tianocore

UEFI & EDK II Training

EDK II Debugging with Windows Lab

tianocore.org

LESSON OBJECTIVE

- Define `DebugLib` and its attributes
- List the ways to debug
- Using PCDs to Configure `DebugLib` - LAB
- Change Compiler & Linker Flags for debugging
- Change the `DebugLib` instance to modify the debug output - LAB
- Debug EDK II using VS Debugger - LAB

DEBUGGING OVERVIEW

Debug Methods

DEBUG and ASSERT macros in EDK II code

DEBUG instead of Print functions

Software/hardware debuggers

Shell commands to test capabilities for simple debugging

EDK II DebugLib Library

Debug and **Assert** macros in code

Enable/disable when compiled (target.txt)

Connects a Host to capture debug messages

tianocore

DEBUGGING WITH PCDS

Using PCDs to Configure DebugLib

MdePkg Debug Library Class

```
[PcdsFixedAtBuild. PcdsPatchableInModule]
    . . .
  gEfiMdePkgTokenSpaceGuid.PcdDebugPropertyMask|0x1f
  gEfiMdePkgTokenSpaceGuid.PcdDebugPrintErrorLevel|0x80000040
```

PCDs Set which drivers report errors and change what messages get printed

PcdDebugPropertyMask Values

Debugging *Features* Enabled

```
#define DEBUG_PROPERTY_DEBUG_ASSERT_ENABLED       0x01
#define DEBUG_PROPERTY_DEBUG_PRINT_ENABLED        0x02
#define DEBUG_PROPERTY_DEBUG_CODE_ENABLED         0x04
#define DEBUG_PROPERTY_CLEAR_MEMORY_ENABLED       0x08
#define DEBUG_PROPERTY_ASSERT_BREAKPOINT_ENABLED  0x10
#define DEBUG_PROPERTY_ASSERT_DEADLOOP_ENABLED    0x20
```

Default value in `OvmfPkg` is `0x2f`

Default value in `EmulatorPkg` is `0x1f`

Determines which debugging features are enabled

PcdDebugPrintErrorLevel Values

Debug *Messages* Displayed

```
#define DEBUG_INIT      0x00000001  // Initialization
#define DEBUG_WARN      0x00000002  // Warnings
#define DEBUG_LOAD      0x00000004  // Load events
#define DEBUG_FS        0x00000008  // EFI File system
#define DEBUG_POOL      0x00000010  // Alloc & Free's  Pool
#define DEBUG_PAGE      0x00000020  // Alloc & Free's  Page
#define DEBUG_INFO      0x00000040  // Verbose
#define DEBUG_DISPATCH  0x00000080  // PEI/DXE Dispatchers
#define DEBUG_VARIABLE  0x00000100  // Variable
#define DEBUG_BM        0x00000400  // Boot Manager
#define DEBUG_BLKIO     0x00001000  // BlkIo Driver
#define DEBUG_NET       0x00004000  // SNP / Network Io Driver
#define DEBUG_UNDI      0x00010000  // UNDI Driver
#define DEBUG_LOADFILE  0x00020000  // Load File
#define DEBUG_EVENT     0x00080000  // Event messages
#define DEBUG_GCD       0x00100000  // Global Coherency Database changes
#define DEBUG_CACHE     0x00200000  // Memory range cache-ability changes
#define DEBUG_VERBOSE   0x00400000  // Detailed debug messages that may
                                    // significantly impact boot performance
#define DEBUG_ERROR     0x80000000  // Error
```

Determines which messages we want to print

Changing PCD Values

Change all instances of a PCD in platform DSC

```
[PcdsFixedAtBuild.IA32]
gEfiMdePkgTokenSpaceGuid.PcdDebugPrintErrorLevel|0x00000000
```

Change a single module's PCD values in DSC

```
MyPath/MyModule.inf {
<PcdsFixedAtBuild>
gEfiMdePkgTokenSpaceGuid.PcdDebugPrintErrorLevel|0x80000000
}
```

Minimize message output and minimize size increase

Other Debug Related Libraries

ReportStatusCodeLib – Progress codes

`gEfiMdePkgTokenSpaceGuid.PcdReportStatusCodePropertyMask`

PostCodeLib – Enable Post codes

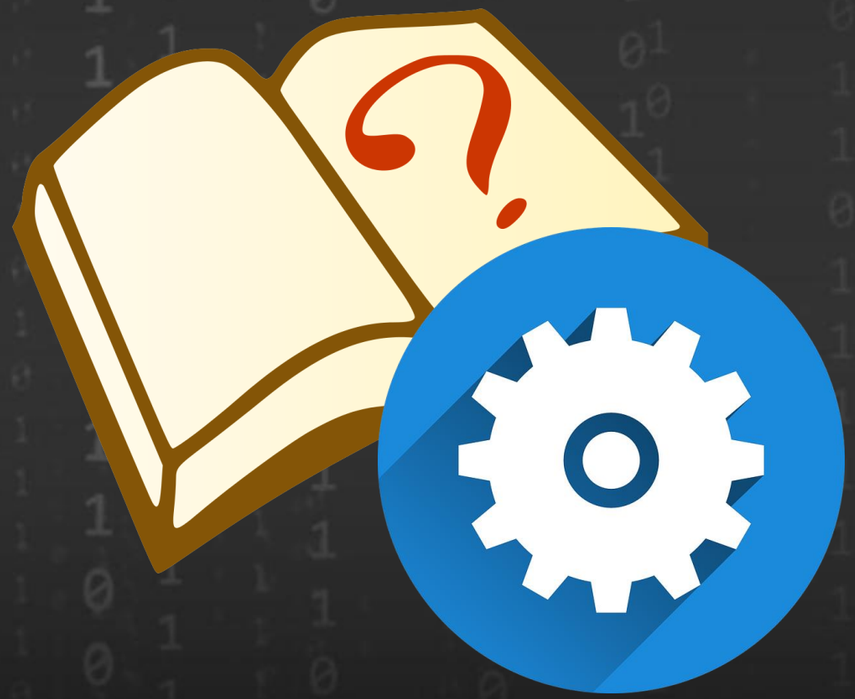
`gEfiMdePkgTokenSpaceGuid.PcdPostCodePropertyMask`

PerformanceLib – Enable Measurement

`gEfiMdePkgTokenSpaceGuid.PcdPerformanceLibraryPropertyMask`

Lab 1 – Adding Debug Statements

In this lab, you'll add debug statements to the previous lab's SampleApp UEFI Shell application



Lab 1: Catch up from previous lab

Skip if Lab Writing UEFI App Lab completed

- Perform Lab Setup from previous Labs
- Create a Directory under the workspace `C:/FW/edk2-ws/edk2` “`SampleApp`”
- Copy contents of `C:../FW/LabSampleCode/SampleAppDebug` to `C:/FW/edk2-ws/edk2/SampleApp`
- Open `C:/FW/edk2/EmulatorPkg/EmulatorPkg.dsc`
- Add the following to the [Components] section:

```
# Add new modules here
 SampleApp/SampleApp.inf
```

- Save and close the file EmulatorPkg.dsc

Lab 1: Add debug statements to SampleApp

- Open a VS Command Prompt and type: `cd C:/FW/edk2-ws` then

```
C:/FW/edk2-ws > setenv.bat
C:/FW/edk2-ws > cd edk2
C:/FW/edk2-ws/edk2 > edksetup
```

- Open `C:/FW/edk2-ws/edk2/SampleApp/SampleApp.c`

- Add the following to the include statements at the top of the file after below the last “`include`” statement:

```
#include <Library/DebugLib.h>
```

Lab 1: Add debug statements to SampleApp

Locate the UefiMain function. Then copy and paste the following code after the "`EFI_INPUT_KEY  KEY;`" statement: and before the first `Print()` statement as shown in the screen shot below:

```
DEBUG ((0xffffffff, "\n\nUEFI Base Training DEBUG DEMO\n") );
DEBUG ((0xffffffff, "0xffffffff USING DEBUG ALL Mask Bits Set\n") );

DEBUG ((DEBUG_INIT,     " 0x%08x USING DEBUG DEBUG_INIT\n" , (UINTN)(DEBUG_INIT))  );
DEBUG ((DEBUG_WARN,     " 0x%08x USING DEBUG DEBUG_WARN\n", (UINTN)(DEBUG_WARN))  );
DEBUG ((DEBUG_LOAD,     " 0x%08x USING DEBUG DEBUG_LOAD\n", (UINTN)(DEBUG_LOAD))  );
DEBUG ((DEBUG_FS,       " 0x%08x USING DEBUG DEBUG_FS\n", (UINTN)(DEBUG_FS))  );
DEBUG ((DEBUG_POOL,     " 0x%08x USING DEBUG DEBUG_POOL\n", (UINTN)(DEBUG_POOL))  );
DEBUG ((DEBUG_PAGE,     " 0x%08x USING DEBUG DEBUG_PAGE\n", (UINTN)(DEBUG_PAGE))  );
DEBUG ((DEBUG_INFO,     " 0x%08x USING DEBUG DEBUG_INFO\n", (UINTN)(DEBUG_INFO))  );
DEBUG ((DEBUG_DISPATCH, " 0x%08x USING DEBUG DEBUG_DISPATCH\n",(UINTN)(DEBUG_DISPATCH)));
DEBUG ((DEBUG_VARIABLE, " 0x%08x USING DEBUG DEBUG_VARIABLE\n",(UINTN)(DEBUG_VARIABLE)));
DEBUG ((DEBUG_BM,       " 0x%08x USING DEBUG DEBUG_BM\n", (UINTN)(DEBUG_BM))  );
DEBUG ((DEBUG_BLKIO,    " 0x%08x USING DEBUG DEBUG_BLKIO\n", (UINTN)(DEBUG_BLKIO))  );
DEBUG ((DEBUG_NET,      " 0x%08x USING DEBUG DEBUG_NET\n", (UINTN)(DEBUG_NET))  );
DEBUG ((DEBUG_UNDI,     " 0x%08x USING DEBUG DEBUG_UNDI\n", (UINTN)(DEBUG_UNDI))  );
DEBUG ((DEBUG_LOADFILE, " 0x%08x USING DEBUG DEBUG_LOADFILE\n",(UINTN)(DEBUG_LOADFILE)));
DEBUG ((DEBUG_EVENT,    " 0x%08x USING DEBUG DEBUG_EVENT\n", (UINTN)(DEBUG_EVENT))  );
DEBUG ((DEBUG_GCD,      " 0x%08x USING DEBUG DEBUG_GCD\n", (UINTN)(DEBUG_EVENT)) );
DEBUG ((DEBUG_CACHE,    " 0x%08x USING DEBUG DEBUG_CACHE\n", (UINTN)(DEBUG_CACHE)) );
DEBUG ((DEBUG_VERBOSE,  " 0x%08x USING DEBUG DEBUG_VERBOSE\n", (UINTN)(DEBUG_VERBOSE)) );
DEBUG ((DEBUG_ERROR,    " 0x%08x USING DEBUG DEBUG_ERROR\n", (UINTN)(DEBUG_ERROR))  );
```

Lab 1: Build, Run and Test Result

At the VS Command Prompt

```
$> Build
$> RunEmulator.bat
```

Run the application from the shell

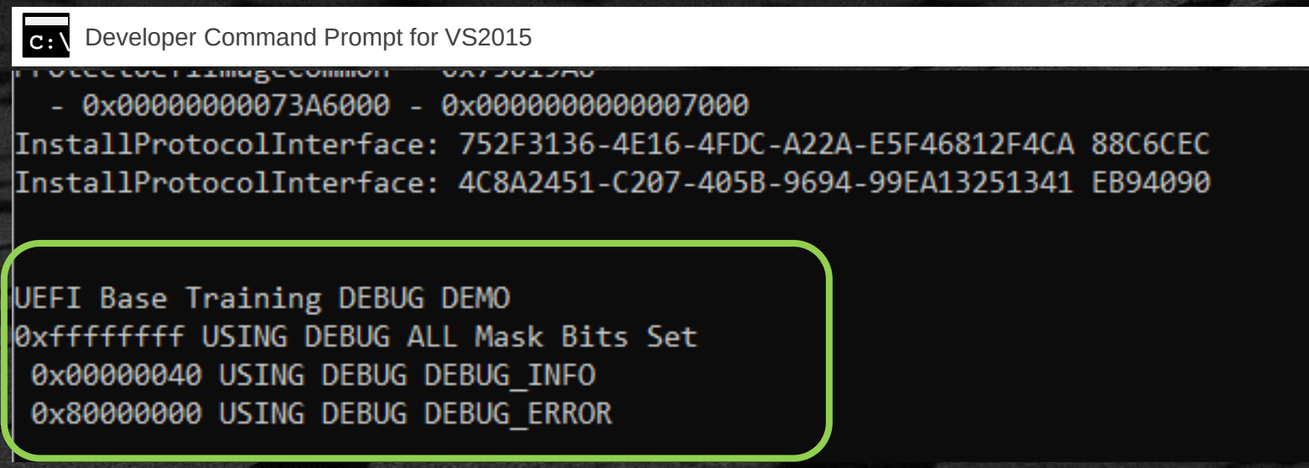
```
Shell> SampleApp
```

Check the VS Debug output

Exit

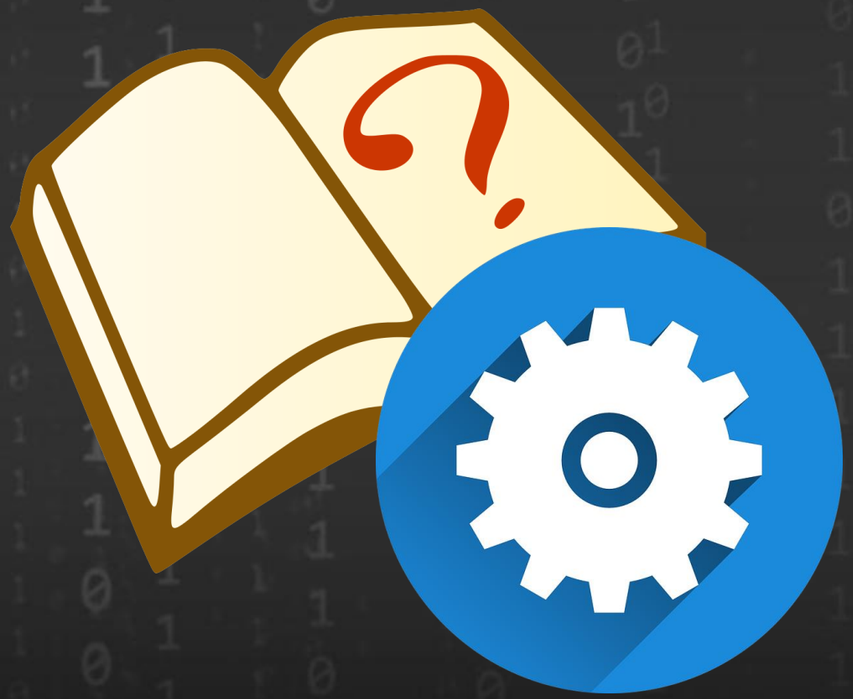
```
Shell> Reset
```

Visual Studio command prompt window output



Lab 2 – Changing PCD Value

In this lab, you'll learn how to use PCD values to change debugging capabilities.



Lab 2: Change PCDs for SampleApp

Open `C:/FW/edk2-ws/edk2/EmulatorPkg/EmulatorPkg.dsc`
Replace `SampleApp/SampleApp.inf` with the following:

```
SampleApp/SampleApp.inf {
   <PcdsFixedAtBuild>
    gEfiMdePkgTokenSpaceGuid.PcdDebugPropertyMask|0xff
    gEfiMdePkgTokenSpaceGuid.PcdDebugPrintErrorLevel|0xffffffff
 }
```

Save and **close** `EmulatorPkg.dsc`

Lab 1: Build, Run and Test Result

At the VS Command Prompt

```
$> Build
$> RunEmulator.bat
```

Run the application from the shell

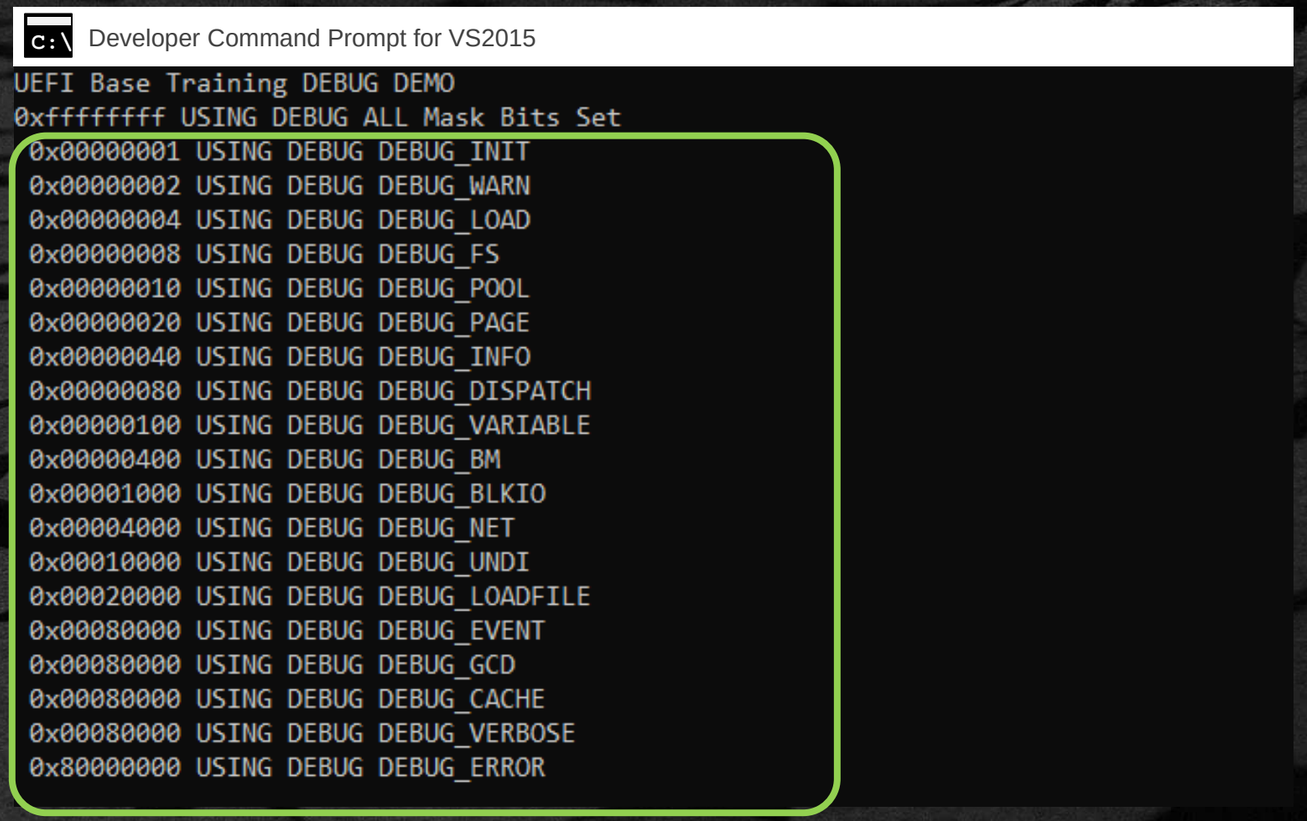
```
Shell> SampleApp
```

Check the VS Debug output

Exit

```
Shell> Reset
```

Visual Studio command prompt window output



CHANGING FLAGS

Changing Compiler & Linker Flags

Precedence for Debug Flags Hierarchy

DSC [BuildOptions] section
rm

INF [BuildOptions] section

DSC <BuildOptions> under a specific module

1. Tools_def.txt
2. DSC [BuildOptions] section (platform scope)
3. INF [BuildOptions] section (module scope)
4. DSC <BuildOptions> under a specific module

tianocore

Compiler / Linker Flags

Example from Microsoft* compiler to turn off optimization

"/O2" to "/O1" requires "/Od /O1" flags

Change common flags in platform DSC

```
[BuildOptions]
    DEBUG_*_IA32_CC_FLAGS = /Od /Oy-
```

Change a single module's flags in DSC

```
MyPath/MyModule.inf {
<BuildOptions>
   DEBUG_*_IA32_CC_FLAGS = /Od /Oy-
}
```

DebugLib USAGE

The DebugLib Class Interface

MdePkg\Include\Library\DebugLib.h

Macros

(where PCDs are checked)

```
ASSERT (Expression)
DEBUG (Expression)
ASSERT_EFI_ERROR (StatusParameter)
ASSERT_PROTOCOL_ALREADY_INSTALLED(…)
```

Advanced Macros

```
DEBUG_CODE (Expression)
DEBUG_CODE_BEGIN() & DEBUG_CODE_END()
DEBUG_CLEAR_MEMORY(…)
```

tianocore

Implementation

DebugLib Instances (1)

BaseDebugLibSerialPort

- Instance of `DebugLib`
- Uses `SerialPortLib` class to send debug output to serial port
- Default for many platforms: `BaseDebugLibNull`
- OVMF uses it with Switch `DEBUG_ON_SERIAL_PORT`

tianocore

DebugLib Instances (2)

Implementation

UefiDebugLibConOut UefiDebugLibStdErr

- Instances of `DebugLib` (for apps and drivers)
- Send all debug output to console/debug console

DebugLib Instances (3)

Implementation

PeiDxeDebugLibReportStatusCode

- Sends ASCII String specified by Description Value to the `ReportStatusCode()`
- May also use the `SerialPortLib` class to send debug output to serial port
- `BaseDebugLibNull` - Resolves references

Default for most platforms

Changing Library Instances

Change common library instances in the platform DSC by module type

```
[LibraryClasses.common.IA32]
DebugLib|MdePkg/Library/BaseDebugLibNull/BaseDebugLibNull.inf
```

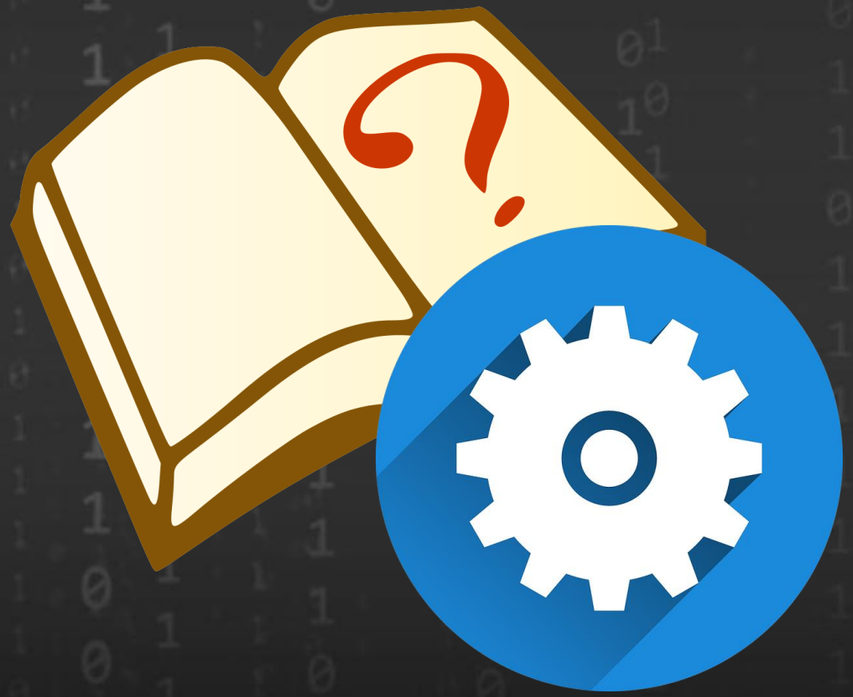
Change a single module's library instance in the platform DSC

```
MyPath/MyModule.inf {
<LibraryClasses>
DebugLib|MdePkg/Library/BaseDebugLibSerialPort.inf
}
```

tianocore

Lab 2 – Library Instances for Debugging

In this lab, you'll learn how to add specific debug library instances.



Lab 3: Using Library Instances for Debugging

Open `C:/FW/edk2-ws/edk2/EmulatorPkg/EmulatorPkg.dsc`
Replace `SampleApp/SampleApp.inf { . . .}` with the following:

```
SampleApp/SampleApp.inf {
   <LibraryClasses>
    DebugLib|MdePkg/Library/UefiDebugLibConOut/UefiDebugLibConOut.inf
 }
```

Save and **close** `EmulatorPkgPkg.dsc`

Lab 3: Build, Run and Test Result

At the VS Command Prompt

```
$> Build
$> RunEmulator.bat
```

Run the application from the shell

```
Shell> SampleApp
```

See that the output from the Debug statements now goes to the console

Exit

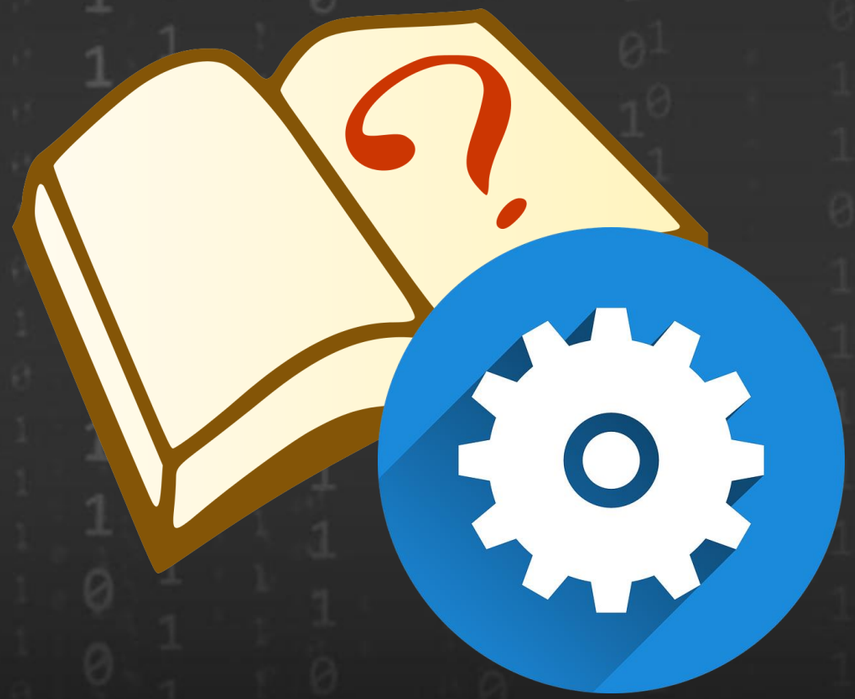
```
Shell> Reset
```

Debug output to console

```
Shell>
Shell> sampleapp


UEFI Base Training DEBUG DEMO
0xffffffff USING DEBUG ALL Mask Bits Set
 0x00000001 USING DEBUG EFI_D_INIT
 0x00000002 USING DEBUG EFI_D_WARN
 0x00000004 USING DEBUG EFI_D_LOAD
 0x00000008 USING DEBUG EFI_D_FS
 0x00000040 USING DEBUG EFI_D_INFO
 0x80000000 USING DEBUG EFI_D_ERROR
System Table: 0x07E33018

Press any Key to continue :

Enter text. Include a dot ('.') in a sentence then <Enter> to exit

.
Shell> _
```

Lab 4: Null Instance of `DebugLib`

In this lab, you'll change the `DebugLib` to the Null instance.



Lab 4: Using Null Library Instances

Open C:/FW/edk2-ws/edk2/EmulatorPkg/EmulatorPkg.dsc
Replace SampleApp/SampleApp.inf { . . .} with the following:

```
SampleApp/SampleApp.inf {
   <LibraryClasses>
   DebugLib|MdePkg/Library/BaseDebugLibNull/BaseDebugLibNull.inf
}
```

Save and **close** EmulatorPkg.dsc

Lab 4: Build, Run and Test Result

At the VS Command Prompt

```
$> Build
$> RunEmulator.bat
```

Run the application from the shell

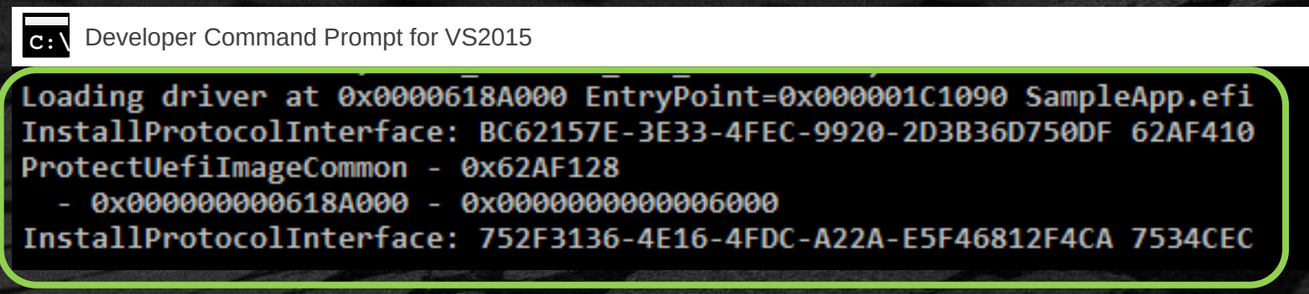
```
Shell> SampleApp
```

Check – now **NO** Debug output

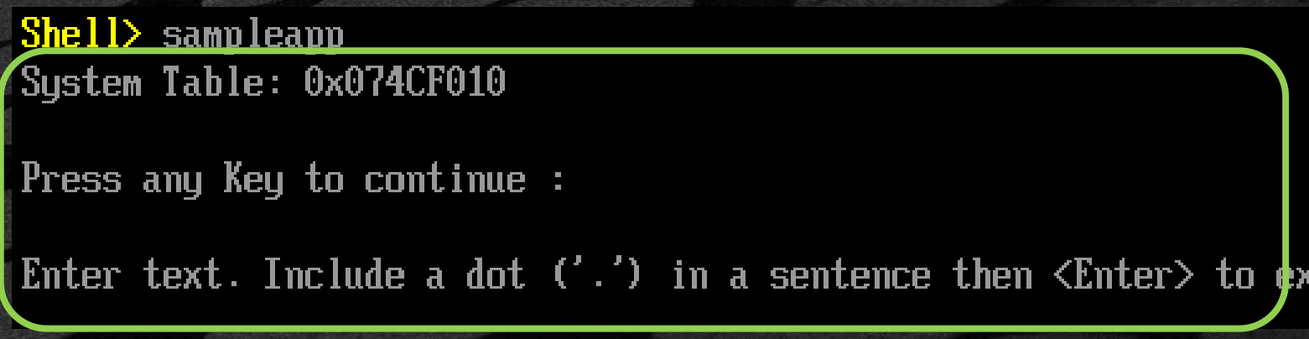
Exit

```
Shell> Reset
```

Visual Studio command prompt window output – NO DEBUG

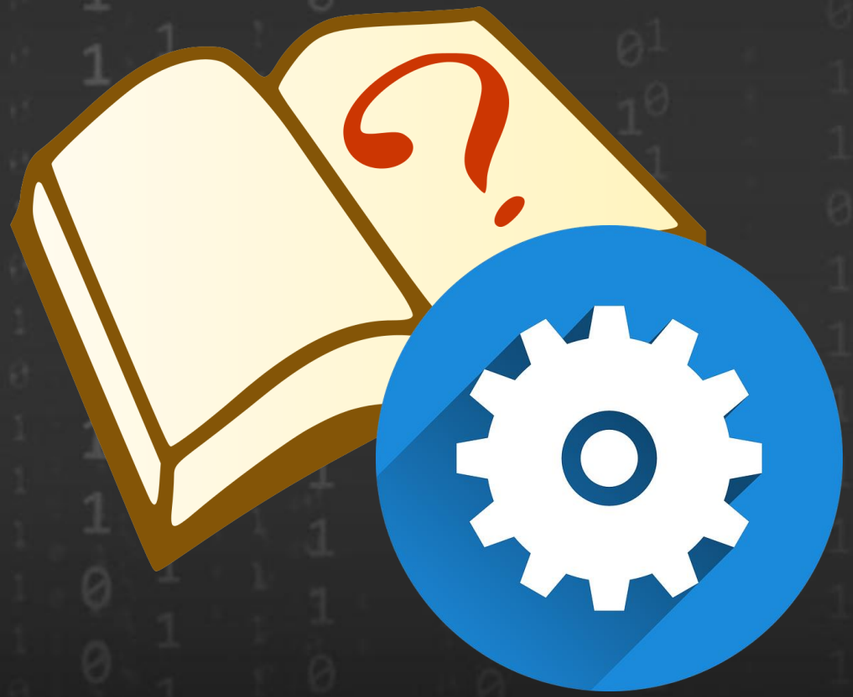


Console window – NO DEBUG



Lab 5: Debugging EDK II with VS Debugger

In this lab, you'll learn how setup the VS to debug the EDK II emulation



Lab 5: Debug with VS

Edit the SampleApp.c and add an "**ASSERT_EFI_ERROR**" :
Add the following:

```
EFI_STATUS      Status;
Status = EFI_NO_RESPONSE;
      . . .
ASSERT_EFI_ERROR(Status);
```

```
EFI_STATUS Status;
Status = EFI_NO_RESPONSE;  // or any EFI Error

DEBUG((0xffffffff, "\n\nUEFI Base Training DEBUG DEMO\n"));
DEBUG((0xffffffff, "0xffffffff USING DEBUG ALL Mask Bits Set\n"));

ASSERT_EFI_ERROR(Status);
```

Save SampleApp.c

Lab 5: Debug with VS

At the VS Command Prompt

```
$> Build
$> RunEmulator.bat
```

Run the application from the shell

```
Shell> SampleApp
```

Assert in VS Command Prompt

Visual Studio command prompt window output

```
Developer Command Prompt for VS2015 - runEmulator.bat
Loading driver at 0x2C1AEC13000 EntryPoint=0x0006BB11000 Shell.efi
InstallProtocolInterface: BC62157E-3E33-4FEC-9920-2D3B36D750DF 2C1AF262298
ProtectUefiImageCommon - 0xAF110840
  - 0x000002C1AEC13000 - 0x0000000000133000
PROGRESS CODE: V03058001 I0
InstallProtocolInterface: 387477C2-69C7-11D2-8E39-00A0C969723B 2C1AF011B20
InstallProtocolInterface: 752F3136-4E16-4FDC-A22A-E5F46812F4CA 2C1AF010F98
InstallProtocolInterface: 6302D008-7F9B-4F30-87AC-60C9FEF5DA4E 6BBD9A30
[Security] 3rd party image[0] can be loaded after EndOfDxe: VenHw(5CF32E0B-8EDF-2E44-9CDA-93205E99EC1C,000
B22-6459-11D2-8E39-00A0C969723B,00000000)/\sampleapp.EFI.
InstallProtocolInterface: 5B1B31A1-9562-11D2-8E3F-00A0C969723B 2C1AEEEF440
LoadLibraryEx (
  c:\fw\edk2-ws\Build\EmulatorX64\DEBUG_VS2015x86\X64\SampleApp\SampleApp\DEBUG\SampleApp.DLL,
  NULL, DONT_RESOLVE_DLL_REFERENCES)
Loading driver at 0x2C1AEE94000 EntryPoint=0x0006BB01000 SampleApp.efi
InstallProtocolInterface: BC62157E-3E33-4FEC-9920-2D3B36D750DF 2C1AEEF0018
ProtectUefiImageCommon - 0xAEEEF440
  - 0x000002C1AEE94000 - 0x000000000000E000
InstallProtocolInterface: 752F3136-4E16-4FDC-A22A-E5F46812F4CA 2C1AB43D628

>>>>>[UefiMain] Entry point: 0x6BB01244 <<<<<<


UEFI Base Training DEBUG DEMO
0xffffffff USING DEBUG ALL Mask Bits Set

ASSERT_EFI_ERROR (Status = 00000000)

DXE_ASSERT!: [SampleApp] c:\fw\edk2-ws\edk2\SampleApp\SampleApp.c (49): !EFI_ERROR (0x8000000000000000)
```

Lab 5: Debug with VS

Windows* VS Debugger
Will Pop UP

```
//
// Generate a Breakpoint, DeadLoop, or NOP based on PCD settings
//
if ((PcdGet8 (PcdDebugPropertyMask) & DEBUG_PROPERTY_ASSERT_BREAKPOINT_ENABLED) != 0) {
  CpuBreakpoint ();
} else if ((PcdGet8 (PcdDebugPropertyMask) & DEBUG_PROPERTY_ASSERT_DEADLOOP_ENABLED) != 0) {
  CpuDeadLoop ();
}
}
```

SampleApp.c

Miscellaneous Files | (Global Scope)

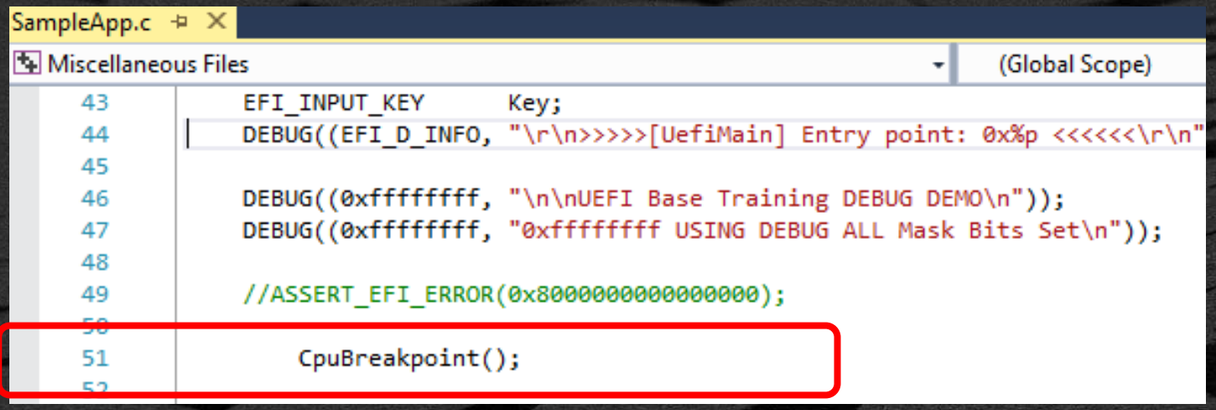
```
43      EFI_INPUT_KEY      Key;
44      DEBUG((EFI_D_INFO, "\r\n>>>>>[UefiMain] Entry point: 0x%p <<<<<<\r\n"
45
46      DEBUG((0xffffffff, "\n\nUEFI Base Training DEBUG DEMO\n"));
47      DEBUG((0xffffffff, "0xffffffff USING DEBUG ALL Mask Bits Set\n"));
48
49      ASSERT_EFI_ERROR(0x8000000000000000);  ≤1ms elapsed
50
```

```
Developer Command Prompt for VS2015 - runEmulator.bat
Loading driver at 0x2C1AEC13000 EntryPoint=0x0006BB11000 Shell.efi
InstallProtocolInterface: BC62157E-3E33-4FEC-9920-2D3B36D750DF 2C1AF262298
                                                              2C1AF011B20
                                                              2C1AF010F98
                                                              5BBD9A30
                                                              Hw(5CF32E0B-8EDF-2E44-9CDA-93205E99EC1C,00000000)
                                                              2C1AEEEF440
                                                              \SampleApp\DEBUG\SampleApp.DLL,
 NULL, DONT_RESOLVE_DLL_REFERENCES)
Loading driver at 0x2C1AEE94000 EntryPoint=0x0006BB01000 SampleApp.efi
InstallProtocolInterface: BC62157E-3E33-4FEC-9920-2D3B36D750DF 2C1AEEF0018
eCommon - 0xAEEEF440
EE94000 - 0x000000000000E000
InstallProtocolInterface: 752F3136-4E16-4FDC-A22A-E5F46812F4CA 2C1AB43D628
Entry point: 0x6BB01244 <<<<<<
G DEBUG ALL Mask Bits Set
R (Status = 00000000)
ampleApp] c:\fw\edk2-ws\edk2\SampleApp\SampleApp.c (49): !EFI_ERROR (0x8000000000000000)
```

Lab 5: Debug with VS

Edit the `SampleApp.c` and add "`CpuBreakpoint();`" Statement and comment out the "`ASSERT`":

```
CpuBreakpoint();
```



Save `SampleApp.c`

Lab 5: Debug with VS

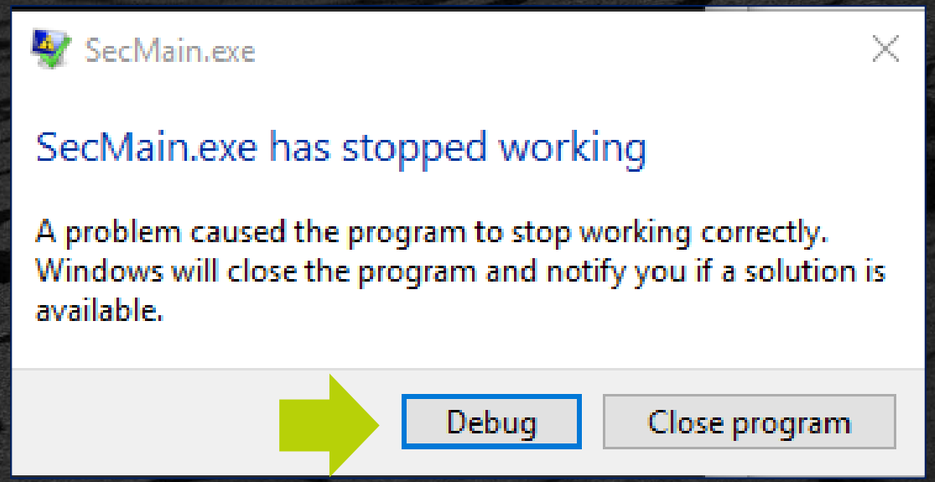
At the VS Command Prompt

```
$> Build
$> RunEmulator.bat
```

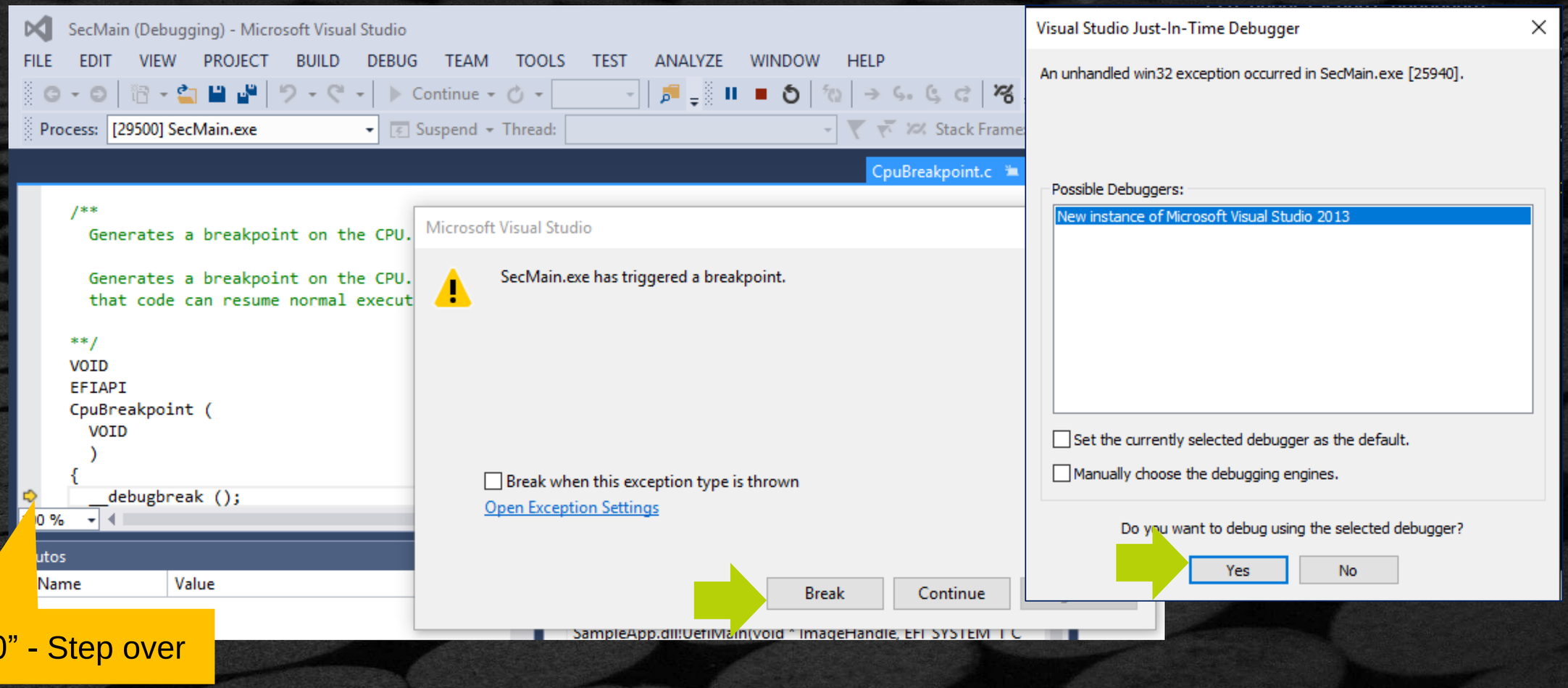
Run the application from the shell

```
Shell> SampleApp
```

VS option go to VS Debugger

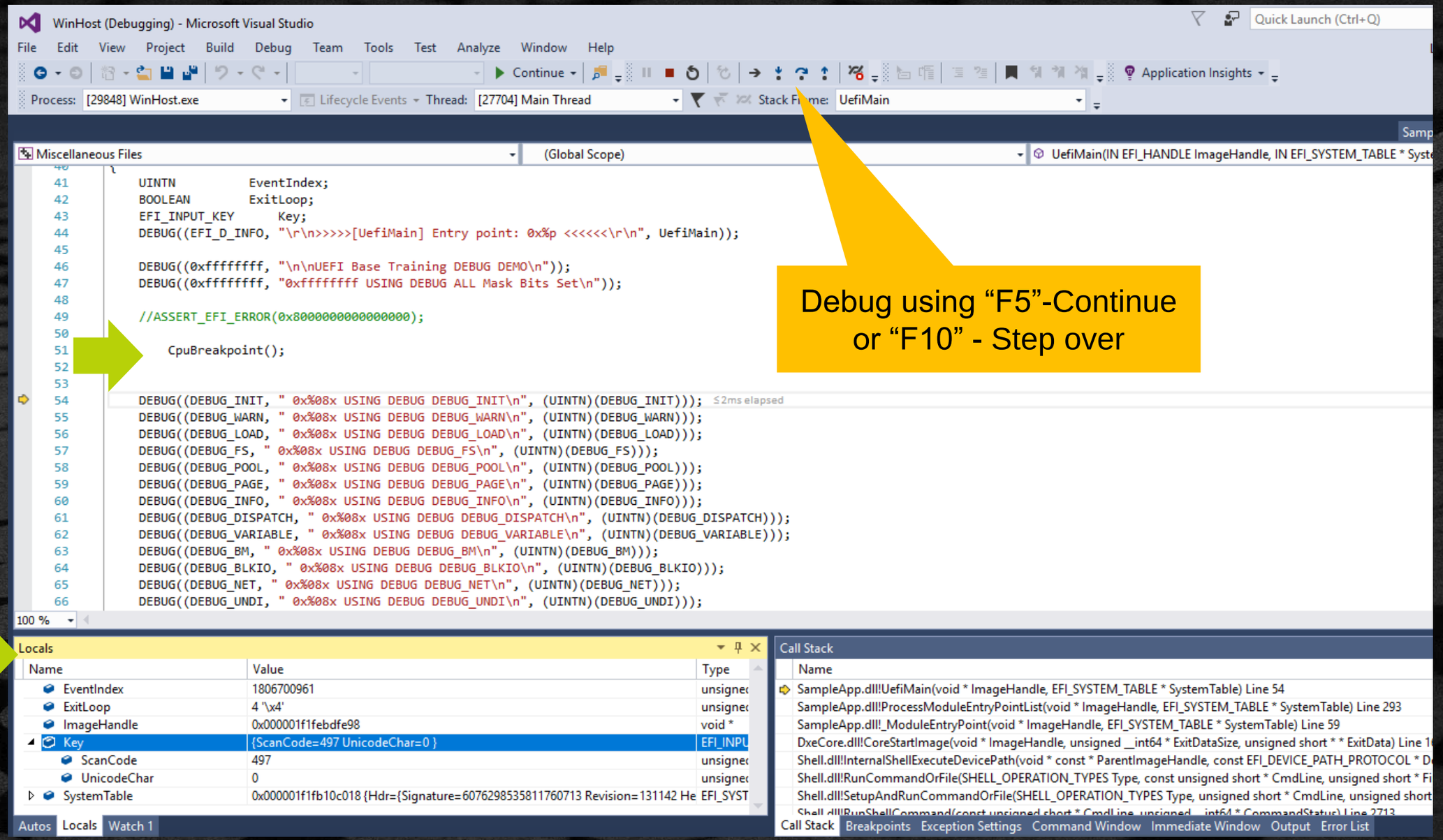


Invoke Windows Visual Studio Debugger



"F10" - Step over

Invoke Windows Visual Studio Debugger



tianocore

SUMMARY

- Define `DebugLib` and its attributes
- List the ways to debug
- Using PCDs to Configure `DebugLib` - LAB
- Change Compiler & Linker Flags for debugging
- Change the `DebugLib` instance to modify the debug output - LAB
- Debug EDK II using VS Debugger - LAB


Questions?

Return to Main Training Page

Return to Training Table of contents for next presentation link


tianocore

tianocore

BACK UP

tianocore

ISSUE:

Debugging in Emulatior with Windows 7 and Visual Studio does not work?

Symptom: With Windows 7 a **`CpuBreakpoint()`** or **`ASSERT`** just exits with an error from the “Build Run” command.

Link to fix this issue:
https://github.com/tianocore/tianocore.github.io/wiki/NT32#Debugging_in_Nt32_Emulation_with_Windows_7_and_Visual_Studio_does_not_work

1. Run the RegEdt32
2. Navigate to the HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\AeDebug
3. Add a string value entry called "Auto" with a value of "1“

Windows 10 Visual Studio does not seem to have this issue