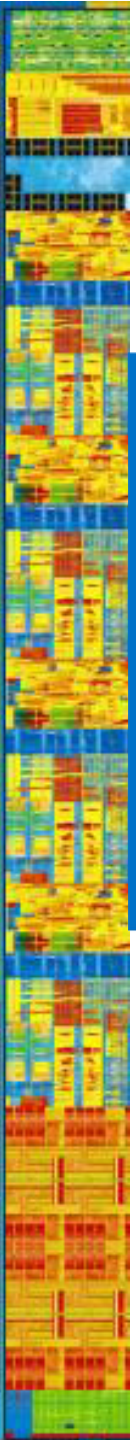
Proper UEFI driver entry point

Basic driver libraries/headers

Skeletons for common driver functions

Error values until ported
EFI_UNSUPPORTED, EFI_DEVICE_ERROR

Lab 8.2 Building a UEFI Driver



Lab 8.3 Editing MyWizardDriver/ComponentName.c

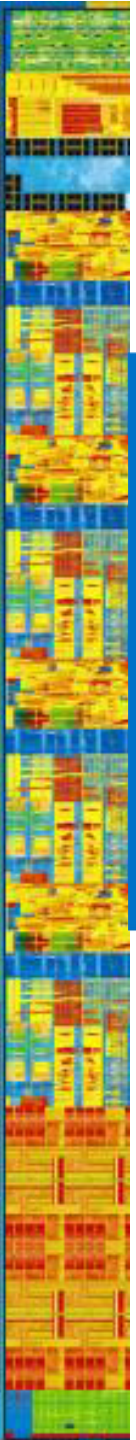
Porting Driver Code

Change the driver's name

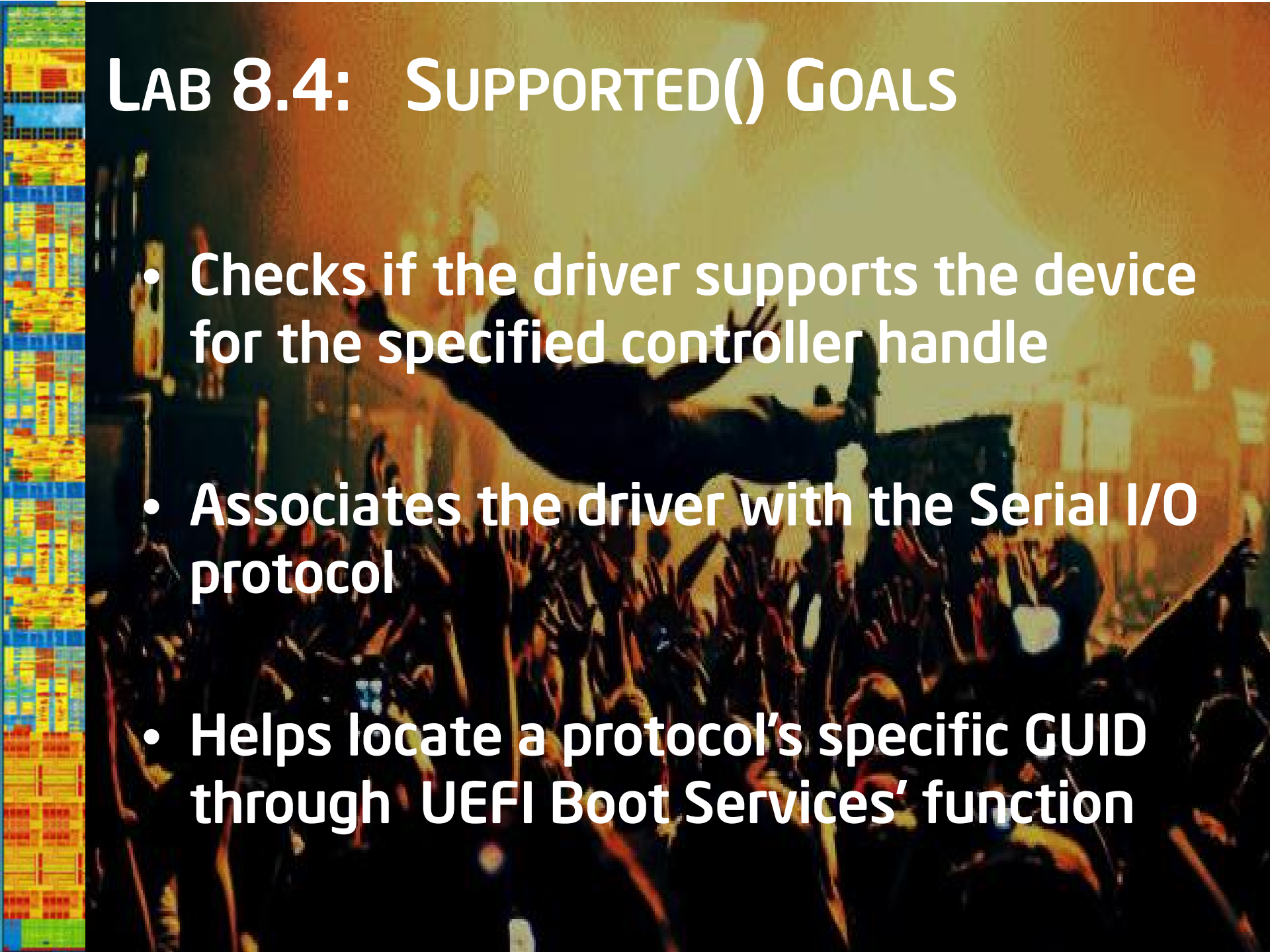
Port `Supported()`

Port `Start()`

LAB 8.4 WRITING THE SUPPORTED FUNCTION FOR A UEFI DRIVER



Lab 8.4: Supported() Goals



- Checks if the driver supports the device for the specified controller handle
- Associates the driver with the Serial I/O protocol
- Helps locate a protocol's specific GUID through UEFI Boot Services' function

START() GOALS

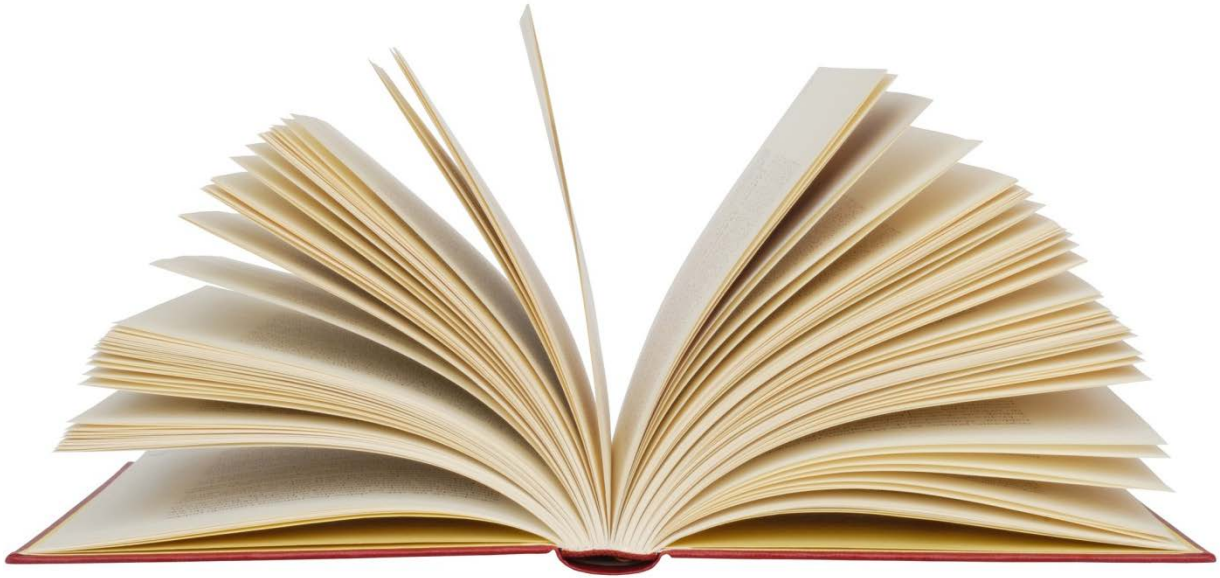
Allocate a string buffer in memory

Fill the memory range with a pattern

ROBUST LIBRARIES

- **AllocateZeroPool()**
  [MemoryAllocationLib.h]

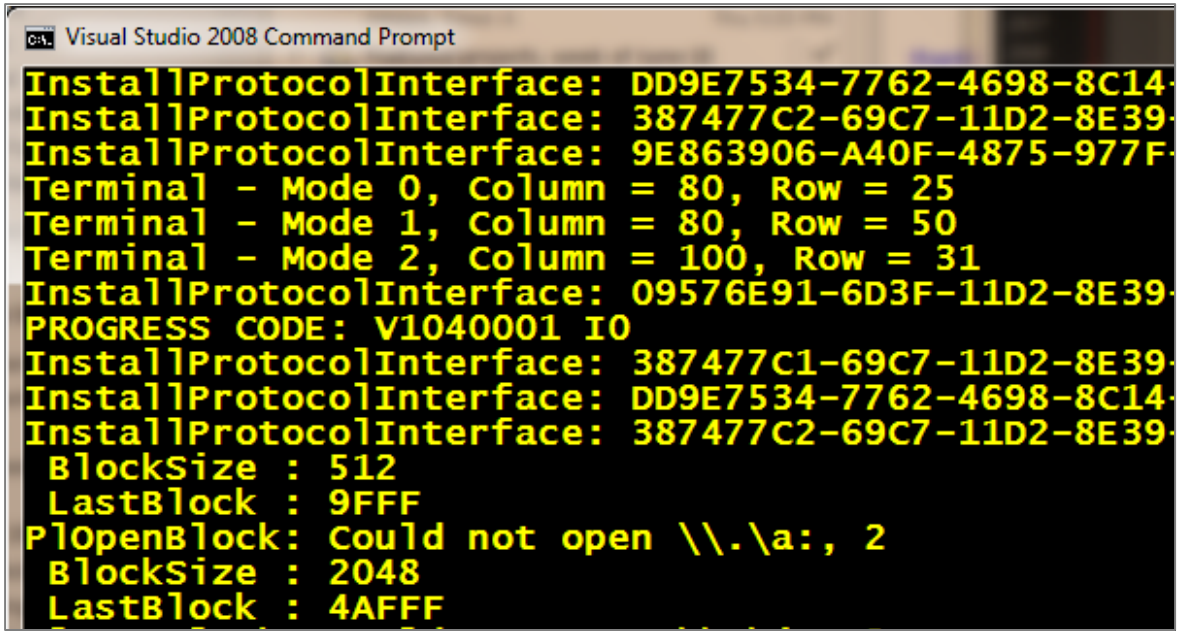
- **SetMem16()**
  [BaseMemoryLib.h]



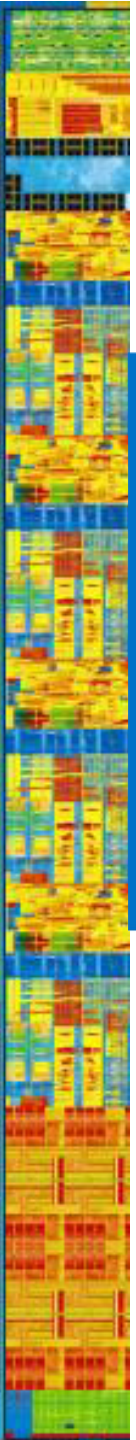
Check the **MdePkg** *help file (.chm format)*

DEBUGGING BEFORE TESTING THE DRIVER

- UEFI drivers can use the EDK II debug library
  - Enables **DEBUG()** **ASSERT()**
- **DEBUG()** statements



Lab 8.5 Returning a Successfully Supported Function



Lab 8.5: Returning a Successful Supported Function

Create Non-Volatile UEFI Variable

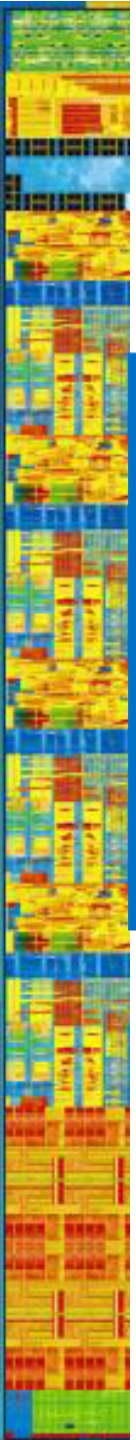
Runtime Services - SetVariable(. . .)

Runtime Services - GetVariable(. . .)

Adding Non-Volitile UEFI Variables

- Create .h file with new `typedef` and GUID
- Include new .h file
- **`EntryPoint()`** Init new buffer for NVRam Variable
- **`Supported()`** Call function to set/get NVRam Variable

Lab 8.5 Porting Unload( ) and Stop( ) functions

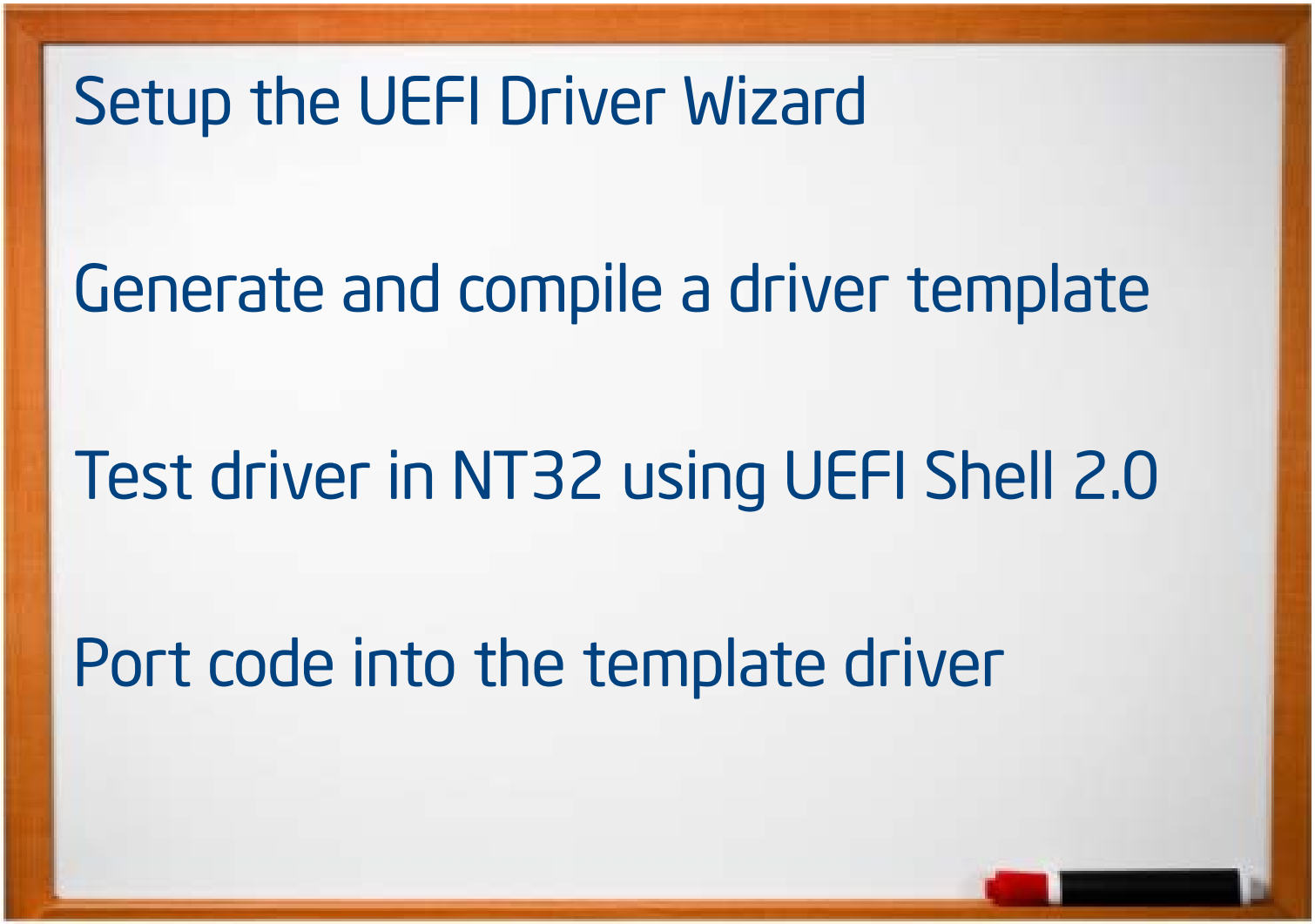


Porting Unload() and Stop() functions

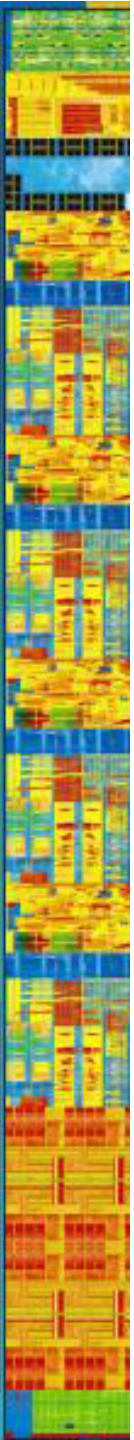
Additional Porting

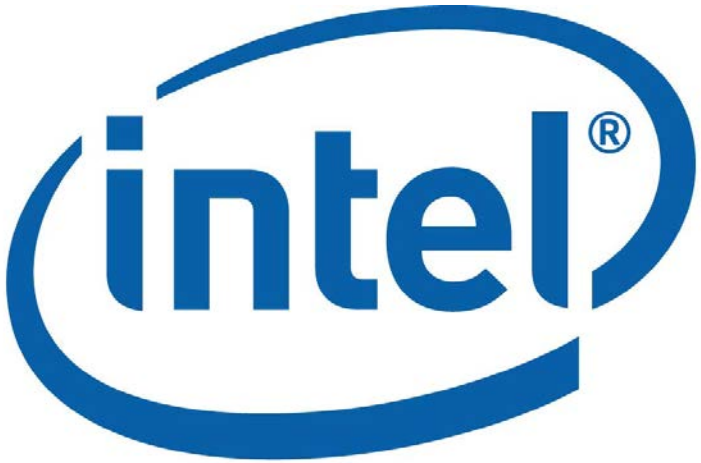
- Adding strings and forms to setup (HII)
- Publish & consume protocols
- Hardware initialization

Lesson 8 Summary

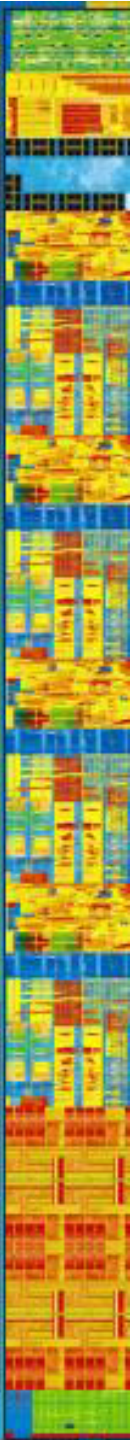



“Any questions?”



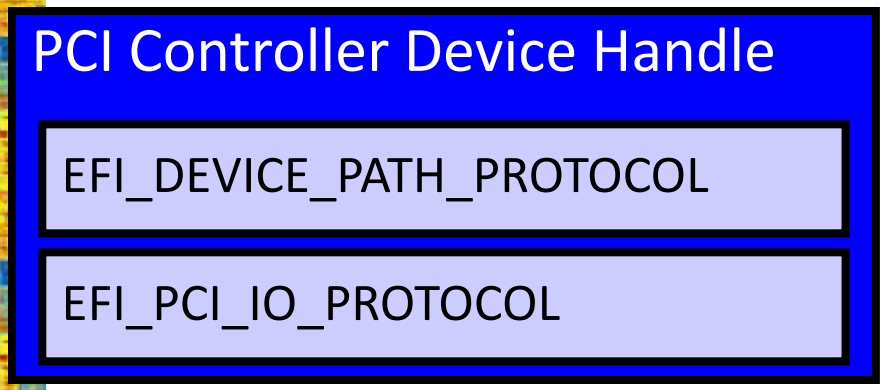


Software



BACKUP

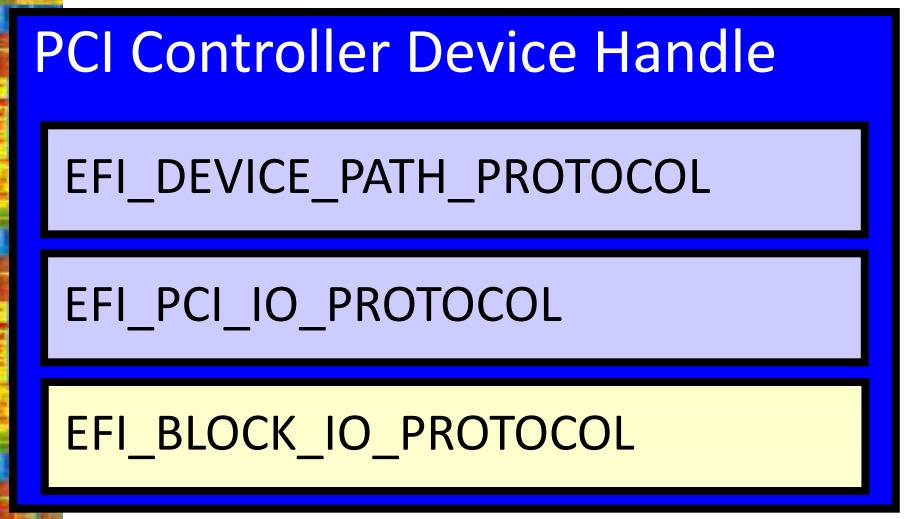
Supported - PCI Controller Device Handle



1. Opens PCI_IO Protocol
2. Checks
3. Closes PCI_IO Protocol
4. Returns: *Supported* or *Not Supported*

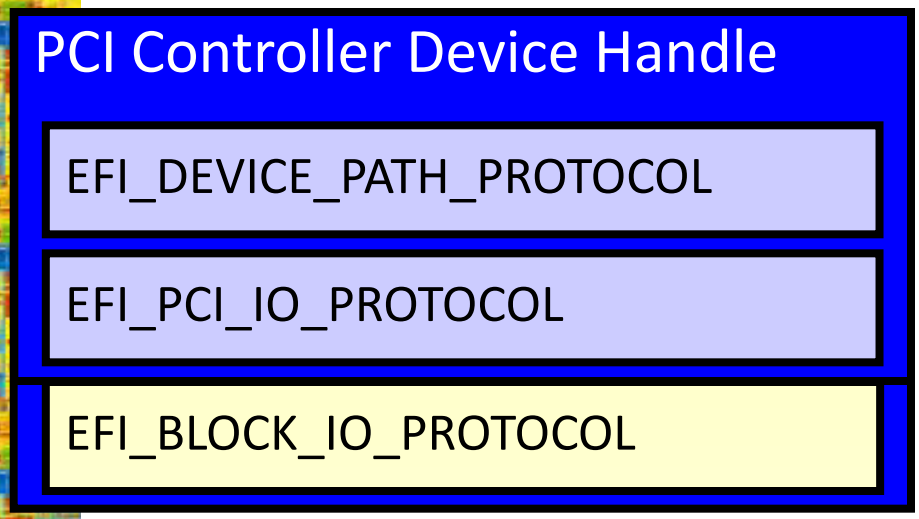
- Inputs:
  - "This"
  - Controller to manage
  - Remaining Device Path
- Supported()
  - Checks to see if a driver supports a controller
  - Check should not change hardware state of controller
  - Minimize execution time, move complex I/O to Start()
  - May be called for controller that is already managed
  - Child is optionally specified

START - PCI CONTROLLER DEVICE HANDLE



- Inputs:
  - "This"
  - Controller to manage,
  - Remaining Device Path
- Start()
  - *Opens* PCI I/O
  - Starts a driver on a controller
  - Can create ALL child handles or ONE child handle

Stop - PCI Controller Device Handle



- Inputs:
  - "This"
  - Controller to manage,
  - Remaining Device Path

- Stop()
  - *Closes* PCI I/O
  - Stops a driver from managing a controller
  - Destroys all specified child handles
  - If no children specified, controller is stopped
  - Stopping a bus controller requires 2 calls
    - One call to stop the children. A second call to stop the bus controller itself

DISCLAIMER

- THIS INFORMATION COTNAINED IN THIS DOCUMENT, INCLUDING ANY TEST RESULTS ARE PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT OR BY THE SALE OF INTEL PRODUCTS. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, or life sustaining applications.

- Intel retains the right to make changes to its specifications at any time, without notice.
- Recipients of this information remain solely responsible for the design, sale and functionality of their products, including any liability arising from product infringement or product warranty.
- Intel may make changes to specifications, product roadmaps and product descriptions at any time, without notice.
- Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

-

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2®, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804