tianocore

UEFI & EDK II Training

EDK II Debugging

tianocore.org

Lesson Objective

- Debugging Overview
- 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

Bugs & Generic Debug Methods

Bugs

What’s the hardest bug you ever met?

Generic Debug Methods

Print Statements

Assertions

UEFI Shell

GUI Debuggers

UEFI Debug Methods

Print

DEBUG()

DEBUG_CODE_BEGIN()

DEBUG_CODE_END()

Assertion

ASSERT_EFI_ERROR()/ASSERT()

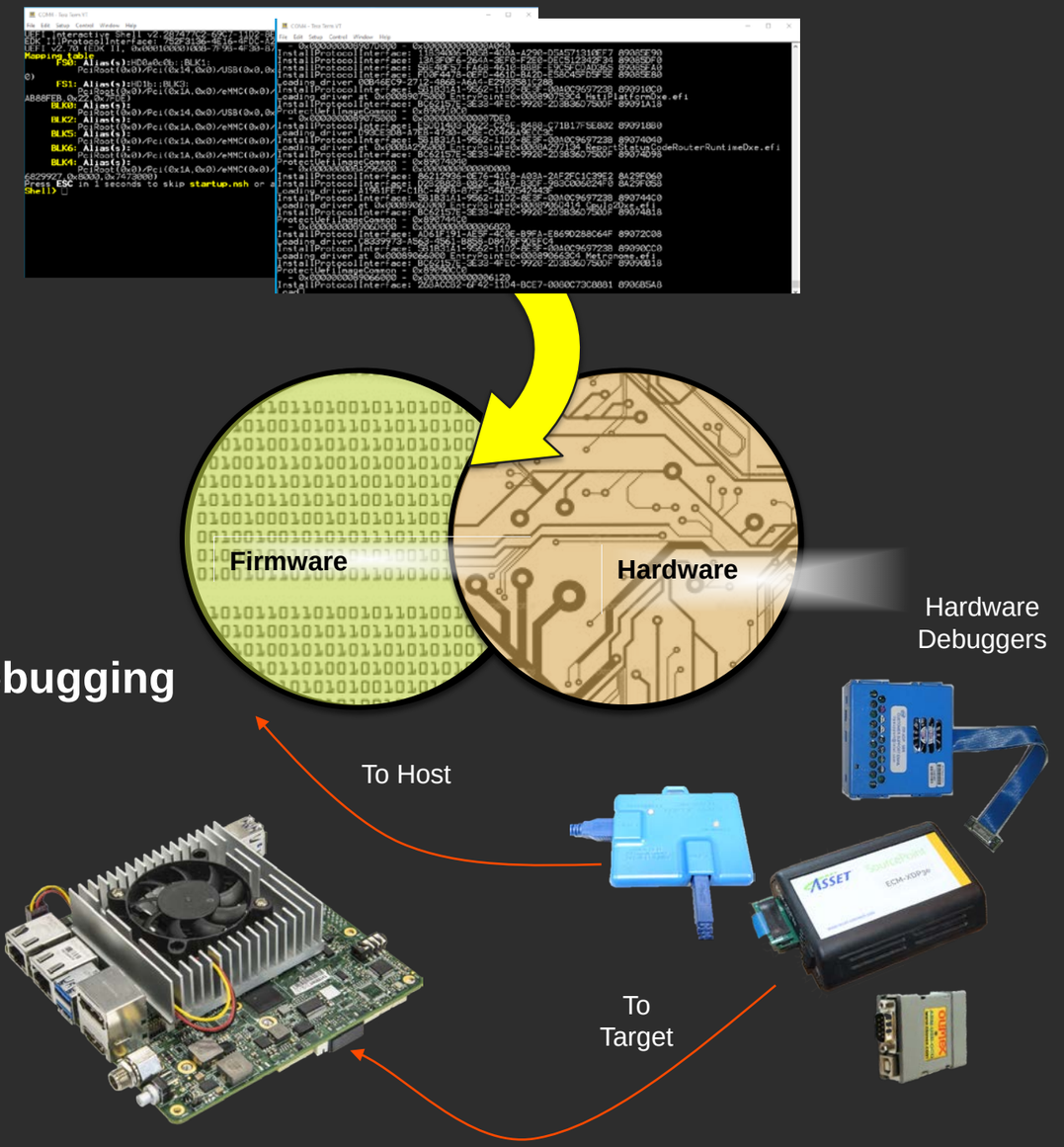
UEFI Shell

Dh, Dmpstore, Mem, MemMap, etc.

GUI debugger

Software/hardware debuggers

GDB, Visual Studio, Asset, Etc.



UEFI Debug Methods

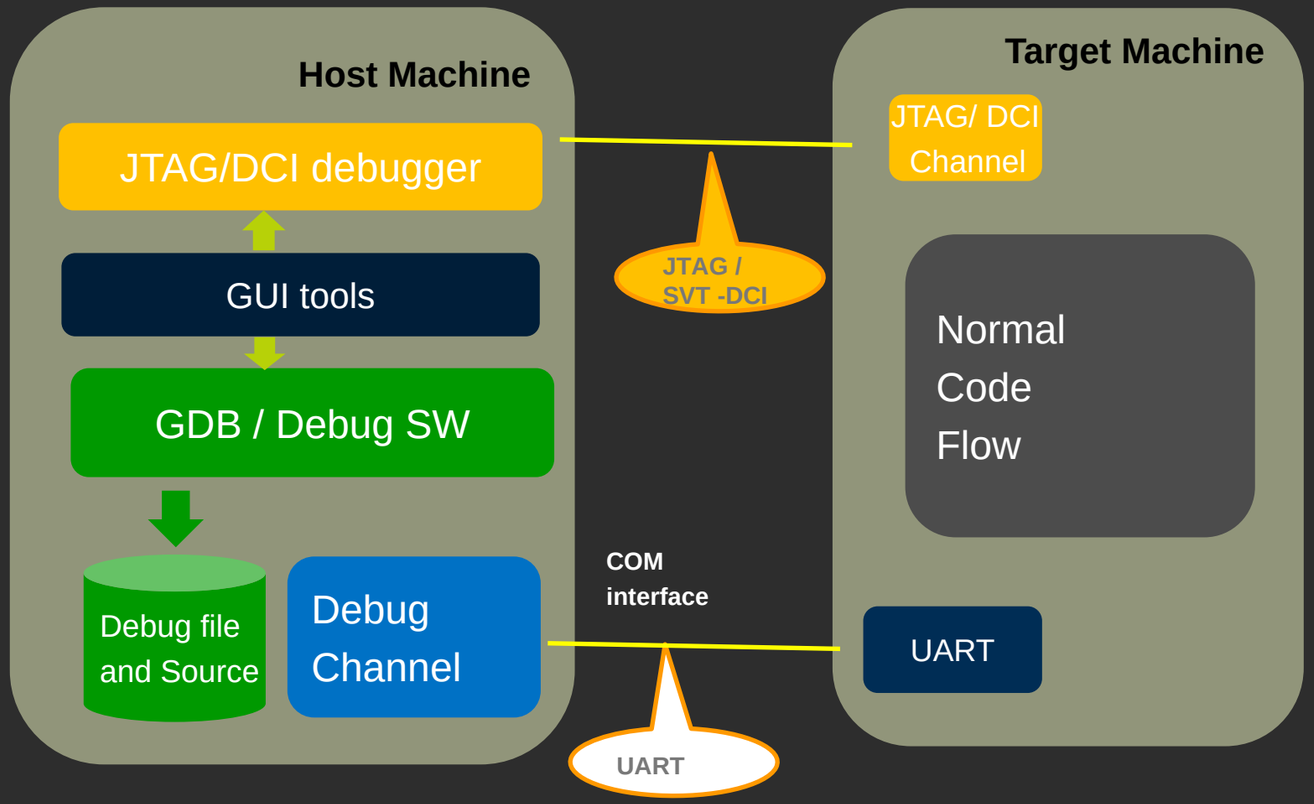
Difference for Hardware Debuggers

- Target & host must co-work

  Some logic only works with timing/register rules

UEFI/FW Debug steps

- Enable debug
- Add DEBUG macros in code
- Check debug output log
- Find module that has issue
- Trace the module until bug root caused

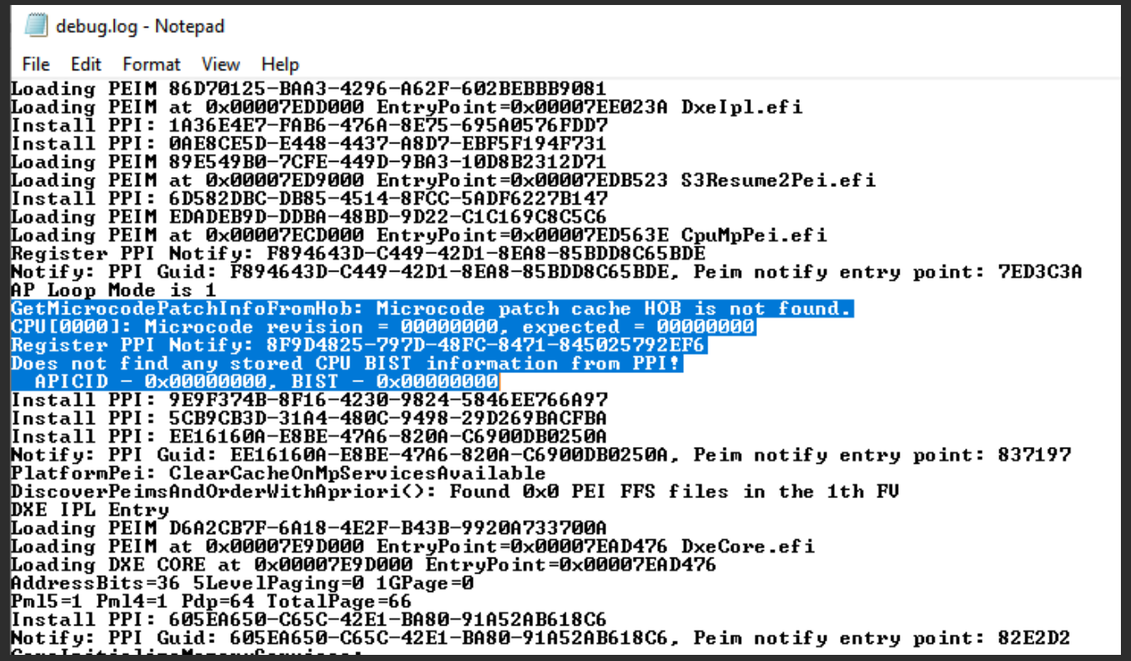


EDK II DebugLib Library

Add **Debug** and **Assert** macros in code

Enable/disable when compiled through PCD, Library Instances & 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

DebugLib Marco Examples

```
#define DEBUG(Expression)                \
    do {                                 \
      if (DebugPrintEnabled ()) {        \
        _DEBUG (Expression);             \
      }                                  \
    } while (FALSE)
```

Config by PCD
PcdDebugPropertyMask

```
VOID
EFIAPI
DebugPrint (
  IN  UINTN        ErrorLevel,
  IN  CONST CHAR8  *Format,
  ...
  )
```

Config by PCD
PcdDebugPrintErrorLevel

```
AhciMode.c (edk2\mdemodulepkg\bus\ata\ataatapipassthru):  DEBUG ((DEBUG_INFO, "Port CMD/DEVSLP = %08x / %08x\n", PortCmd, PortDevSlp));
AhciMode.c (edk2\mdemodulepkg\bus\ata\ataatapipassthru):  DEBUG ((DEBUG_INFO, "IDENTIFY DEVICE: [77] = %04x, [78] = %04x, [79] = %04x\n",
AhciMode.c (edk2\mdemodulepkg\bus\ata\ataatapipassthru):    DEBUG ((DEBUG_INFO, "DevSlp feature is not supported for device at port [%d] PortMultiplier [%d]!\n",
AhciMode.c (edk2\mdemodulepkg\bus\ata\ataatapipassthru):    DEBUG ((DEBUG_INFO, "DevSlp set feature for device at port [%d] PortMultiplier [%d] - %r\n",
AhciMode.c (edk2\mdemodulepkg\bus\ata\ataatapipassthru):  DEBUG ((DEBUG_INFO, "Read Log Ext at port [%d] PortMultiplier [%d] - %r\n", Port, PortMultiplier, Status));
AhciMode.c (edk2\mdemodulepkg\bus\ata\ataatapipassthru):    DEBUG ((DEBUG_INFO, "DevSlpTiming: Supported(%d), Deto(%d), Madt(%d)\n",
AhciMode.c (edk2\mdemodulepkg\bus\ata\ataatapipassthru):  DEBUG ((DEBUG_INFO, "Enabled DevSlp feature at port [%d] PortMultiplier [%d], Port CMD/DEVSLP = %08x / %08x\n",
AhciMode.c (edk2\mdemodulepkg\bus\ata\ataatapipassthru):    DEBUG ((DEBUG_INFO, "CMD_PUIS_SET_DEVICE_SPINUP for device at port [%d] PortMultiplier [%d] - %r!\n",
AhciMode.c (edk2\mdemodulepkg\bus\ata\ataatapipassthru):    DEBUG ((DEBUG_INFO, "Read LBA 0 for device at port [%d] PortMultiplier [%d] - %r!\n",
AhciMode.c (edk2\mdemodulepkg\bus\ata\ataatapipassthru):    DEBUG ((DEBUG_ERROR, "Read IDD failed for device at port [%d] PortMultiplier [%d] - %r!\n",
AhciMode.c (edk2\mdemodulepkg\bus\ata\ataatapipassthru):  DEBUG ((DEBUG_INFO, "IDENTIFY DEVICE: [0] = %016x, [2] = %016x, [83] = %016x, [86] = %016x\n",
AhciMode.c (edk2\mdemodulepkg\bus\ata\ataatapipassthru):  DEBUG ((DEBUG_INFO, "%a PUIS feature at port [%d] PortMultiplier [%d] - %r!\n",
AhciMode.c (edk2\mdemodulepkg\bus\ata\ataatapipassthru):      DEBUG ((EFI_D_WARN,
AhciMode.c (edk2\mdemodulepkg\bus\ata\ataatapipassthru):        DEBUG ((
AhciMode.c (edk2\mdemodulepkg\bus\ata\ataatapipassthru):          DEBUG_INFO, "IDENTIFY DEVICE: [0] = %016x, [2] = %016x, [83] = %016x, [86] = %016x\n",
AhciMode.c (edk2\mdemodulepkg\bus\ata\ataatapipassthru):            DEBUG ((DEBUG_ERROR, "Spin up standby device failed - %r\n", Status));
AhciMode.c (edk2\mdemodulepkg\bus\ata\ataatapipassthru):      DEBUG ((DEBUG_INFO, "port [%d] port multitplier [%d] has a [%a]\n",
AhciMode.c (edk2\mdemodulepkg\bus\ata\ataatapipassthru):        DEBUG ((EFI_D_ERROR, "Calculate Mode Fail, Status = %r\n", Status));
AhciMode.c (edk2\mdemodulepkg\bus\ata\ataatapipassthru):        DEBUG ((EFI_D_ERROR, "Set transfer Mode Fail, Status = %r\n", Status));
AhciMode.c (edk2\mdemodulepkg\bus\ata\ataatapipassthru):          DEBUG ((DEBUG_ERROR, "PUIS enable/disable failed, Status = %r\n", Status));
AhciPei.c (edk2\mdemodulepkg\bus\ata\ahcipei):  DEBUG ((DEBUG_INFO, "%a: Enters.\n", __FUNCTION__));
AhciPei.c (edk2\mdemodulepkg\bus\ata\ahcipei):    DEBUG ((DEBUG_ERROR, "%a: Fail to get the current boot mode.\n", __FUNCTION__));
AhciPei.c (edk2\mdemodulepkg\bus\ata\ahcipei):    DEBUG ((DEBUG_ERROR, "%a: Failed to locate AtaAhciHostControllerPpi.\n", __FUNCTION__));
```

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))  );
```

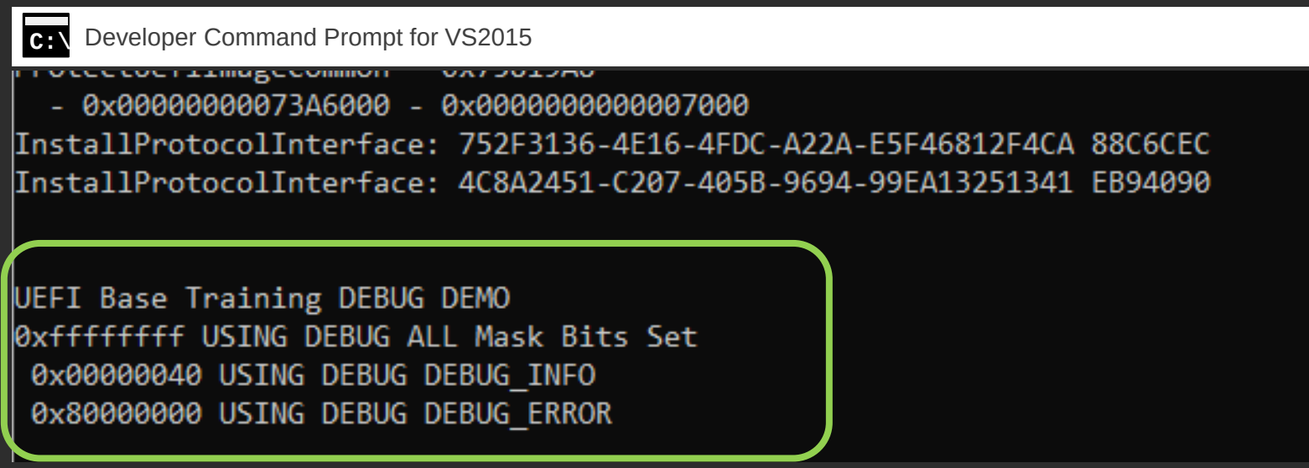
Demo: Run and Test Result

Run the application from the shell

```
Shell> SampleApp
```

Check the VS Debug output

Visual Studio command prompt window output



Demo: Change PCDs for SampleApp

The following was added to `EmulatorPkg.dsc`

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

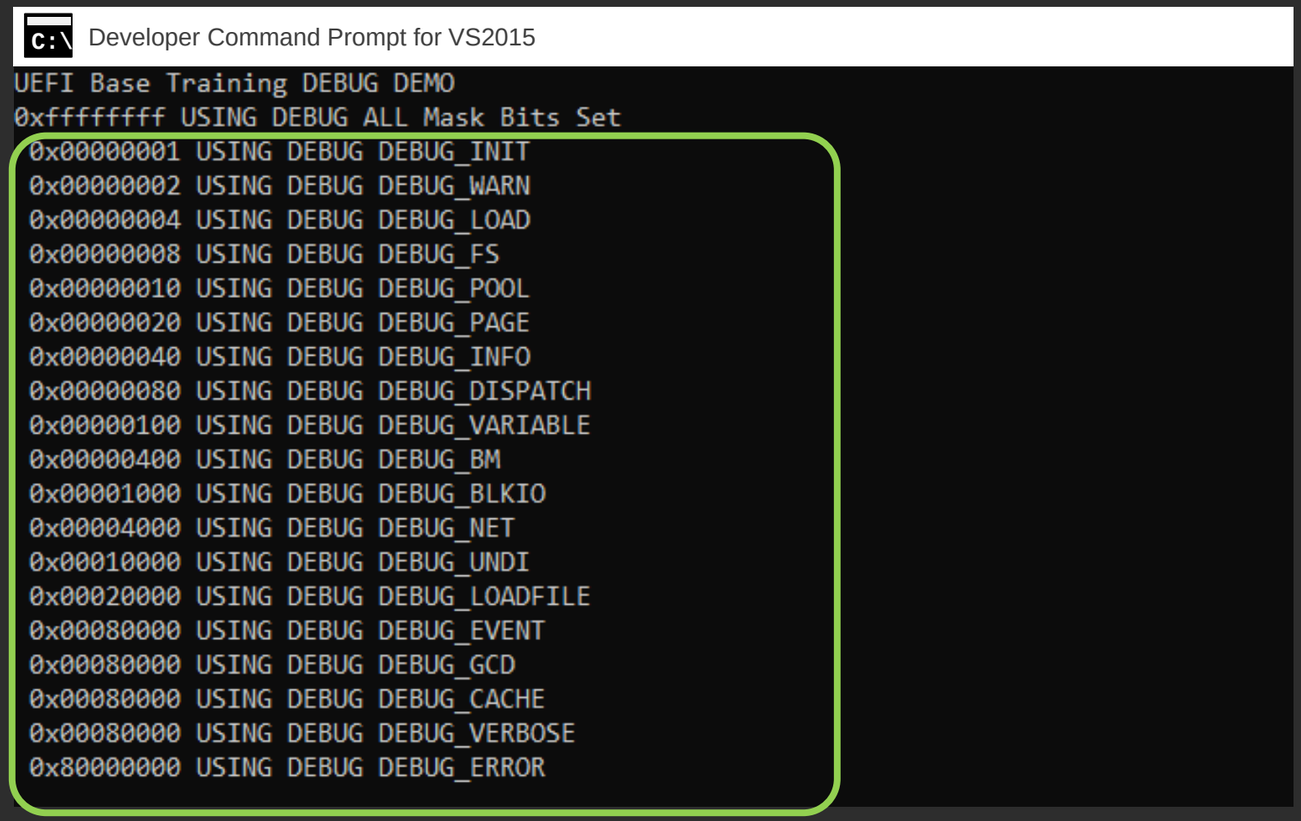
Demo: Build, Run and Test Result

Run the application from the shell

```
Shell> SampleApp
```

Check the VS Debug output

Visual Studio command prompt window output



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

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-
}
```

tianocore

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

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`

tianocore

DebugLib Instances (2)

Implementation

UefiDebugLibConOut UefiDebugLibStdErr

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

tianocore

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

DebugLib Instances (4)

Implementation

BaseDebugLibNull

- Resolves references
- Return Success

Instance to use to disable Debug

tianocore

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

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
 }
```

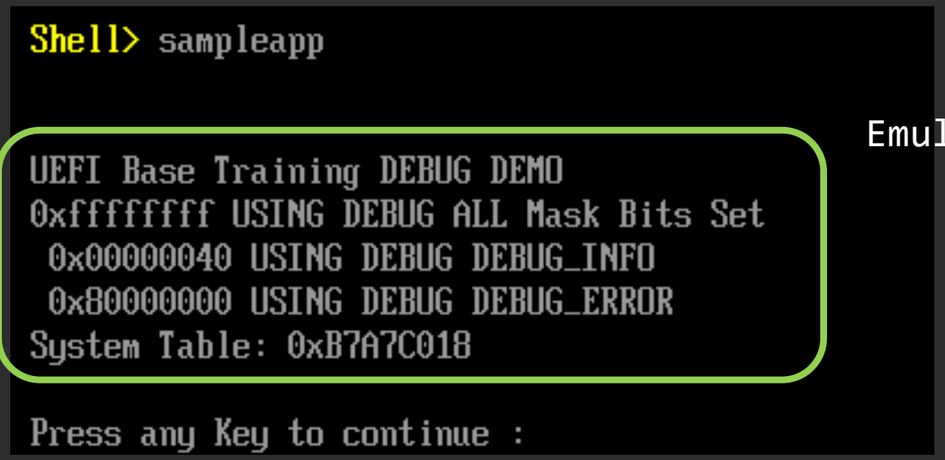
Demo: Debug Output in the Console

Application from the shell

```
Shell> SampleApp
```

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

Debug output to console



EmulatorPkg

tianocore

Demo: Debugging EDK II with VS Debugger

Debug with VS

SampleApp.c has an "ASSERT_EFI_ERROR" statement added

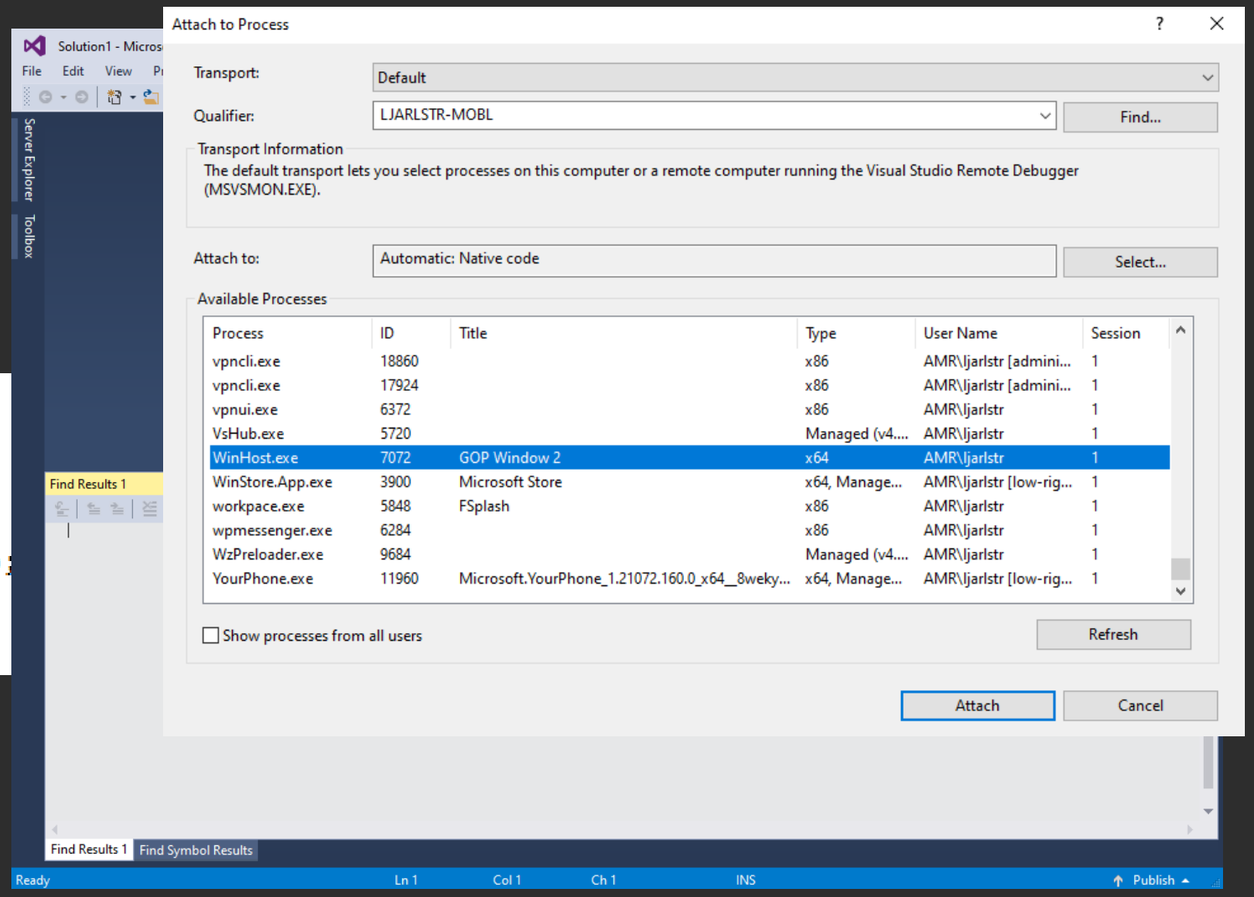
```
EFI_STATUS      Status;
Status = EFI_NO_RESPONSE;
       . . .
ASSERT_EFI_ERROR(Status);
```

```
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);
```

Visual Studio enable the WinHost for Debugging



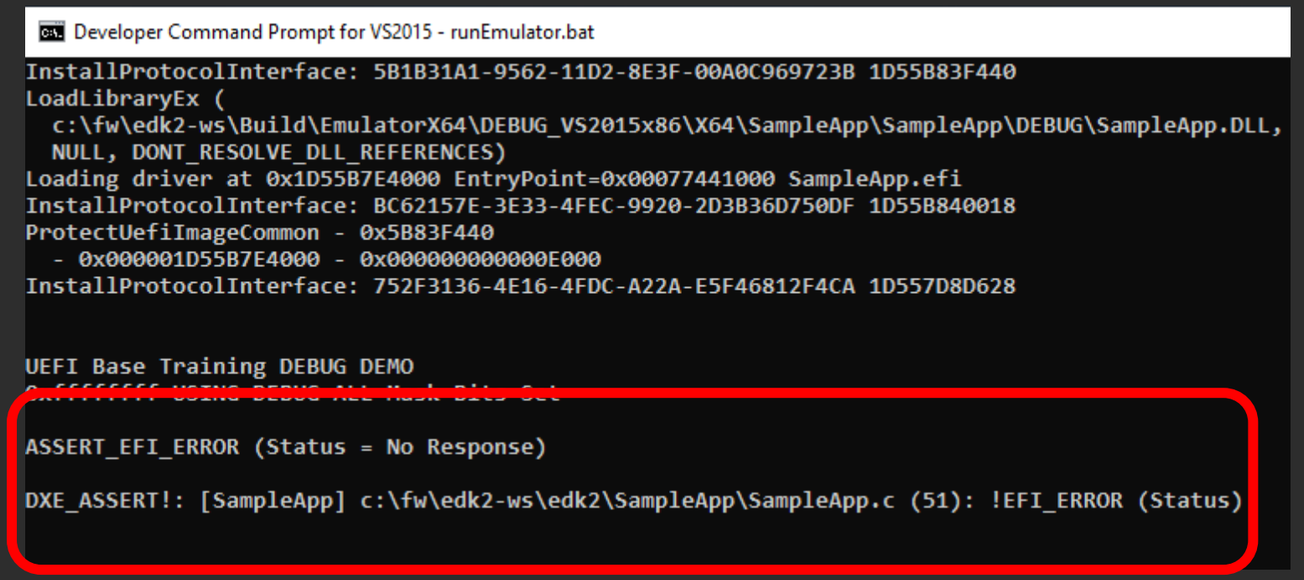
Demo: Debug with VS - ASSERT

Application from the shell

```
Shell> SampleApp
```

Assert in VS Command Prompt

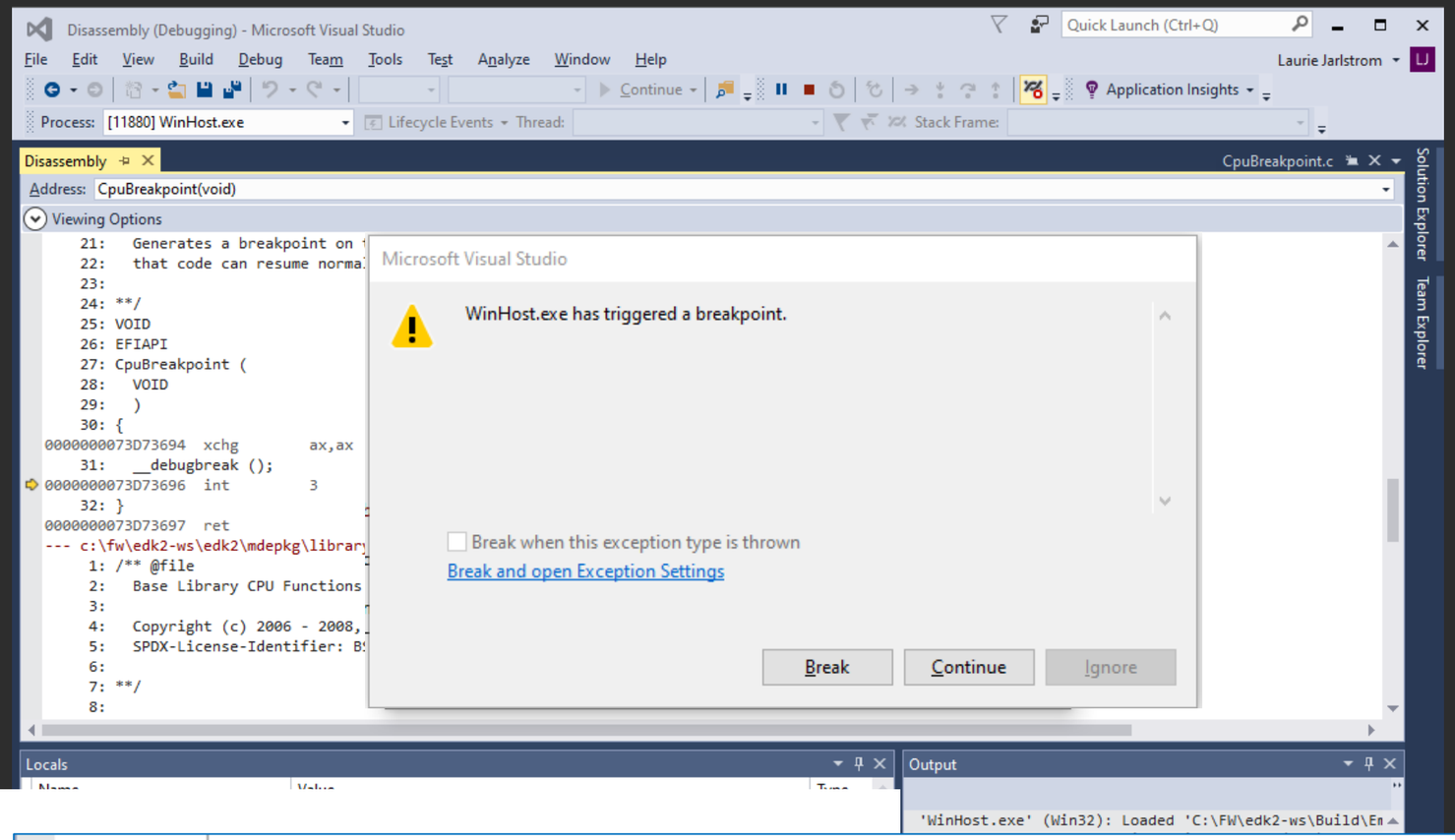
Visual Studio command prompt window output



Lab 5: Debug with VS - ASSERT

Windows* VS Debugger

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




```
24   **/
25   VOID
26   EFIAPI
27   CpuBreakpoint (
28     VOID
29     )
30   {
31     __debugbreak ();
32   }
33
```

“F10” - Step over 2-3 times

```
53
54     DEBUG((0xffffffff, "\n\nUEFI Base Training DEBUG DEMO\n"));
55     DEBUG((0xffffffff, "0xffffffff USING DEBUG ALL Mask Bits Set\n"));
56
57     ASSERT_EFI_ERROR(Status);  ≤1ms elapsed
58
59
60     DEBUG((DEBUG_INIT, " 0x%08x USING DEBUG DEBUG_INIT\n", (UINTN)(DEBUG_INIT)));
```

SampleApp.c

Demo: Debug with VS - CpuBreakpoint

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

```
CpuBreakpoint();
```

SampleApp.c

Miscellaneous Files (Global Scope)

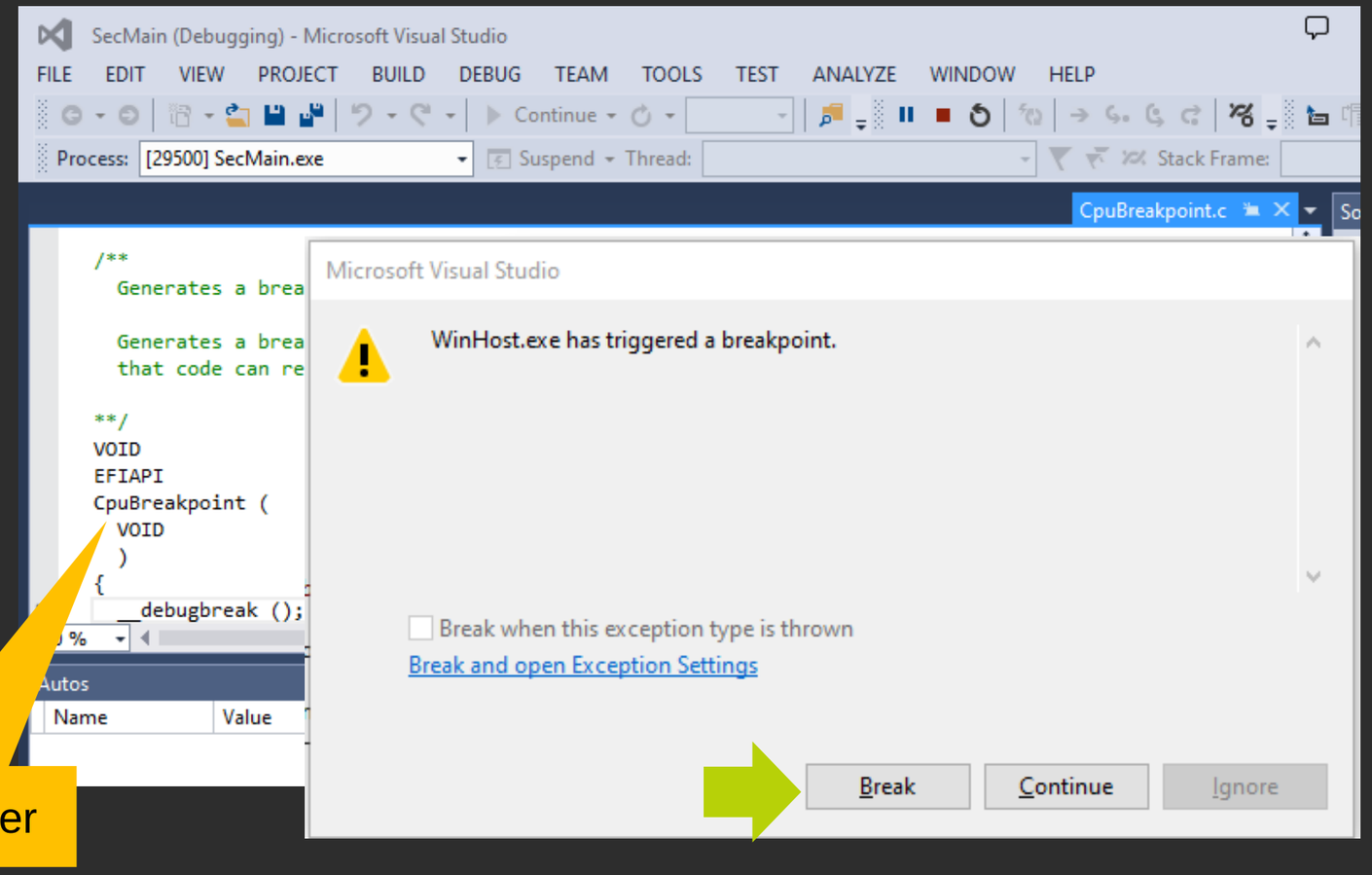
```
43      EFI_INPUT_KEY      Key;
44      DEBUG((EFI_D_INFO, "\r\n>>>>>[UefiMain] Entry point: 0x%p <<<<<<\r\n"
45
46      DEBUG((0xffffffff, "\n\nUEFI Base Training DEBUG DEMO\n"));
47      DEBUG((0xffffffff, "0xffffffff USING DEBUG ALL Mask Bits Set\n"));
48
49      //ASSERT_EFI_ERROR(0x8000000000000000);
50
51          CpuBreakpoint();
52
```

Demo: Debug with VS

Application from the shell

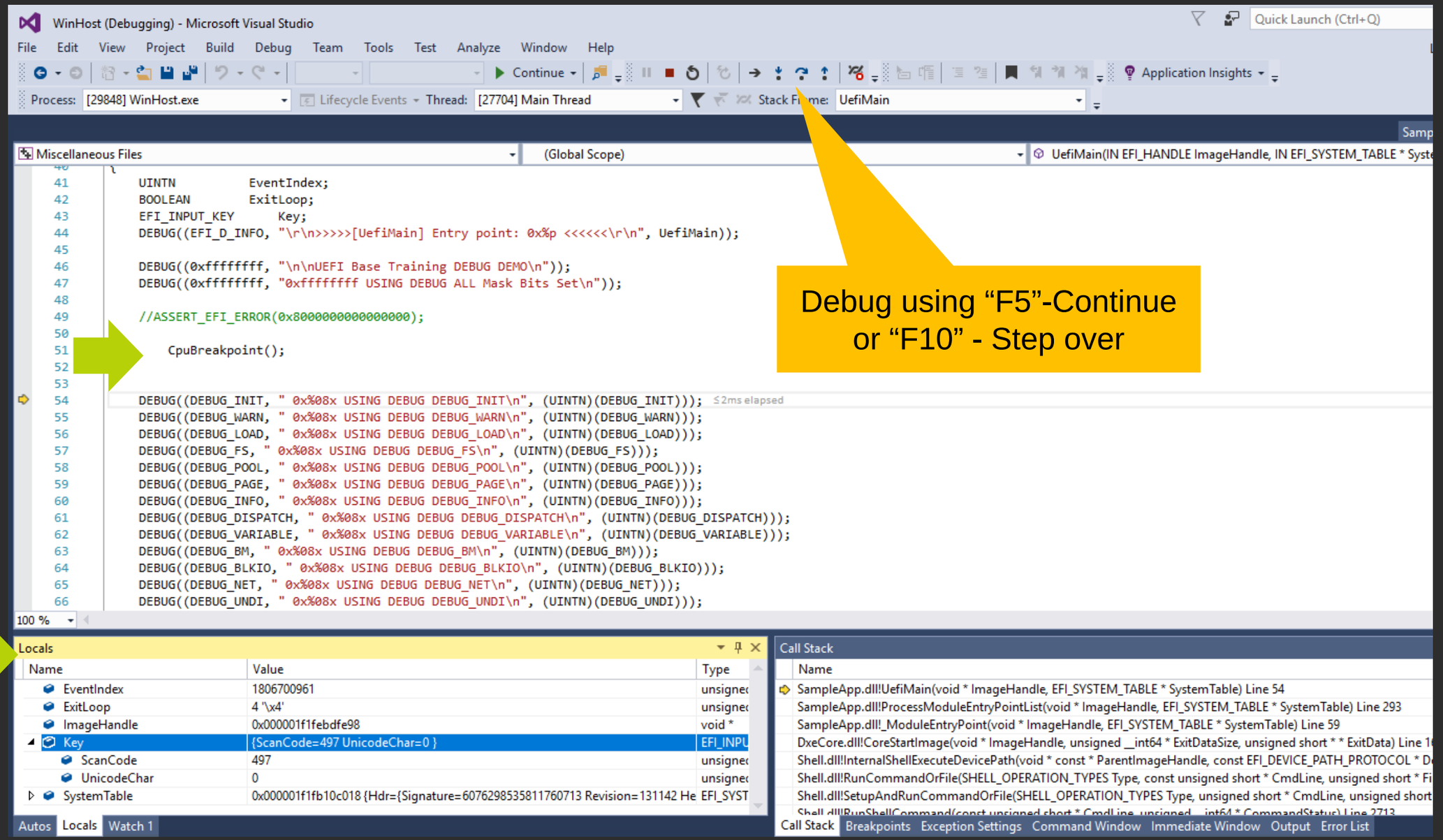
```
Shell> SampleApp
```

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



“F10” - Step over

Demo Windows Visual Studio Debugger



tianocore

Summary

- Debugging Overview
- 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

tianocore

Questions?

Return to Main Training Page

Return to Training Table of contents for next presentation link

tianocore

ACKNOWLEDGEMENTS

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

LINUX EXAMPLE

GDB and QEMU using `$ gdb --tui`

The GDB window will look similar to this