

EFI Framework Debugging

Overview

The Intel® Platform Innovation Framework for Extensible Firmware Interface (EFI), commonly known as the EFI Framework, is a new firmware architecture standard that defines a set of software interfaces and replaces the legacy BIOS found on traditional PC computers. This framework provides the kind of modularity, flexibility, and extensibility that were formerly unavailable with traditional BIOS. With EFI, BIOS developers can now write all their code in 'C', rather than assembly language. See Intel's website at <http://www.intel.com/technology/framework> or <http://www.tianocore.org> for more information on EFI Framework.

Along with this new firmware architecture and the 'C' code that implements it comes the need for source-level debugging. Arium's debugger, SourcePoint™ (versions 7.0 and later) for Intel and AMD processors offers native debug support for EFI Framework platforms. Users can set breakpoints, single step, view variables, see the call stack, and access all of the feature-rich functionality SourcePoint normally provides. This includes source-level debugging during the PEI, DXE, and OS Boot phases of EFI. Below is a set of instructions for setting up SourcePoint to debug the EFI Framework.

EFI Macros

Note: The macros described below are installed in the Macro\EFI sub-folder of the SourcePoint install path. Several of the EFI macro files contain directory paths to other macro files. If you move the macro files or change the current working directory in SourcePoint (via the 'cwd' command), you will need to update the macros files with the new locations.

SetupEFI.mac

After installing SourcePoint, run the SetupEFI.mac macro file located in the Macro\EFI directory. This creates four custom toolbar buttons and associates each with its corresponding EFI macro file.

- The Pre-EFI Initialization Modules (PEIMs) icon loads the symbol files for the PEI modules found in target memory.
- The Driver Execution Environments (DXEs) icon loads the symbol files for the DXE modules found in target memory.
- The Hand-Off Blocks (HOBs) icon displays a list of EFI HOBs found in target memory.
- The SysConfigTable icon displays the contents of the EFI system configuration table.



EFI_Functions.mac toolbar buttons

EFI_Functions.mac

The EFI_Functions.mac macro file contains the support functions. Inside this file, a constant definition called gEfiSystemTablePointerStart may need to be changed, depending on the location of the top of RAM in the target.

```
// DXE only:  
// Set this to the highest address to begin searching for the EFI system  
// table pointer. This should be your top of RAM. 0x20000000p works well  
// for most platforms.  
//  
define pointer gEfiSystemTablePointerStart = 0xF8000000P;
```

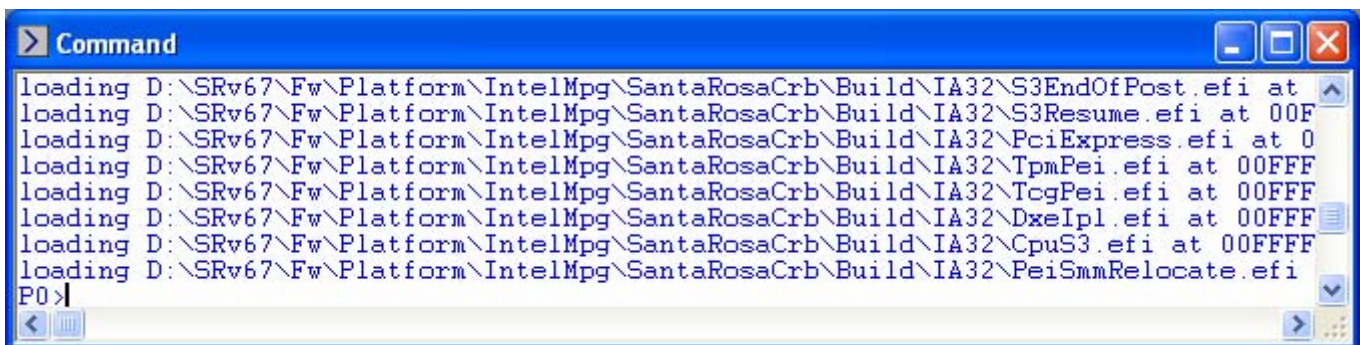
32-Bit or 64-Bit Builds

Depending on whether you build a 32-bit or 64-bit image, one parameter needs to be set in the EFI_Functions.mac file. Use Notepad or another editor to change the following text near the beginning of the file:

```
// DXE only:  
// If your platform has an EM64T processor, set this to 8 for 64-bit compiler builds  
// or set it to 4 for 32-bit compiler builds.  
//  
define ord4 gEM64TPointerSize = 4;
```

PEI Debugging

The PEI environment requires a specialized configuration of SourcePoint. PEI gets control shortly after target reset. PEI modules are dispatched and executed after cache RAM is mapped into system memory and the stack is initialized. Having a stack this early allows 'C' language code to execute, but a special memory map must be configured to take advantage of it.



```
Command  
loading D:\SRv67\Fw\Platform\IntelMpg\SantaRosaCrb\Build\IA32\S3EndOfPost.efi at  
loading D:\SRv67\Fw\Platform\IntelMpg\SantaRosaCrb\Build\IA32\S3Resume.efi at 00F  
loading D:\SRv67\Fw\Platform\IntelMpg\SantaRosaCrb\Build\IA32\PciExpress.efi at 0  
loading D:\SRv67\Fw\Platform\IntelMpg\SantaRosaCrb\Build\IA32\TpmPei.efi at 00FFF  
loading D:\SRv67\Fw\Platform\IntelMpg\SantaRosaCrb\Build\IA32\TcgPei.efi at 00FFF  
loading D:\SRv67\Fw\Platform\IntelMpg\SantaRosaCrb\Build\IA32\DxeIpl.efi at 00FFF  
loading D:\SRv67\Fw\Platform\IntelMpg\SantaRosaCrb\Build\IA32\CpuS3.efi at 00FFFF  
loading D:\SRv67\Fw\Platform\IntelMpg\SantaRosaCrb\Build\IA32\PeiSmmRelocate.efi  
P0>|
```

Command window after running PEIMs macro function

To configure SourcePoint for source-level debugging of PEI code, follow these steps.

1. Optional: Select **Options|Target Configuration|Memory Map** from the menu, and set it similar to the following (your system may vary depending on your memory map):

Start	End	Type
00000000P	000FFFFFFP	DRAM
FEF00000P	FFEFFFFFFP	SRAM
FFF00000P	FFFFFFFFFP	FLASH

The first entry in the table designates the first 1MB of system memory. The middle entry designates the location of the cache RAM mapped into system memory. The third entry designates the firmware ROM.

2. Select **Options|Preferences|Breakpoints** and set the default breakpoint type to **Processor**.
3. Click the PEIMs toolbar icon to load the PEIM symbols.

The screenshot shows a code window titled "Code P0*: Tracking IP: d:\srv67\fw\platform\generic\monostatusc...". The code is as follows:

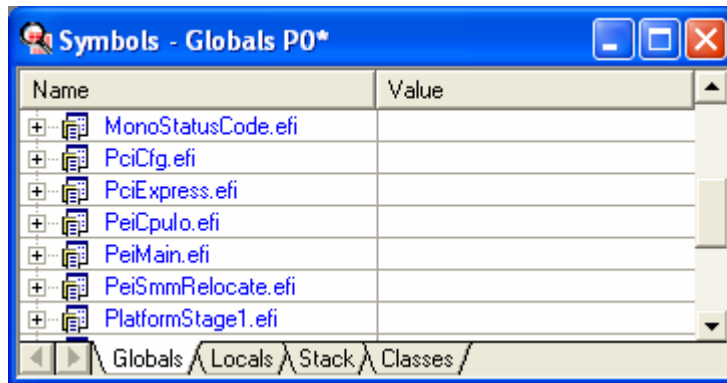
```

2222
2223     for (; *OutputString != 0; OutputString++) {
2224     DebugSerialWrite (*OutputString);
2225     }
2226 }
2227
2228 EFI_STATUS
2229 EFI_API
2230 SerialReportStatusCode (
2231     IN EFI_PEI_SERVICES          **PeiServices,
2232     IN EFI_STATUS_CODE_TYPE     CodeType,
2233     IN EFI_STATUS_CODE_VALUE    Value,
2234     IN UINT32                   Instance,
2235     IN EFI_GUID                 * CallerId,
2236     IN EFI_STATUS_CODE_DATA     * Data OPTIONAL
2237 )
2238 /*++
2239
2240 Routine Description:
2241
2242 Provide a serial port print
2243

```

The window also features a toolbar at the bottom with the following controls: a dropdown menu showing "0010:FFFD65C7", a "Source" dropdown, "Go Cursor", "Set Break", a checked "Track IP" checkbox, and a "View IP" button.

Code window after loading PEIM modules



Symbols window after loading PEIM modules

DXE Debugging

Once system RAM is initialized and the PEI phase completes, the DXE environment is entered. This is less specialized than PEI; nevertheless, it requires a few SourcePoint parameters to be set.

To configure SourcePoint for source-level debugging of DXE code, follow these steps:

1. Optional: Select **Options|Target Configuration|Memory Map** from the menu, and set it up as described below.

The entry in the table designates the entire 4GB address space as DRAM.

Start	End	Type
00000000P	FFFFFFFFP	DRAM

2. Run the target to the EFI shell, or as far as it will go in DXE.
3. Stop the target.
4. Click the DXEs toolbar icon to load the DXE symbols.
5. Browse the source code files using the **Symbols** window and set breakpoints in your code.
6. Reset the target and go until you hit a breakpoint.

```

Code P0*: (32-bit) Tracking IP: 0010:00000000 - 0010:FFFFFFFF
0010:1EE8E535 POP      ECX
0010:1EE8E536 JNE      short ptr CpuIoServiceRead+9f
318          }
319          break;
0010:1EE8E538 JMP      short ptr CpuIoServiceRead+d1
309          case EfiCpuIoWidthUint8:
310          for (; Count > 0; Count--, Buffer.buf += OutS
0010:1EE8E53A MOV      EBX, dword ptr [EBP]+18
0010:1EE8E53D TEST     EBX, EBX
0010:1EE8E53F JBE      short ptr CpuIoServiceRead+d1
311          *Buffer.ui8 = CpuIoRead8 ((UINT16) Address:
0010:1EE8E541 PUSH     dword ptr [EBP]+14
0010:1EE8E544 CALL     near32 ptr CpuIoRead8
→ 0010:1EE8E549 ADD      dword ptr [EBP]+14, EDI
0010:1EE8E54C MOV      byte ptr [ESI], AL
0010:1EE8E54E ADD      ESI, dword ptr [EBP]+1c
0010:1EE8E551 DEC      EBX
0010:1EE8E552 POP      ECX
0010:1EE8E553 JNE      short ptr CpuIoServiceRead+bd
329          }
330
331          return EFI_SUCCESS;
0010:1EE8E555 XOR      EAX, EAX
0010:1EE8E557 POP      EDI
0010:1EE8E558 POP      ESI
0010:1EE8E559 POP      EBX
332          }
0010:1EE8E55A POP      EBP
0010:1EE8E55B RETN

```

DXE Code window

HOBs

Open the **Command** window, and then click the HOBs toolbar icon to display the hand-off blocks on the target.

```

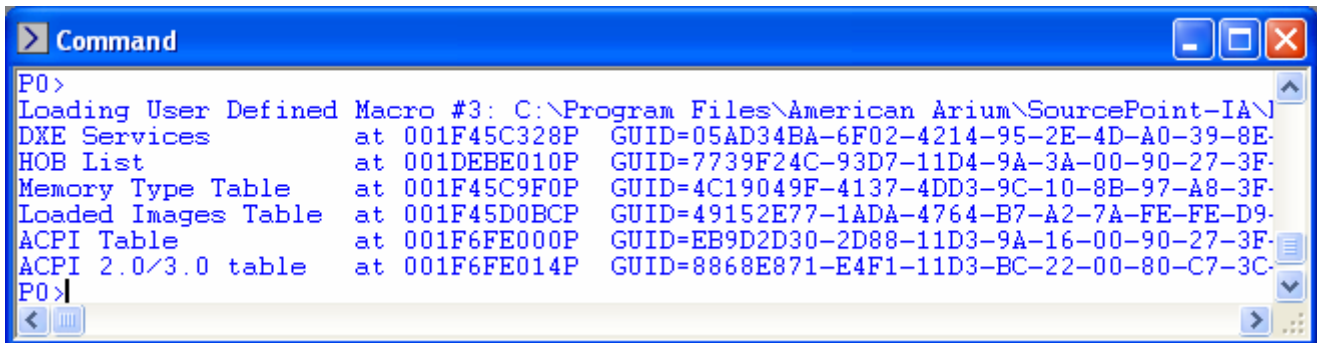
Command
HOB Resource descriptor at 001DEBEC08P
Resource type      0x0 (system memory)
Attributes         0x3C03
                  Present
                  Initialized
                  Uncacheable
                  Write-combinable
                  Write-through cacheable
                  Write-back cacheable
Base address       0x0000000000000000
Length            0x000000000000A000
HOB Resource descriptor at 001DEBEC38P
Resource type      0x5 (reserved memory)
Attributes         0x0

```

Example of HOB display

System Configuration Table

Open the **Command** window, and then click the SysConfigTable toolbar icon to display the contents of the EFI system configuration table on the target.



```
P0>
Loading User Defined Macro #3: C:\Program Files\American Arium\SourcePoint-IA\
DXE Services          at 001F45C328P  GUID=05AD34BA-6F02-4214-95-2E-4D-A0-39-8E-
HOB List              at 001DEBE010P  GUID=7739F24C-93D7-11D4-9A-3A-00-90-27-3F-
Memory Type Table    at 001F45C9F0P  GUID=4C19049F-4137-4DD3-9C-10-8B-97-A8-3F-
Loaded Images Table  at 001F45D0BCP  GUID=49152E77-1ADA-4764-B7-A2-7A-FE-FE-D9-
ACPI Table           at 001F6FE000P  GUID=EB9D2D30-2D88-11D3-9A-16-00-90-27-3F-
ACPI 2.0/3.0 table  at 001F6FE014P  GUID=8868E871-E4F1-11D3-BC-22-00-80-C7-3C-
P0>
```

Example of System Configuration Table

Notes

1. Multiple Project Files Tip

It may be convenient to create one project file for PEI debugging and another project file for DXE debugging. That way you can quickly pull up the SourcePoint configuration you need for that debugging session.

2. DXE Debugging Tip

To stop the target and load symbols just before a DXE module is dispatched, open the **Symbols** window, choose the **Globals** tab, and drill down to:

```
program:  DXEMAIN.efi
module:   image (image.c)
function: CoreStartImage()
```

Right click on **CoreStartImage** and select **Open Code Window** from the pop-up menu.

Set a processor breakpoint in **CoreStartImage()** where **ImageEntryPoint()** is called. This hits before each DXE module is dispatched, but afterwards its entry is placed in tables. Each time you hit this breakpoint, click the DXEs toolbar icon to load the DXE symbols.

Now you can load symbols just before your DXE module runs instead of running to the EFI shell, then loading symbols, then resetting the target, then running to your breakpoint.

3. Watchdog Timer on Intel Platforms

Some motherboards with Intel processors have a TCO timer that will assert RESET independent of the emulator. See the Arium application note titled, "Disabling the TCO Timer in an Intel I/O Controller Hub" for details. Resetting the target from SourcePoint can cause a **Target state undefined** error message to appear because the timer asserts RESET and confuses the emulator. The solution to this problem is to configure the ICH_TCO_Timer_Disable.mac macro to run at every target reset.

4. The EFI firmware on the target contains strings that hold the paths to the program symbol files on your hard drive. SourcePoint macros read target memory, find these strings, then load the symbol files specified in these paths. The symbol files must be located in the path specified in the EFI firmware.

For example, one path might look like this:

```
"Z:\Platform\IntelSsg\D845GRG\Build\IA32\DxeMain.efi"
```

This architecture, defined by Intel, presents a requirement for EFI debugging. You must have the EFI

symbol files on the host computer in the same directories as specified in the firmware on the target. This should not be a problem if you build the EFI firmware on the same host from which you run SourcePoint.

If the drive letter or path doesn't match exactly, you can use the 'subst' command from the Windows command prompt to map a drive letter to a desired path.

