# Protocols Debugger Support Test

## EFI\_DEBUG\_SUPPORT\_PROTOCOL Test

Reference Document:

*UEFI Specification,* EFI\_DEBUG\_SUPPORT\_PROTOCOL Section.

### GetMaximumProcessorIndex()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.12.1.1.1 | 0x2ac7927c, 0xd9df, 0x4c32, 0x87, 0xb4, 0xad, 0x0a, 0xc4, 0xbb, 0xd5, 0x92 | EFI\_DEBUG\_SUPPORT\_PROTOCOL.GetMaximumProcessorIndex - Invokes GetMaximumProcessorIndex() returns EFI\_SUCCESS and the out parameter contains a UINTN value. | Call GetMaximumProcessorIndex(). It should return EFI\_SUCCESS and the out parameter should contain a UINTN value. |

### RegisterPeriodicCallback()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.12.1.2.1 | 0x1e43071e, 0xa00d, 0x46eb, 0xbd, 0xdd, 0x8f, 0x54, 0x22, 0xef, 0x24, 0x30 | EFI\_DEBUG\_SUPPORT\_PROTOCOL.RegisterPeriodicCallback - Invokes RegisterPeriodicCallback() installs an interrupt handler function and returns EFI\_SUCCESS. | Call RegisterPeriodicCallback() with a valid interrupt handler function. The return code should be EFI\_SUCCESS. |
| 5.12.1.2.2 | 0x792e517a, 0xf006, 0x46e6, 0xb3, 0x19, 0xc0, 0xc8, 0x7e, 0x43, 0x8b, 0x32 | EFI\_DEBUG\_SUPPORT\_PROTOCOL.RegisterPeriodicCallback - The SYSTEM\_TIMER\_VECTOR interrupt invokes the PeriodicCallback(). | Wait for the PeriodicCallback() to be invoked by the SYSTEM\_TIMER\_VECTOR interrupt. The PeriodicCallback() should be invoked. |
| 5.12.1.2.3 | 0xef21928d, 0xa7c3, 0x4c92, 0xaa, 0x22, 0x97, 0xc3, 0x3d, 0x4d, 0xd2, 0x00 | EFI\_DEBUG\_SUPPORT\_PROTOCOL.RegisterPeriodicCallback - The PeriodicCallback() is invoked earlier than the time event callback function. | Create a time event and register a callback function for it with less time than the machine clock. Wait for two callback functions to be invoked. The PeriodicCallback() should be invoked earlier than the time event callback function. |
| 5.12.1.2.4 | 0x9f3d4d83, 0xee41, 0x41dd, 0x83, 0x13, 0x6c, 0xc0, 0x59, 0x7f, 0x22, 0x21 | EFI\_DEBUG\_SUPPORT\_PROTOCOL.RegisterPeriodicCallback - Invokes RegisterPeriodicCallback() installs another interrupt handler function and returns EFI\_ALREADY\_STARTED. | Call RegisterPeriodicCallback() with a valid interrupt handler function. The return code should be EFI\_ALREADY\_STARTED. |
| 5.12.1.2.5 | 0x29778e36, 0x09ad, 0x47db, 0x82, 0x4c, 0x5b, 0x46, 0x25, 0xd0, 0xe5, 0xb4 | EFI\_DEBUG\_SUPPORT\_PROTOCOL.RegisterPeriodicCallback - Invokes RegisterPeriodicCallback() unstalls the interrupt handler function and returns EFI\_SUCCESS. | Call RegisterPeriodicCallback() with a NULL interrupt handler function. The return code should be EFI\_SUCCESS. |
| 5.12.1.2.6 | 0xc34688c4, 0x9f84, 0x40a7, 0x90, 0x84, 0xe6, 0x5e, 0x2c, 0xbe, 0xae, 0x45 | EFI\_DEBUG\_SUPPORT\_PROTOCOL.RegisterPeriodicCallback - The PeriodicCallback() is not invoked after the SYSTEM\_TIMER\_VECTOR interrupt. | Wait for the SYSTEM\_TIMER\_VECTOR interrupt. The PeriodicCallback() should not be invoked. |

### RegisterExceptionCallback()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.12.1.3.1 | 0x20bc4ac1, 0x8958, 0x446a, 0x8b, 0x5f, 0x27, 0xb3, 0xcc, 0x77, 0x41, 0x06 | EFI\_DEBUG\_SUPPORT\_PROTOCOL.RegisterExceptionCallback - Invokes RegisterExceptionCallback() installs an interrupt handler function. | Call RegisterExceptionCallback() with a valid InterrruptHandler function, the exception type is EXCEPT\_IA32\_BREAKPOINT. The return code should be EFI\_SUCCESS. |
| 5.12.1.3.2 | 0xfbfa47e8, 0xbd32, 0x4f81, 0x89, 0x38, 0xb7, 0x36, 0x47, 0x08, 0xa2, 0xb9 | EFI\_DEBUG\_SUPPORT\_PROTOCOL.RegisterExceptionCallback - Calling INT3 invokes the interrupt handler function. | Use “INT 3” instruction to invokes the interrupt. After “INT 3” is called, the interrupt handler function should be invoked. |
| 5.12.1.3.3 | 0x14362c36, 0xf284, 0x4a95, 0xab, 0x1b, 0x3b, 0x67, 0xa9, 0x6e, 0x1d, 0xe8 | EFI\_DEBUG\_SUPPORT\_PROTOCOL.RegisterExceptionCallback - Invokes RegisterPeriodicCallback() installs the Periodic interrupt handler function and two callback functions are invoked. | Call RegisterPeriodicCallback() with a valid InterrruptHandler function.Use “INT 3” instruction to invokes the Exception callback function, and wait for the periodic callback function to be invoked. The return code of RegisterPeriodicCallback() should be EFI\_SUCCESS.Two callback functions should be invoked successfully. |
| 5.12.1.3.4 | 0x0cf314a2, 0xfe51, 0x4093, 0xb4, 0x22, 0x9f, 0x4a, 0x90, 0x98, 0xd2, 0x89 | EFI\_DEBUG\_SUPPORT\_PROTOCOL.RegisterExceptionCallback - Invokes RegisterExceptionCallback()installs another interrupt handler function. | Call RegisterExceptionCallback() with a valid InterrruptHandler function. The return code should be EFI\_ALREADY\_STARTED. |
| 5.12.1.3.5 | 0x28e232bd, 0xfe72, 0x4963, 0xb3, 0x33, 0x1e, 0x83, 0x61, 0x5e, 0x1e, 0x2e | EFI\_DEBUG\_SUPPORT\_PROTOCOL.RegisterExceptionCallback - Invokes RegisterExceptionCallback()uninstalls the interrupt handler function. | Call RegisterExceptionCallback() with NULL InterrruptHandler function. The return code should be EFI\_SUCCESS. |
| 5.12.1.3.6 | 0x59efd2fb, 0x2f7d, 0x4535, 0xa2, 0x1c, 0x39, 0x25, 0xcb, 0xb3, 0x0b, 0x87 | EFI\_DEBUG\_SUPPORT\_PROTOCOL.RegisterExceptionCallback - Using “INT 3“ instruction does not invokes the previously installed (but now uninstalled) interrupt handler function. | Use “INT 3“ instruction to invokes the interrupt. After “INT 3” is called, the previously installed (but now uninstalled) interrupt handler function should not be invoked. |

### InvalidateInstructionCache()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.12.1.4.1 | 0x41c3bc2c, 0xf066, 0x4272, 0xac, 0xa7, 0xb9, 0x48, 0x9f, 0xac, 0x94, 0x2b | EFI\_DEBUG\_SUPPORT\_PROTOCOL.InvalidateInstructionCache - Invokes InvalidateInstructionCache() returns EFI\_SUCCESS, verifying interface correctness. | Call InvalidateInstructionCache().The return code should be EFI\_SUCCESS. |

### Isa

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.12.1.5.1 | 0x701d9223, 0x1123, 0x40a2, 0xa8, 0x81, 0x5f, 0xd6, 0x68, 0xeb, 0x32, 0x87 | EFI\_DEBUG\_SUPPORT\_PROTOCOL.Isa – The instruction is IA32, IPF, or EBC. | Get the Isa value, it should be IA32 (0x014C), IPF (0x0200), or EBC (0xEBC). |

## EFI\_DEBUGPORT\_PROTOCOL Test

Reference Document:

*UEFI Specification*, EFI\_DEBUGPORT\_PROTOCOL Section.

### Reset()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.12.2.1.1 | 0x6aca7c62, 0x7bbe, 0x4d1b, 0x9c, 0x8a, 0xc7, 0x7a, 0x6c, 0x68, 0x74, 0x76 | EFI\_DEBUGPORT\_PROTOCOL.Reset - Invokes Reset() returns EFI\_SUCCESS, verifying interface correctness within test case. | Call Reset().It should return EFI\_SUCCESS. |

### Write()

No automatic test is designed to verify this function.

### Read()

No automatic test is designed to verify this function.

### Poll()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.12.2.4.1 | 0x4bf087b2, 0xe914, 0x4056, 0x8e, 0x1a, 0x25, 0xf0, 0x13, 0x54, 0x31, 0x26 | EFI\_DEBUGPORT\_PROTOCOL.Poll - Calling Poll()when the debug port has data returns EFI\_SUCCESS. | Call Write() to send data to the debug port. Call Poll() to check the debug port to see if any data is available to be read. The return code of Poll() should be EFI\_SUCCESS. |
| 5.12.2.4.2 | 0x838a1da2, 0x9640, 0x47f3, 0xba, 0xc1, 0x39, 0x26, 0xf3, 0x1d, 0x00, 0xc2 | EFI\_DEBUGPORT\_PROTOCOL.Poll - Calling Poll() when the debug port does not have data returns EFI\_NOT\_READY. | Call Reset() to reset the debug port. Call Poll() to check the debug port to see if any data is available to be read. The return code of Poll() should be EFI\_NOT\_READY. |