



Secure from the START



Phoenix PSI SDK/DDK Overview

Tim Lewis, Nill Ge

16 March 2007

Agenda

- Phoenix Standard Header Files
- C Library Functions
- EFI Support Functions
- Compiler Support
- Property Setting
- Target Files
- Driver And Application Wizards Overview
- Driver Wizard Demo
- Application Wizard Demo
- Summary



DEMO

Phoenix Standard Header Files

SEC

PEI

DXE

Shell

- `\PSI\LIB\INCLUDE\peilib.h`
 - Standard header for PEIMs.
- `\PSI\LIB\INCLUDE\dxelib.h`
 - Standard header for DXE drivers
- `\PSI\LIB\INCLUDE\uefilib.h`
 - Standard header for UEFI drivers.
- `\PSI\LIB\INCLUDE\applib.h`
 - Standard header for UEFI applications.

C Library Functions

- assert.h – Debug assert functions
- conio.h – Intrinsics for port I/O and interrupt disable
 - _inp, _outp, _disable, _enable, etc.
- ctype.h/wctype.h – Character classification functions
- math.h – Floating point math functions
- stdarg.h – Functions for processing variable argument
- string.h – String handling functions

C Library Functions (contd.)

- `stdio.h` – Standard file read/write functions
 - `printf`, `scanf`, `fopen`, `fread`, etc.
- `stdlib.h` – Miscellaneous functions.
- `time.h` – Standard time handling functions
- `wchar.h` – Wide-character versions of functions.

Easy to port over existing C code to UEFI.

UEFI Support Functions

Device Paths	EfiAppendDevicePath(), EfiAppendDevicePathInstance(), EfiDevicePathToInstance(), EfiDevicePathToHandle(), EfiDuplicateDevicePath(), EfiGetDevicePathSize(), EfiGetNextDevicePathInstance()
Linked Lists	ListGetFirstNode(), ListGetNextNode(), ListInitialize(), ListInsertHead(), ListInsertTail(), ListIsEmpty(), ListIsNodeAtEnd(), ListIsNodeNull(), ListRemoveNode(), ListSwapNodes()
Simple File System	FileSimpleClose(), FileSimpleOpen(), FileSimpleGetDeviceHandle(), FileSimpleGetHandle(), FileSimpleOpenFromPath(), FileSimpleRead(), FileDevicePath()

UEFI Support Functions (contd.)

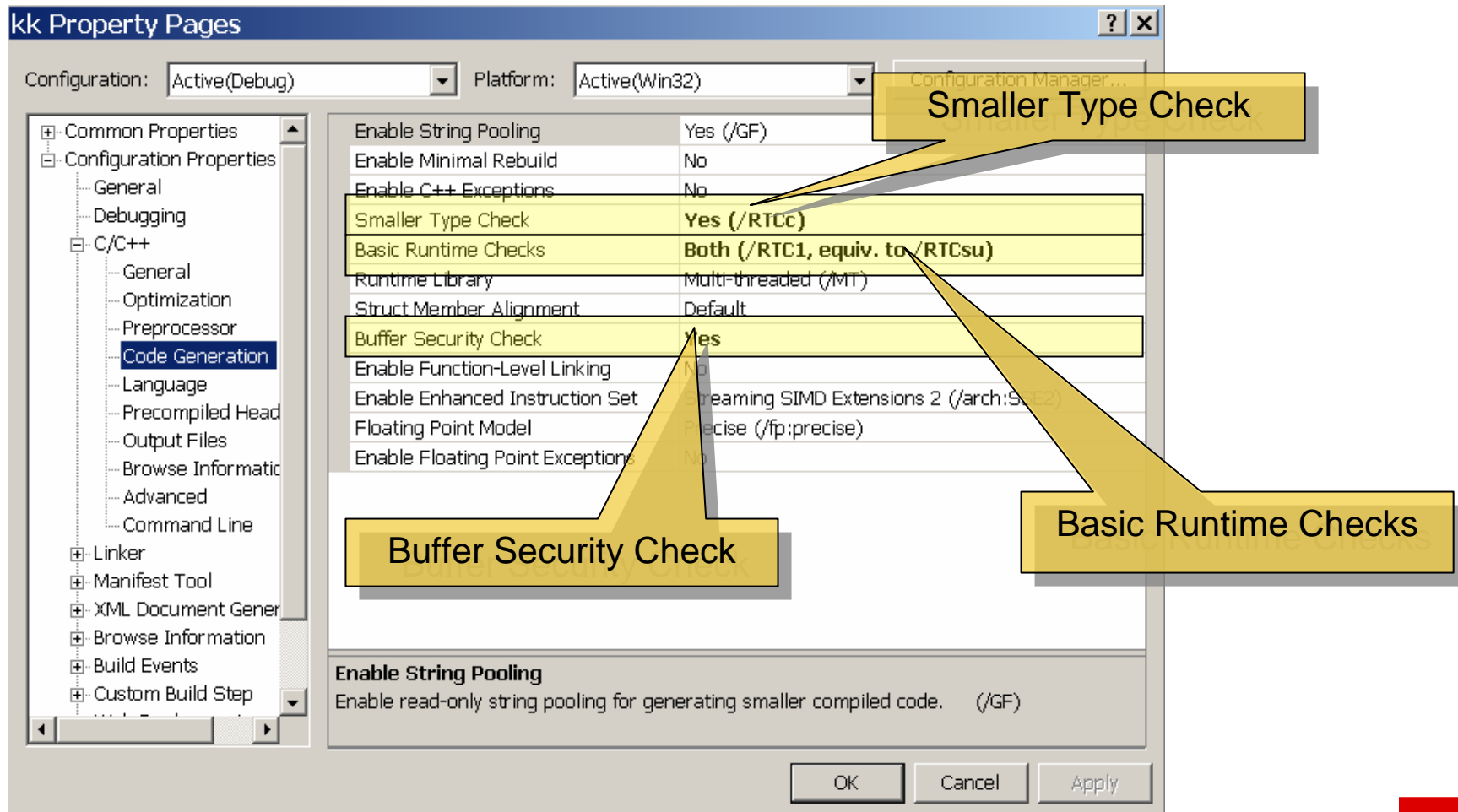
Memory Management	PoolAllocBt(), PoolAllocBtZero(), PoolAllocRt(), PoolDupBt(), PoolDupRt()
CPU	CpuCpuid(), CpuDisableCache(), CpuEnableCache(), CpuGetEflags(), CpuHalt(), CpuInvd(), CpuReadMsr(), CpuReadTsc(), CpuWbinvd(), CpuWriteMsr()
Lock	LockAcquire(), LockAcquireOrFail(), LockInitialize(), LockRelease()
PEI/DXE HOB	PeiBuildHobCpu, PeiBuildHobGuid, GetTimeValue, GetNextGuidHob...

UEFI Support Functions (contd.)

IO/Mem/PCI R/W	IoRead8, MemRead32, PciRead8...
EFI Macros	EFICopyMem, EFISetMem, ,EFIStrCpy, EFILibAllocatePool...
UEFI Global Variables	gST, gBS, gRT, gDS, LoadedImage, GetPeiServices (DDK/SDK will initialize them automatically)
Debug	ASSERT, ASSERT_POINTER, ASSERT_NULL_OR_POINTER DEBUG_ONLY, TRACE, VERIFY, UNUSED_ALWAYS, DbgPrintF, CHECKPOINT

Compiler Support

- Support Of Visual Studio Debug Functions



Compiler Support (contd.)

- Built-In Math Support
- Support For new/delete C++ Operators
- Support Of Visual Studio Performance Enhancement
 - C Compiler Generates Inline Calls To memset.

Tightly integrated with Compiler.

Property Setting

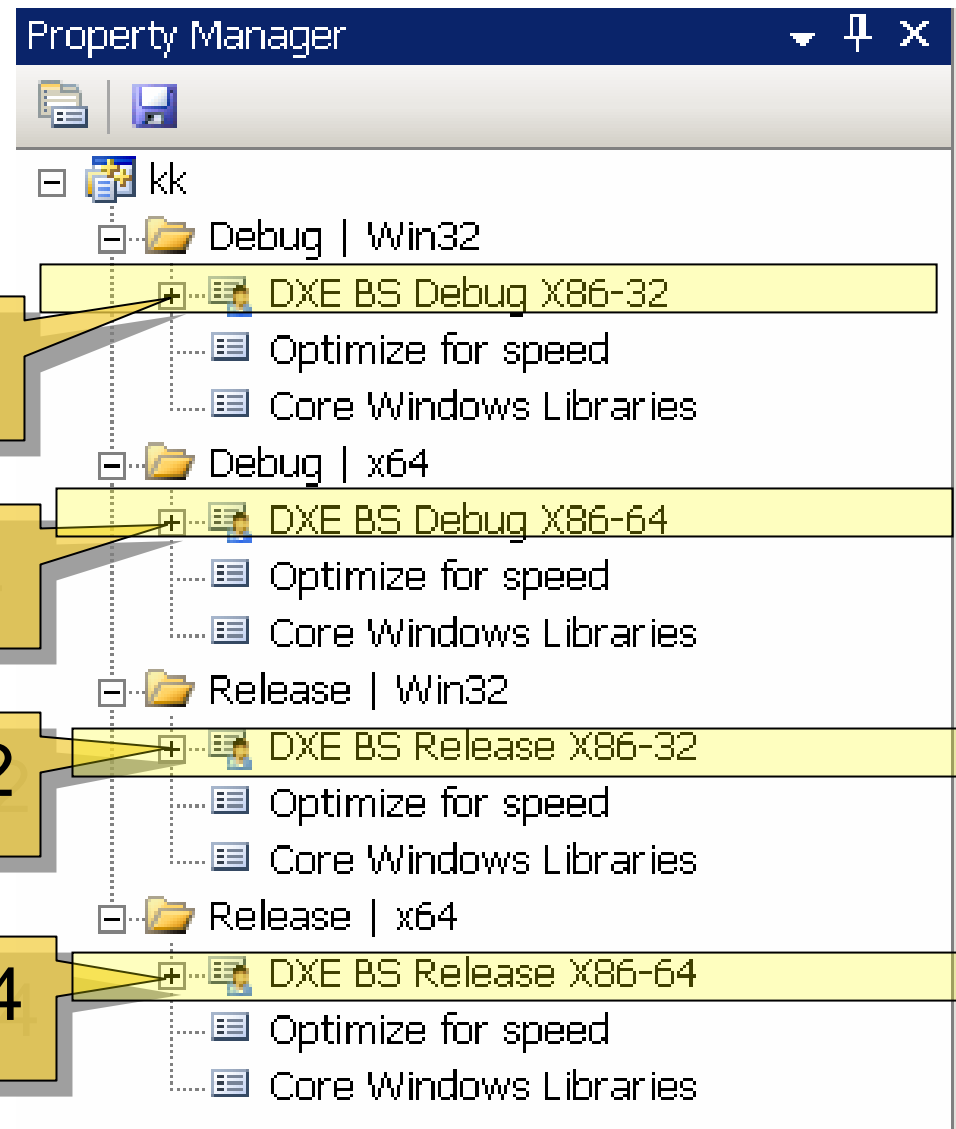
- VSPROPS

DXE BS Debug X86-32

DXE BS Debug X86-64

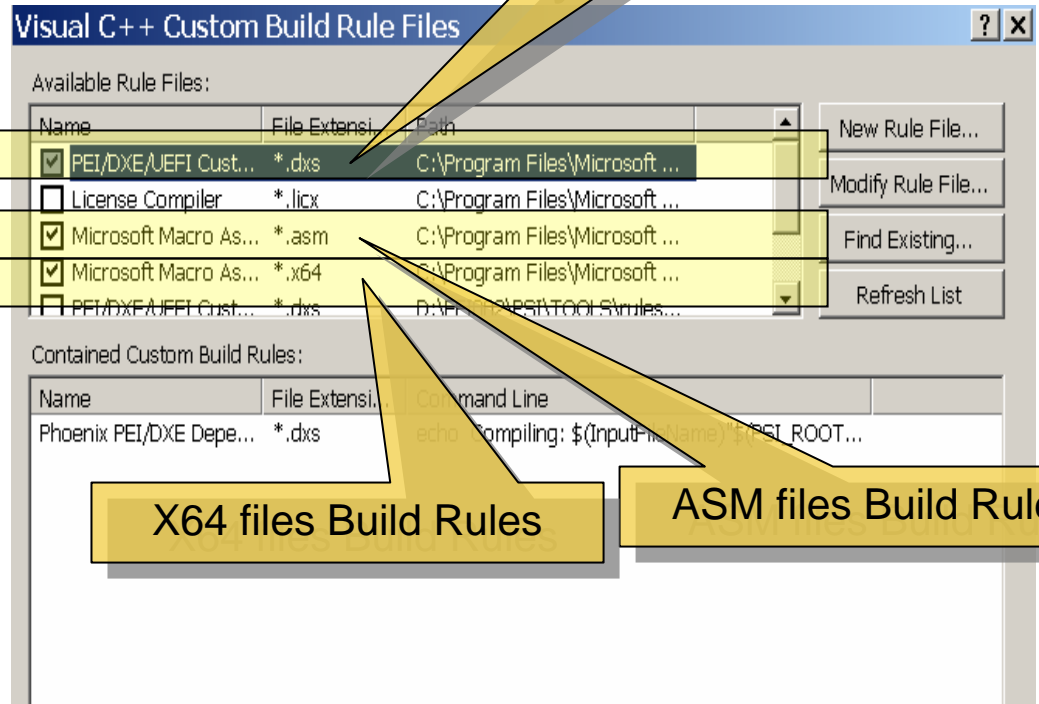
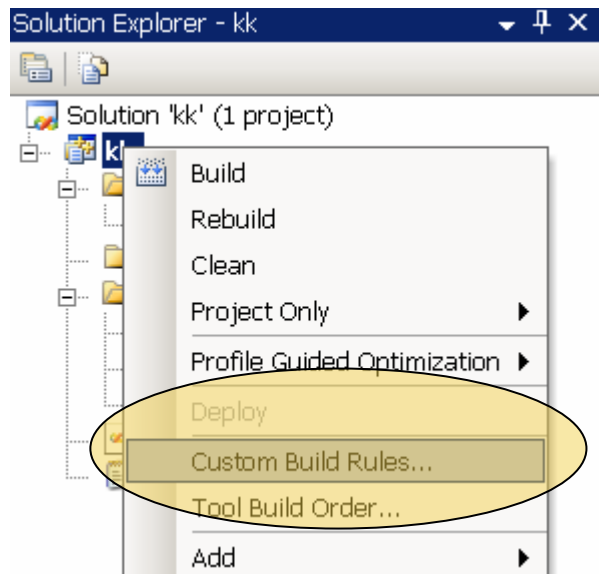
DXE BS Release X86-32

DXE BS Release X86-64



Property Setting (contd.)

- Dependency files
- ASM, X64 files for 32&64 bits assembly code

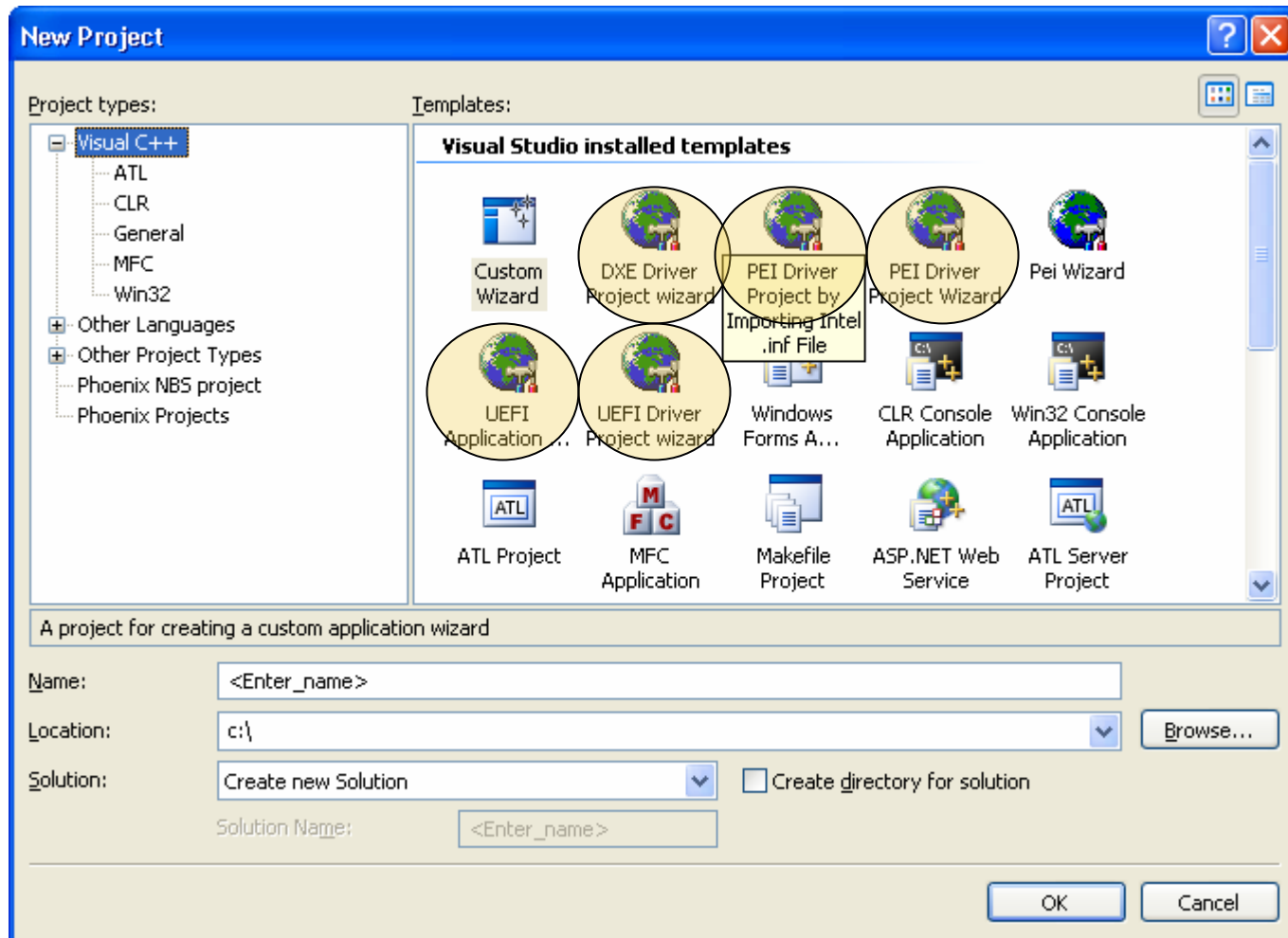


Don't need to specify compilers for them.

Target Files

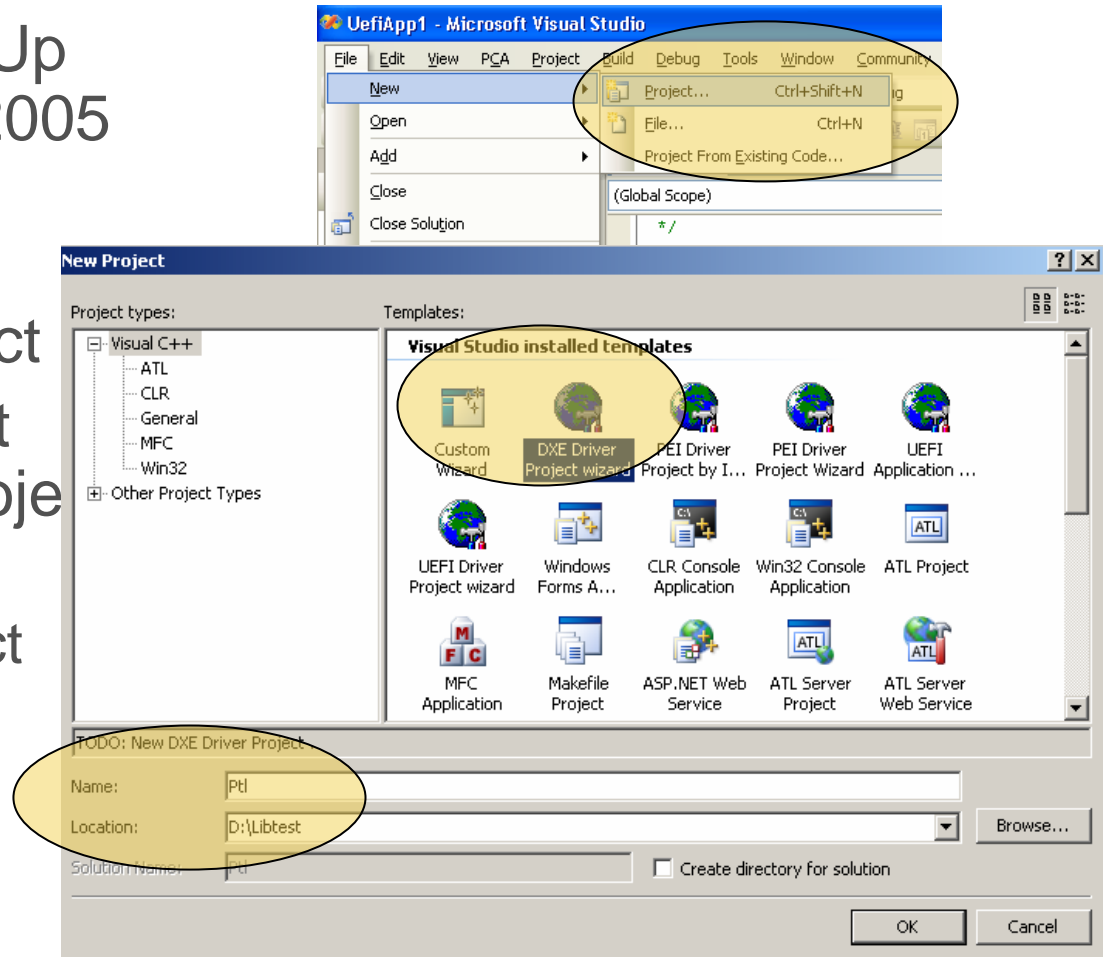
- Phoenix's format drivers and UEFI standard format.
- Debug/Release mode support.
- Merge source into FF files for debug.
- Output PDB files for debug.
- Minimize size for Release files.
- 32-bit and 64-bit(no PEI drivers) support.

Application and Driver Wizards Overview



DXE Wizard Demo

- Step #1: Start Up Visual Studio 2005 SP1
- Step #2: Use File|New|Project
- Step #3: Select DXE Driver Project Wizard
 - Select Project Name and Location
 - Press OK



Driver Wizard Demo (contd.)

- Step #4: Change Project Settings
 - Select DXE Driver Settings
 - Change Driver Name and Copyright
 - Press Finish

DXE Driver Project Wizard - DXEDriver1

Welcome to the DXE Driver Project Wizard

Overview
Driver settings

Please specify the driver settings:

Driver name:
Pti

☐ Empty project

Driver type:
☒ Boot ☐ Runtime

Add these protocols:

☒ ComponentName Protocol2 ☒ Diagnostics Protocol2
☒ DriverBinding Protocol

Copyright:
Phoenix Technologies Ltd

< Previous Next > Finish Cancel

Driver Wizard Demo (contd.)

- Step #5: Develop Your Driver
 - Design the Protocol
 - Create a GUID
 - Finish Protocol's Services
 - Finish all Driver Binding Protocol's Services
 - Set Driver Dependence
 - Add the Instruction into readme.txt

Driver Binding

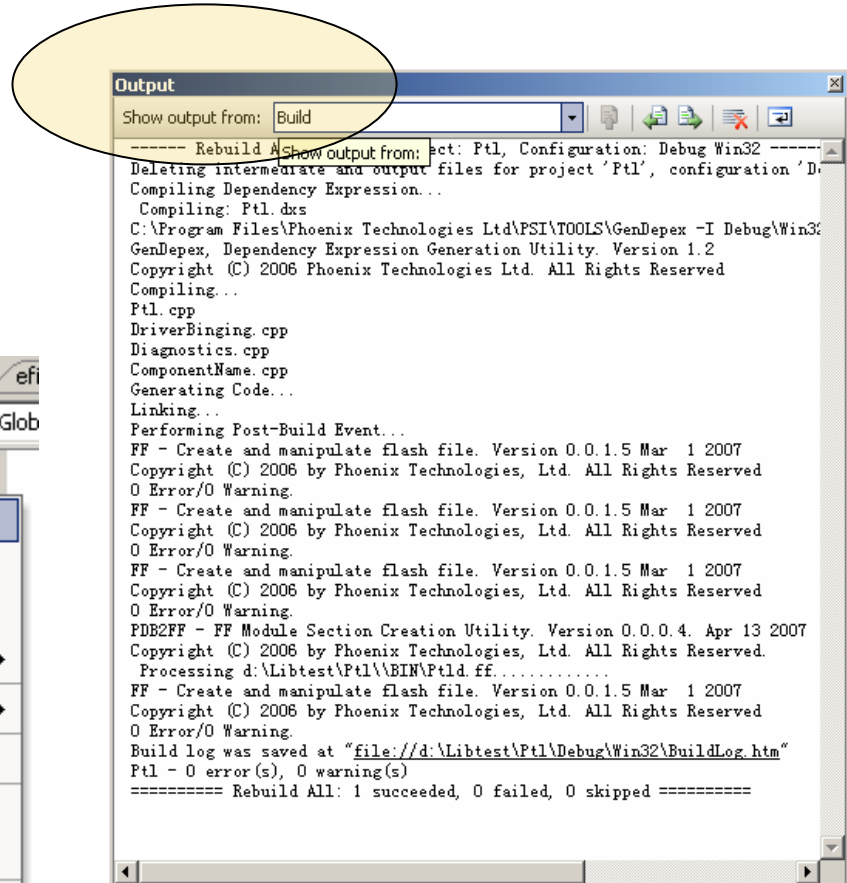
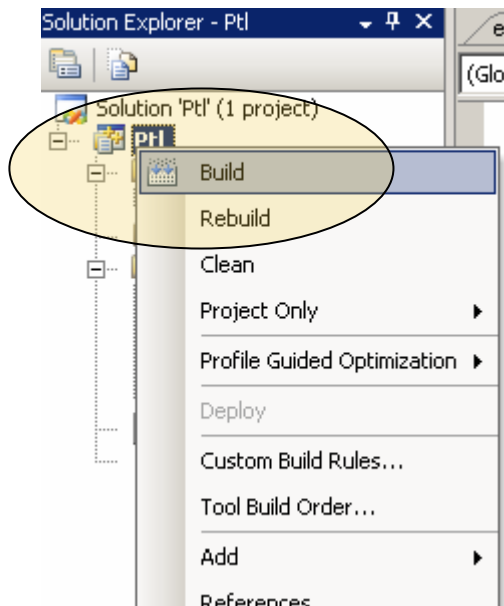
Component Name2

Diagnostics

```
20 Function:
21 DxeMain
22
23 Description:
24 DXE driver's entry point.
25
26 Parameters In:
27 IN EFI_HANDLE ImageHandle,
28 IN EFI_SYSTEM_TABLE *SystemTable
29
30 Return Value:
31 EFI_STATUS
32
33 *****/
34
35 EXTERN_C
36 EFI_STATUS
37 EFIAPP
38 DxeMain(
39     IN EFI_HANDLE ImageHandle,
40     IN EFI_SYSTEM_TABLE *SystemTable
41 )
42 {
43     EFI_STATUS Status = EFI_SUCCESS;
44
45     // install driver binding protocol
46     gPtlDriverBinding.ImageHandle = ImageHandle;
47     gPtlDriverBinding.DriverBindingHandle = ImageHandle;
48
49     Status = gBS->InstallProtocolInterface (
50         &ImageHandle,
51         &gEfiDriverBindingProtocolGuid,
52         EFI_NATIVE_INTERFACE,
53         &gPtlDriverBinding
54     );
55     if (EFI_ERROR (Status)) {
56         return Status;
57     }
58
59     // install Component name2 protocol
60     Status = gBS->InstallProtocolInterface (
61         &ImageHandle,
62         &gEfiComponentName2ProtocolGuid,
63         EFI_NATIVE_INTERFACE,
64         &gPtlComponentName
65     );
66     if (EFI_ERROR (Status)) {
67         return Status;
68     }
69
70     // install driver Diagnostics protocol
71     Status = gBS->InstallProtocolInterface (
72         &ImageHandle,
73         &gEfiDriverDiagnosticsProtocolGuid,
74         EFI_NATIVE_INTERFACE,
75         &gPtlDriverDiagnostics
76     );
77     if (EFI_ERROR (Status)) {
78         return Status;
79     }
80 }
```

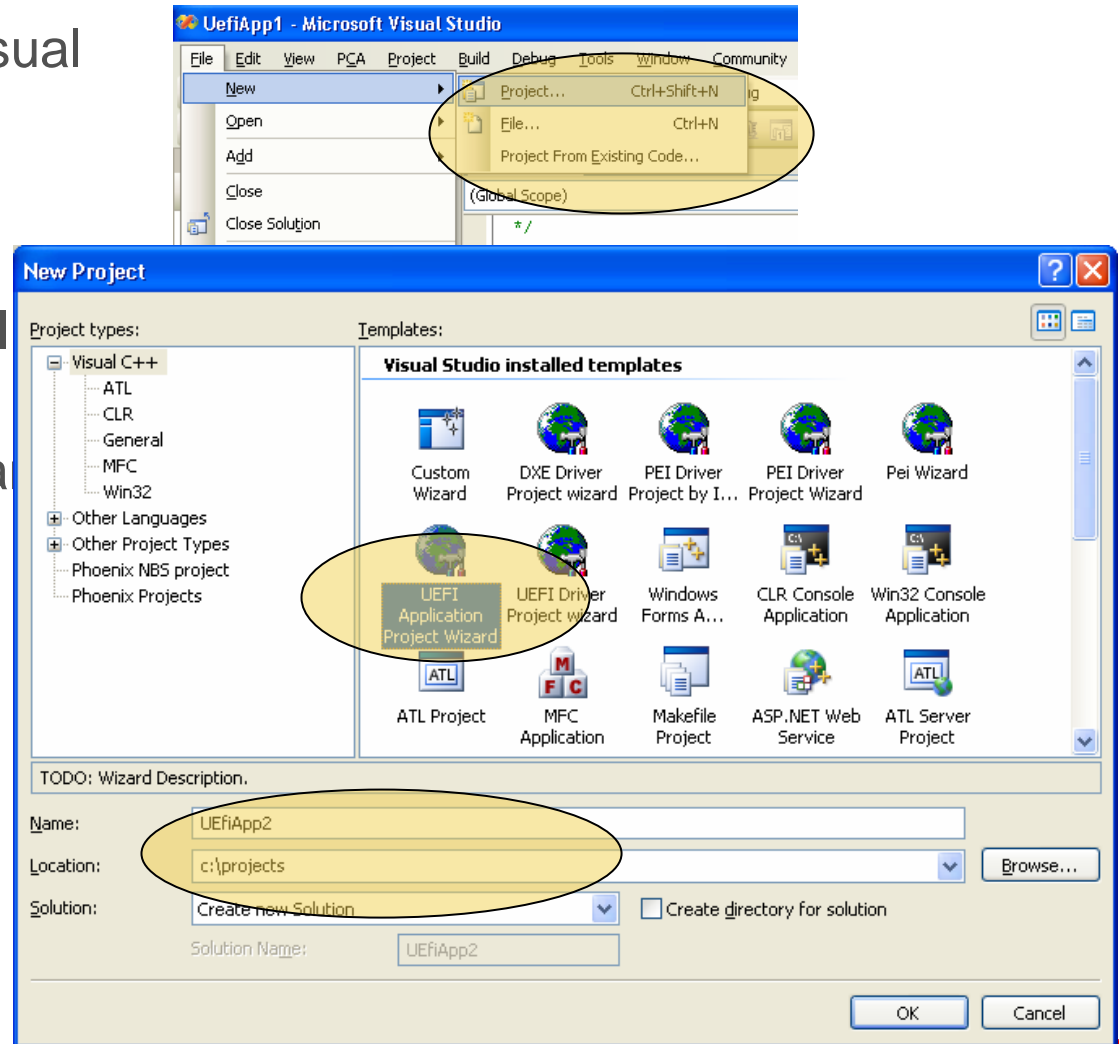
Driver Wizard Demo (contd.)

- Step #6: Build The Driver Using Build|Build
- Step #7: Check Target Files



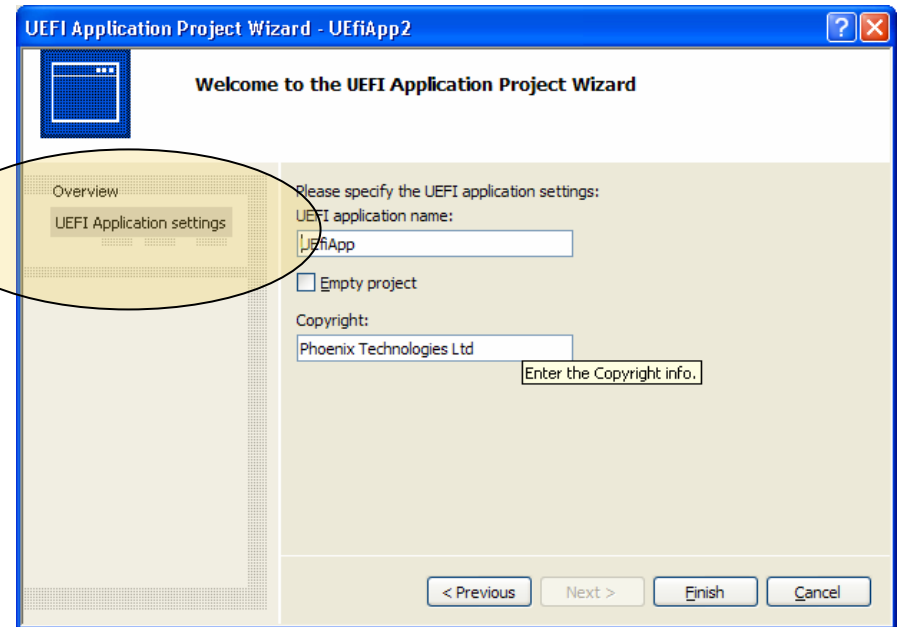
“Hello World” Application Demo

- Step #1: Start Up Visual Studio 2005 SP1
- Step #2: Use File|New|Project
- Step #3: Select UEFI Application Wizard
 - Select Project Name and Location
 - Press OK



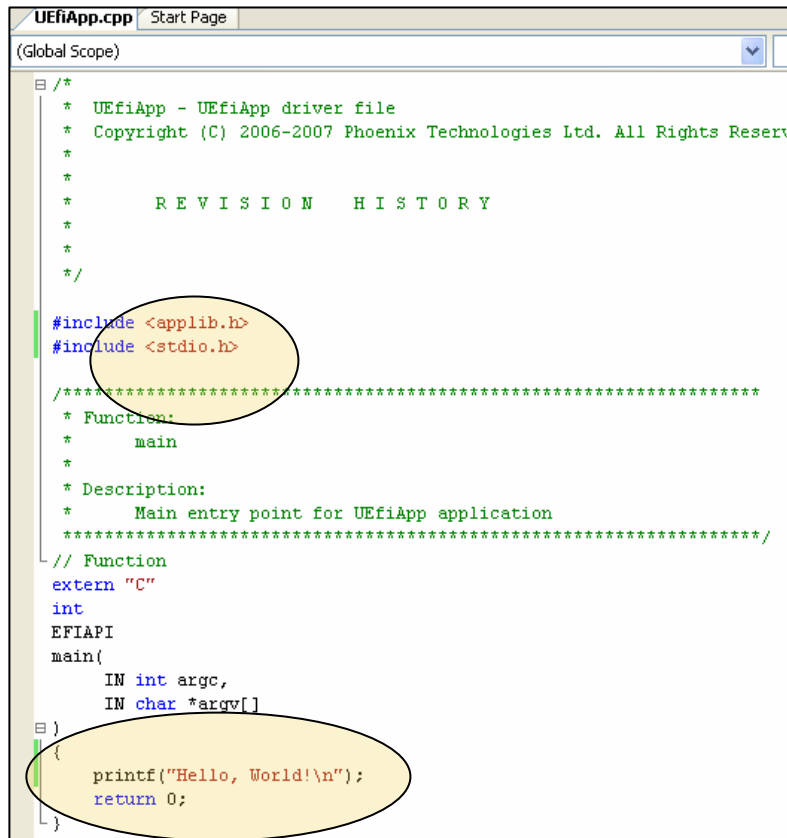
“Hello World” Application Demo (contd.)

- Step #4: Change Project Settings
 - Select UEFI Application Settings
 - Change Application Name and Copyright
 - Press Finish



“Hello World” Application Demo (contd.)

- Step #5: Add Include Files and printf in UefiApp.cpp



The screenshot shows the UefiApp.cpp file in a code editor. Two yellow circles highlight specific code sections. The first circle highlights the include statements: `#include <aplib.h>` and `#include <stdio.h>`. The second circle highlights the printf statement and the return statement: `printf("Hello, World!\n");` and `return 0;`.

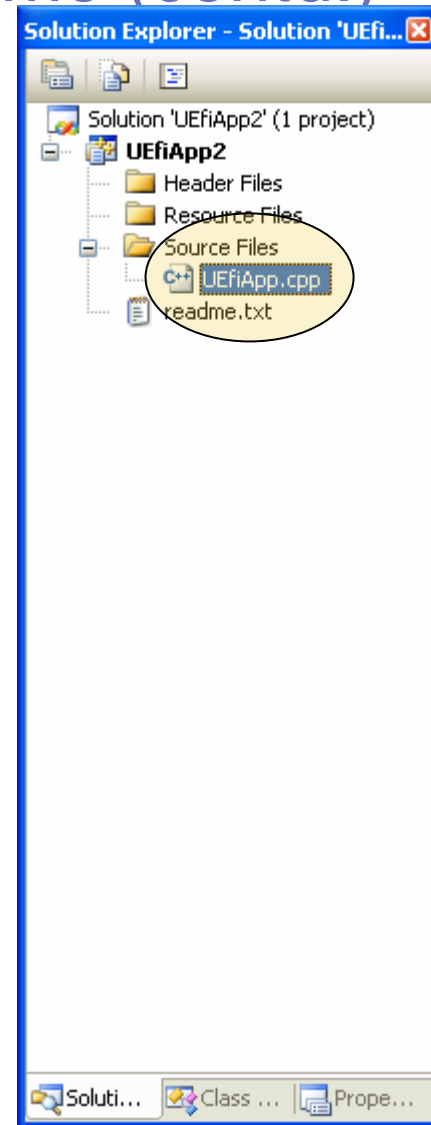
```
UefiApp.cpp | Start Page
(Global Scope)

/*
 * UefiApp - UefiApp driver file
 * Copyright (C) 2006-2007 Phoenix Technologies Ltd. All Rights Reserved
 *
 *      R E V I S I O N   H I S T O R Y
 *
 */

#include <aplib.h>
#include <stdio.h>

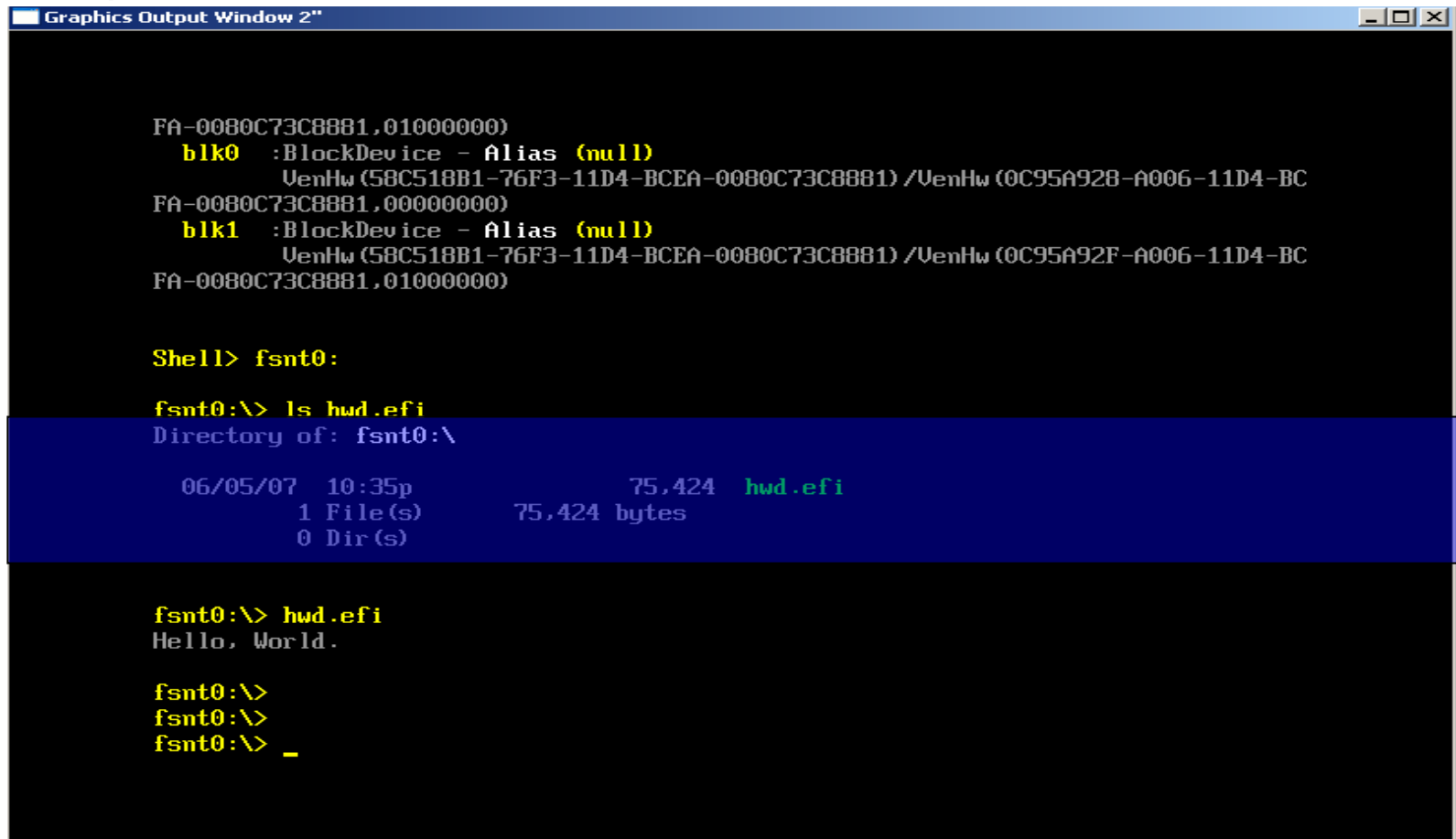
/*****
 * Function:
 *   main
 *
 * Description:
 *   Main entry point for UefiApp application
 *****/

// Function
extern "C"
int
EFIAPI
main(
    IN int argc,
    IN char *argv[]
)
{
    printf("Hello, World!\n");
    return 0;
}
```



“Hello World” Application Demo (contd.)

- Step #6: Run It in EDK Simulator

A screenshot of a window titled "Graphics Output Window 2" from the EDK Simulator. The window has a black background with yellow and white text. It shows the initialization of two block devices, blk0 and blk1, with their respective GUIDs and UEFI paths. Below this, a shell prompt "Shell> fsnt0:" is shown. The user enters "fsnt0:\> ls hwd.efi", and the output shows a directory listing for "fsnt0:\". The listing includes a timestamp "06/05/07 10:35p", a file size of "75,424", and the filename "hwd.efi". Below the listing, the user enters "fsnt0:\> hwd.efi", and the output is "Hello, World.". The user then enters "fsnt0:\>" and "fsnt0:\> _" on subsequent lines.

```
Graphics Output Window 2"

FA-0080C73C8881,01000000)
  blk0 :BlockDevice - Alias (null)
        UenHw (58C518B1-76F3-11D4-BCEA-0080C73C8881) /UenHw (0C95A928-A006-11D4-BC
FA-0080C73C8881,00000000)
  blk1 :BlockDevice - Alias (null)
        UenHw (58C518B1-76F3-11D4-BCEA-0080C73C8881) /UenHw (0C95A92F-A006-11D4-BC
FA-0080C73C8881,01000000)

Shell> fsnt0:

fsnt0:\> ls hwd.efi
Directory of: fsnt0:\

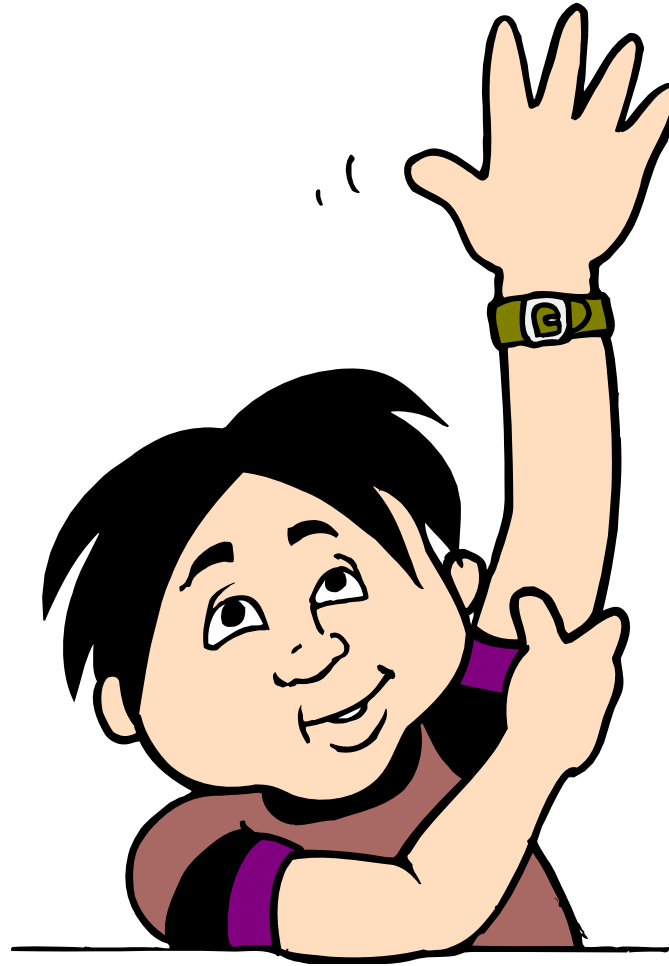
    06/05/07  10:35p                75,424  hwd.efi
              1 File(s)            75,424 bytes
              0 Dir(s)

fsnt0:\> hwd.efi
Hello, World.

fsnt0:\>
fsnt0:\>
fsnt0:\> _
```

Summary

- VS 2005 environment, it's easy for engineers.
- Support standard C functions.
- Tightly integrated Compiler Support.
- Easy for debug.
- Driver wizards provide great starting point.
- Special library support for UEFI.
- Phoenix's BIOS Developer Blog
 - <http://blogs.phoenix.com>



Questions