

UEFI & EDK II Training

EDK II Debugging

tianocore.org



Lesson Objective

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

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

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
```

Aliases EFI_D_INIT == DEBUG_INIT, etc..

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`

Demo – Adding Debug Statements

Adding debug statements to the previous lab's SampleApp UEFI Shell application

Demo: Add debug statements to SampleApp

The following code was added 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)) );
```

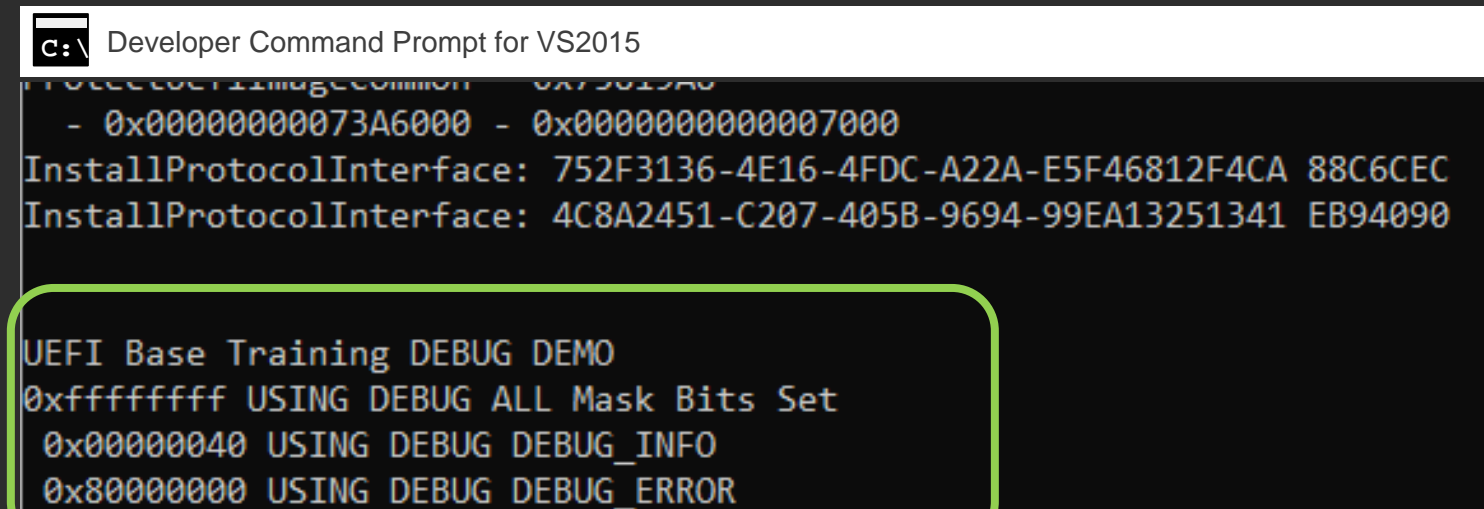
Demo: Run and Test Result

Run the application from the shell

```
Shell> SampleApp
```

Check the VS Debug output

Visual Studio command prompt window output



```
c:\> Developer Command Prompt for VS2015
UEFI Base Training DEBUG DEMO
0xffffffff USING DEBUG ALL Mask Bits Set
0x00000040 USING DEBUG DEBUG_INFO
0x80000000 USING DEBUG DEBUG_ERROR
```

Demo: Change PCDs for SampleApp

The following was added to EmulatorPkg.dsc

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


Demo: Build, Run and Test Result

Run the application from the shell

```
Shell> SampleApp
```

Check the VS Debug output

Visual Studio command prompt window output

```
C:\> Developer Command Prompt for VS2015
UEFI Base Training DEBUG DEMO
0xffffffff USING DEBUG ALL Mask Bits Set
0x00000001 USING DEBUG DEBUG_INIT
0x00000002 USING DEBUG DEBUG_WARN
0x00000004 USING DEBUG DEBUG_LOAD
0x00000008 USING DEBUG DEBUG_FS
0x00000010 USING DEBUG DEBUG_POOL
0x00000020 USING DEBUG DEBUG_PAGE
0x00000040 USING DEBUG DEBUG_INFO
0x00000080 USING DEBUG DEBUG_DISPATCH
0x00000100 USING DEBUG DEBUG_VARIABLE
0x00000400 USING DEBUG DEBUG_BM
0x00001000 USING DEBUG DEBUG_BLKIO
0x00004000 USING DEBUG DEBUG_NET
0x00010000 USING DEBUG DEBUG_UNDI
0x00020000 USING DEBUG DEBUG_LOADFILE
0x00080000 USING DEBUG DEBUG_EVENT
0x00080000 USING DEBUG DEBUG_GCD
0x00080000 USING DEBUG DEBUG_CACHE
0x00080000 USING DEBUG DEBUG_VERBOSE
0x80000000 USING DEBUG DEBUG_ERROR
```

CHANGING FLAGS

Changing Compiler & Linker Flags

Precedence for Debug Flags Hierarchy

DSC [BuildOptions] section
(platform scope)

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

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(...)
```



DebugLib Instances (1)

Implementation

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



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  
}
```

Demo – Library Instances for Debugging

Changing specific debug library instances.

Demo: Using Library Instances for Debugging

The following was added to EmulatorPkg.dsc changing the library instances

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

Demo: Debug Output in the Console

Application from the shell

```
Shell> SampleApp
```

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

Debug output to console

```
Shell> sampleapp
```

```
UEFI Base Training DEBUG DEMO
0xffffffff USING DEBUG ALL Mask Bits Set
0x00000040 USING DEBUG DEBUG_INFO
0x80000000 USING DEBUG DEBUG_ERROR
System Table: 0xB7A7C018
```

```
Press any Key to continue :
```

EmulatorPkg

Demo: Debugging EDK II with VS Debugger

SampleApp.c has an
“ASSERT_EFI_ERROR” statement added

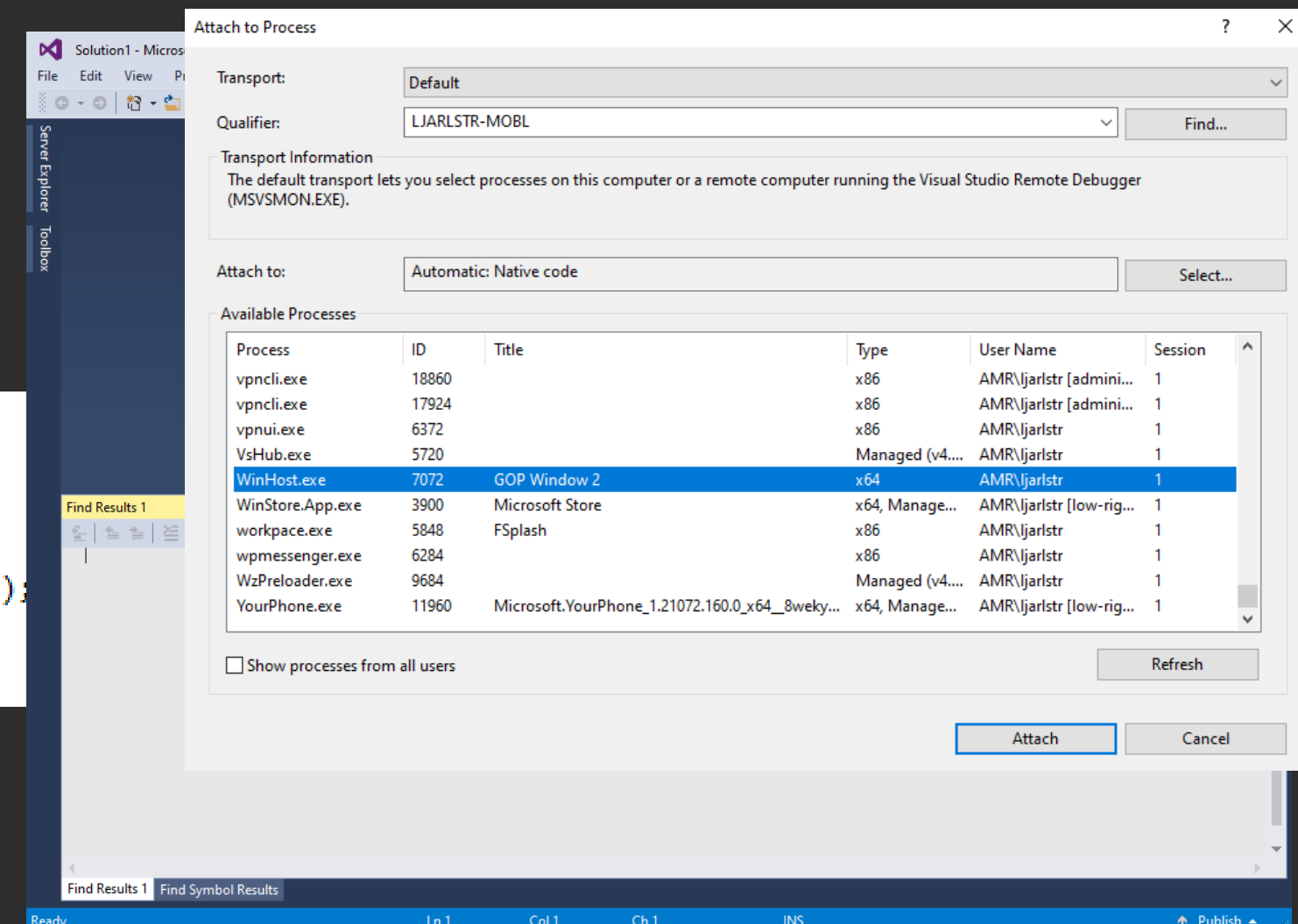
Visual Studio enable the WinHost for
Debugging

```
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);
```



Demo: Debug with VS - ASSERT

Application from the shell

```
Shell> SampleApp
```

Assert in VS Command Prompt

Visual Studio command prompt window output

```
Developer Command Prompt for VS2015 - runEmulator.bat
InstallProtocolInterface: 5B1B31A1-9562-11D2-8E3F-00A0C969723B 1D55B83F440
LoadLibraryEx (
  c:\fw\edk2-ws\Build\EmulatorX64\DEBUG_VS2015x86\X64\SampleApp\SampleApp\DEBUG\SampleApp.DLL,
  NULL, DONT_RESOLVE_DLL_REFERENCES)
Loading driver at 0x1D55B7E4000 EntryPoint=0x00077441000 SampleApp.efi
InstallProtocolInterface: BC62157E-3E33-4FEC-9920-2D3B36D750DF 1D55B840018
ProtectUefiImageCommon - 0x5B83F440
- 0x0000001D55B7E4000 - 0x0000000000000E000
InstallProtocolInterface: 752F3136-4E16-4FDC-A22A-E5F46812F4CA 1D557D8D628

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

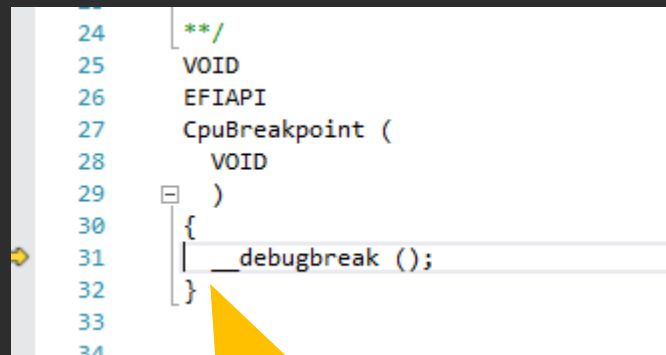
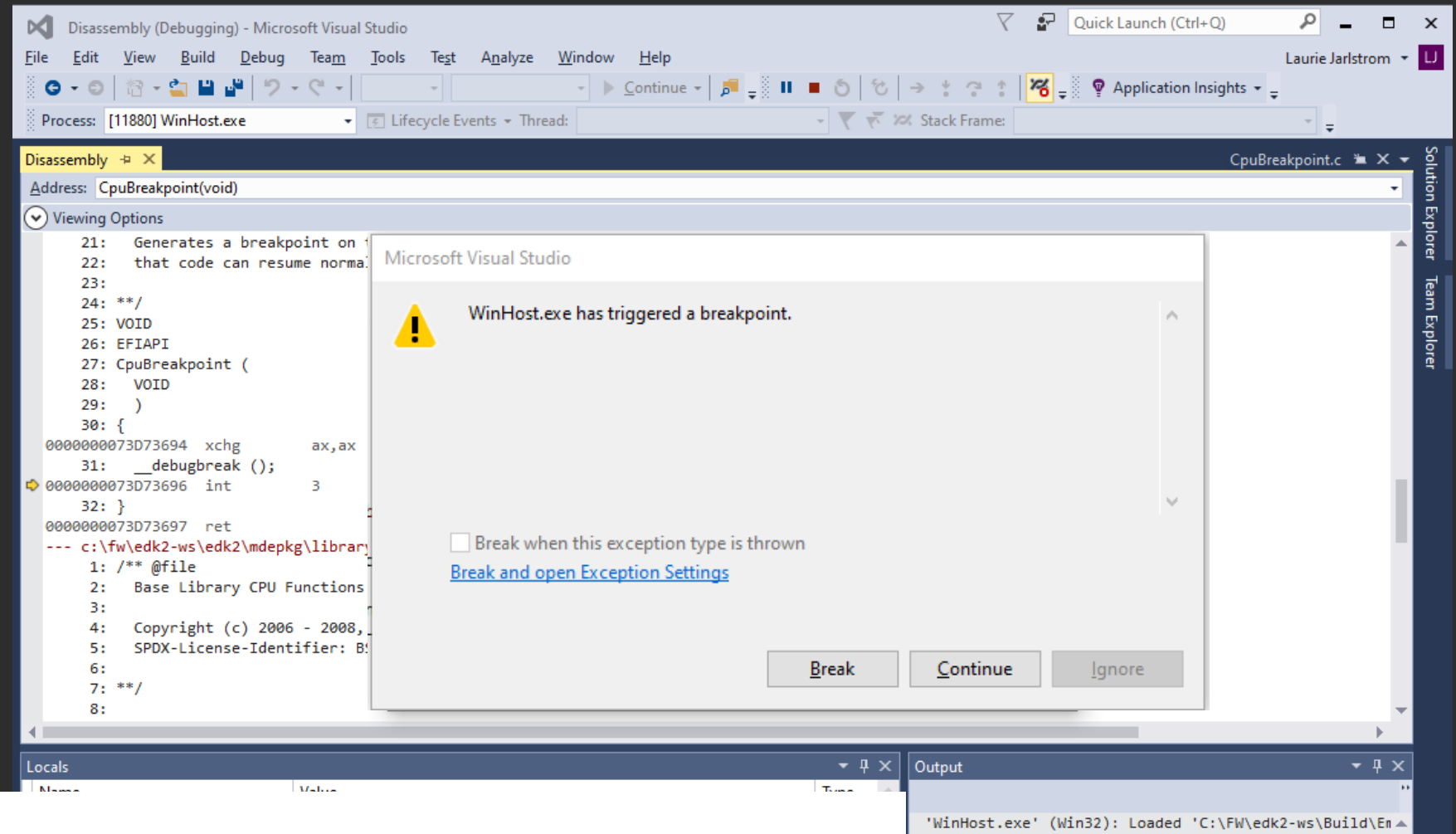
ASSERT_EFI_ERROR (Status = No Response)

DXE_ASSERT!: [SampleApp] c:\fw\edk2-ws\edk2\SampleApp\SampleApp.c (51): !EFI_ERROR (Status)
```

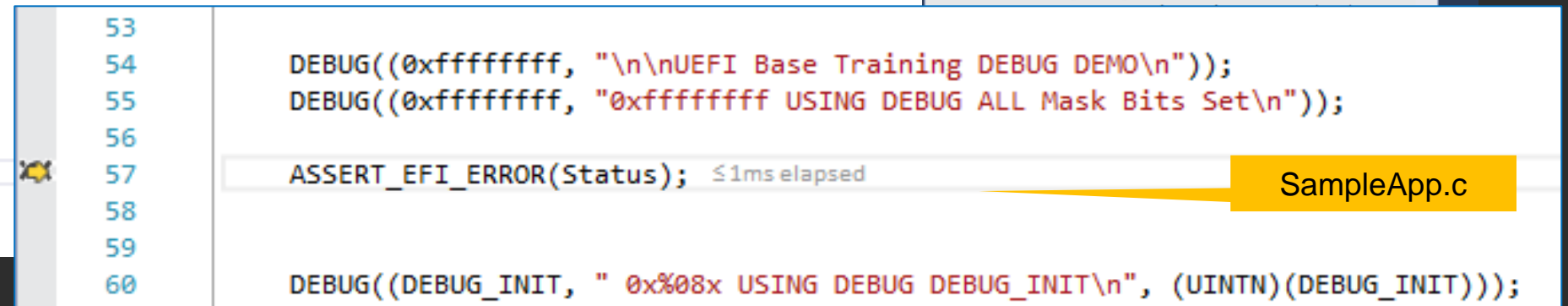
Lab 5: Debug with VS - ASSERT

Windows* VS Debugger

“F5” to continue
“Shift F5” to Stop
debugging



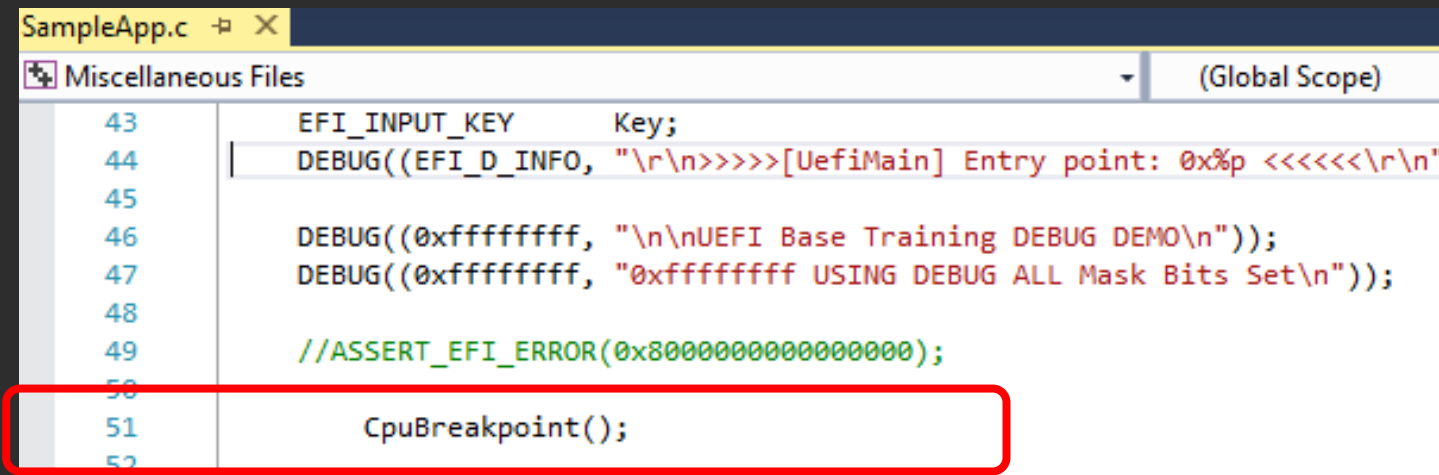
“F10” - Step over 2-3 times



Demo: Debug with VS - CpuBreakpoint

SampleApp.c with “CpuBreakpoint();” Statement and commented out the “ASSERT”

CpuBreakpoint();



The screenshot shows the Visual Studio code editor with the file 'SampleApp.c' open. The editor displays the following code:

```
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);  
50  
51  CpuBreakpoint();  
52
```

The line 'CpuBreakpoint();' at line 51 is highlighted with a red rectangle. The 'Miscellaneous Files' pane on the left shows '(Global Scope)'.

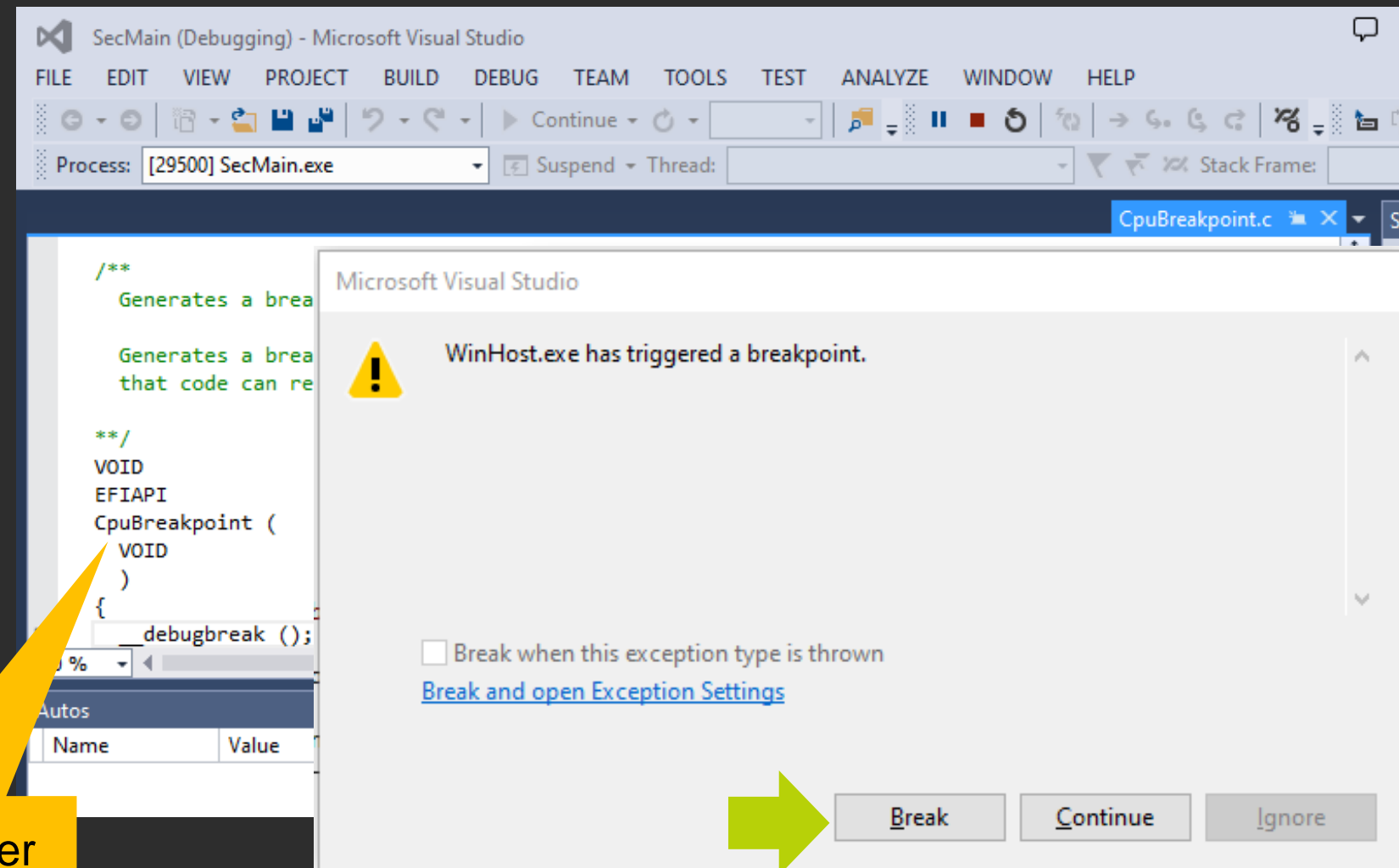
Demo: Debug with VS

Application from the shell

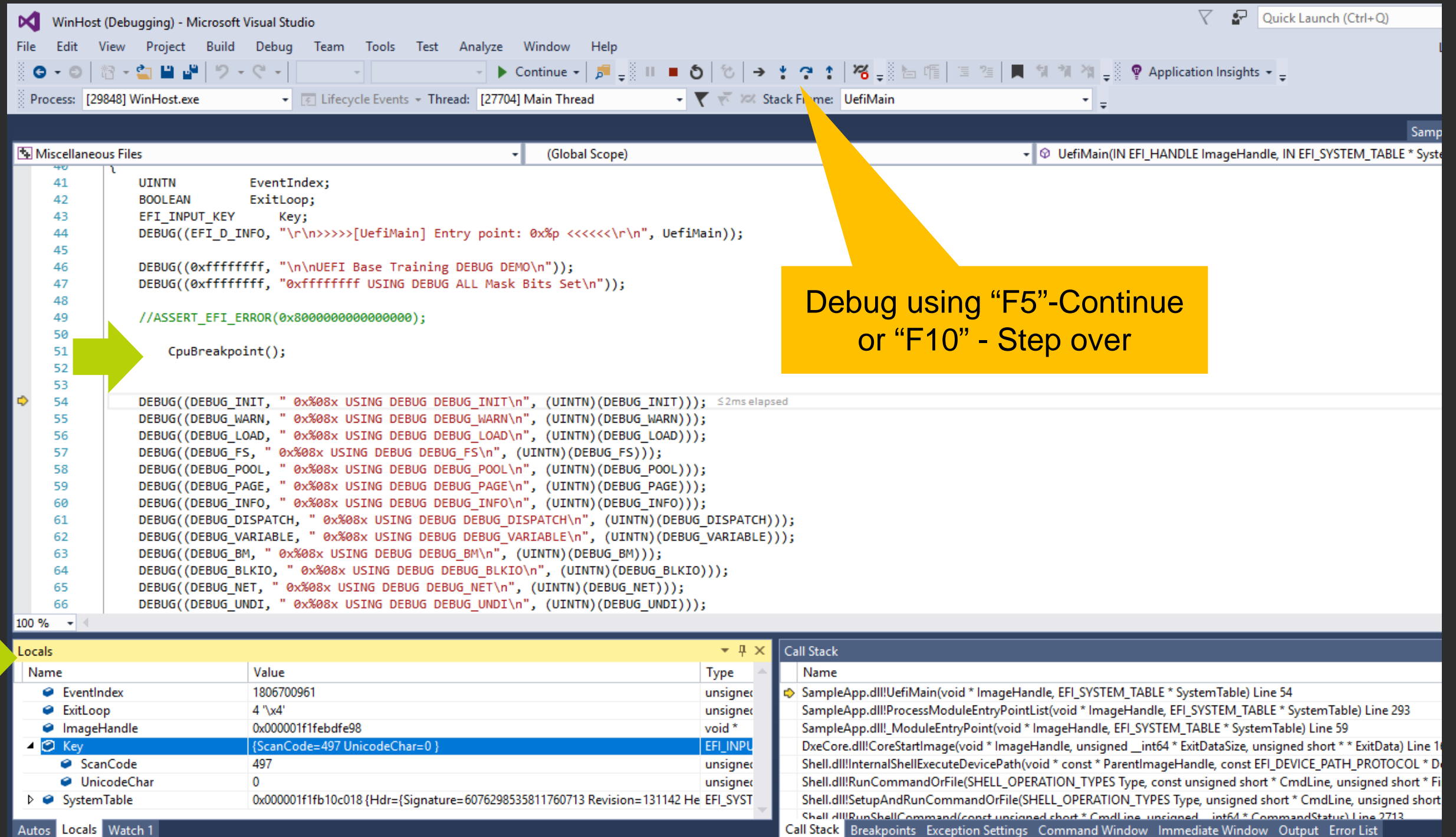
```
Shell> SampleApp
```

VS Debugger pop up,
Press “F10” until
SampleApp.c shows

“F10” - Step over



Demo Windows Visual Studio Debugger



WinHost (Debugging) - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Process: [29848] WinHost.exe Lifecycle Events Thread: [27704] Main Thread Stack Frame: UefiMain

Miscellaneous Files (Global Scope) UefiMain(IN EFI_HANDLE ImageHandle, IN EFI_SYSTEM_TABLE * SystemTable)

```

41  UINTN      EventIndex;
42  BOOLEAN    ExitLoop;
43  EFI_INPUT_KEY  Key;
44  DEBUG((EFI_D_INFO, "\r\n>>>>[UefiMain] Entry point: 0x%p <<<<<\r\n", UefiMain));
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);
50
51  CpuBreakpoint();
52
53
54  DEBUG((DEBUG_INIT, " 0x%08x USING DEBUG DEBUG_INIT\n", (UINTN)(DEBUG_INIT)));
55  DEBUG((DEBUG_WARN, " 0x%08x USING DEBUG DEBUG_WARN\n", (UINTN)(DEBUG_WARN)));
56  DEBUG((DEBUG_LOAD, " 0x%08x USING DEBUG DEBUG_LOAD\n", (UINTN)(DEBUG_LOAD)));
57  DEBUG((DEBUG_FS, " 0x%08x USING DEBUG DEBUG_FS\n", (UINTN)(DEBUG_FS)));
58  DEBUG((DEBUG_POOL, " 0x%08x USING DEBUG DEBUG_POOL\n", (UINTN)(DEBUG_POOL)));
59  DEBUG((DEBUG_PAGE, " 0x%08x USING DEBUG DEBUG_PAGE\n", (UINTN)(DEBUG_PAGE)));
60  DEBUG((DEBUG_INFO, " 0x%08x USING DEBUG DEBUG_INFO\n", (UINTN)(DEBUG_INFO)));
61  DEBUG((DEBUG_DISPATCH, " 0x%08x USING DEBUG DEBUG_DISPATCH\n", (UINTN)(DEBUG_DISPATCH)));
62  DEBUG((DEBUG_VARIABLE, " 0x%08x USING DEBUG DEBUG_VARIABLE\n", (UINTN)(DEBUG_VARIABLE)));
63  DEBUG((DEBUG_BM, " 0x%08x USING DEBUG DEBUG_BM\n", (UINTN)(DEBUG_BM)));
64  DEBUG((DEBUG_BLKIO, " 0x%08x USING DEBUG DEBUG_BLKIO\n", (UINTN)(DEBUG_BLKIO)));
65  DEBUG((DEBUG_NET, " 0x%08x USING DEBUG DEBUG_NET\n", (UINTN)(DEBUG_NET)));
66  DEBUG((DEBUG_UNDI, " 0x%08x USING DEBUG DEBUG_UNDI\n", (UINTN)(DEBUG_UNDI)));

```

100 %

Locals

Name	Value	Type
EventIndex	1806700961	unsigned int
ExitLoop	4 '\x4'	unsigned char
ImageHandle	0x000001f1febdfe98	void *
Key	{ScanCode=497 UnicodeChar=0}	EFI_INPUT_KEY
ScanCode	497	unsigned short
UnicodeChar	0	unsigned char
SystemTable	0x000001f1fb10c018 {Hdr={Signature=6076298535811760713 Revision=131142 He EFI_SYST	EFI_SYSTEM_TABLE

Autos Locals Watch 1

Call Stack

Name
SampleApp.dll!UefiMain(void * ImageHandle, EFI_SYSTEM_TABLE * SystemTable) Line 54
SampleApp.dll!ProcessModuleEntryPointList(void * ImageHandle, EFI_SYSTEM_TABLE * SystemTable) Line 293
SampleApp.dll!ModuleEntryPoint(void * ImageHandle, EFI_SYSTEM_TABLE * SystemTable) Line 59
DxeCore.dll!CoreStartImage(void * ImageHandle, unsigned __int64 * ExitDataSize, unsigned short * ExitData) Line 14
Shell.dll!InternalShellExecuteDevicePath(void * const * ParentImageHandle, const EFI_DEVICE_PATH_PROTOCOL * DevicePath) Line 14
Shell.dll!RunCommandOrFile(SHELL_OPERATION_TYPES Type, const unsigned short * CmdLine, unsigned short * ExitData) Line 14
Shell.dll!SetupAndRunCommandOrFile(SHELL_OPERATION_TYPES Type, unsigned short * CmdLine, unsigned short * ExitData) Line 14
Shell.dll!RunShellCommand(const unsigned short * CmdLine, unsigned __int64 * CommandStatus) Line 2713

Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error List

Summary

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

Questions?



Return to Main Training Page



Return to Training Table of contents for next presentation [link](#)



ACKNOWLEDGEMENTS

Redistribution and use in source (original document form) and 'compiled' forms (converted to PDF, epub, HTML and other formats) with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code (original document form) must retain the above copyright notice, this list of conditions and the following disclaimer as the first lines of this file unmodified.

Redistributions in compiled form (transformed to other DTDs, converted to PDF, epub, HTML and other formats) must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS DOCUMENTATION IS PROVIDED BY TIANOCORE PROJECT "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL TIANOCORE PROJECT BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (c) 2021, Intel Corporation. All rights reserved.

BACK UP

ISSUE:

Debugging in Emulator with Windows 7/10 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 may not have this issue