# Protocols Console Support Test

## EFI\_SIMPLE\_ TEXT\_INPUT\_PROTOCOL Test

Reference Document:

*UEFI Specification*, EFI\_SIMPLE\_TEXT\_INPUT\_PROTOCOL Section.

### Reset()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.1.1.1 | 0x7cb5f8dd, 0x7346, 0x484b, 0xb1, 0xb3, 0xa6, 0x46, 0x69, 0x6d, 0xea, 0xe7 | EFI\_SIMPLE\_TEXT\_INPUT\_PROTOCOL.Reset – Reset() returns EFI\_SUCCESS with ExtendedVerification as FALSE. | 1. Call Reset() with ExtendedVerification as FALSE. The return status should be EFI\_SUCCESS. |
| 5.6.1.1.2 | 0x6fc31add, 0xf34b, 0x4b56, 0x9b, 0xa6, 0x36, 0xb2, 0x7c, 0xbe, 0xf5, 0xa2 | EFI\_SIMPLE\_TEXT\_INPUT\_PROTOCOL.Reset – ReadKeyStroke() returns EFI\_NOT\_READY when there is no key has been stroked. | 1. Call Reset() with ExtendedVerification as FALSE.  2. After Reset(), do not stroke any key, and call ReadKeyStroke(). The return code should be EFI\_NOT\_READY |
| 5.6.1.1.3 | 0x8da56db6, 0xd7df, 0x4029, 0xba, 0x98, 0x37, 0x46, 0x0b, 0x21, 0x0e, 0x3b | EFI\_SIMPLE\_TEXT\_INPUT\_PROTOCOL.Reset – Reset() returns EFI\_SUCCESS with ExtendedVerification as TRUE. | 1. Call Reset() with ExtendedVerification as TRUE. The return status should be EFI\_SUCCESS. |
| 5.6.1.1.4 | 0x3d51b174, 0x59f8, 0x44bc, 0xb7, 0xf7, 0x9a, 0x11, 0x2c, 0x51, 0x82, 0xa1 | EFI\_SIMPLE\_TEXT\_INPUT\_PROTOCOL.Reset – ReadKeyStroke() returns EFI\_NOT\_READY when there is no key that has been stroked. | 1. Call Reset() with ExtendedVerification as TRUE.  2. After Reset(), do not stroke any key, and call ReadKeyStroke(). The return code should be EFI\_NOT\_READY |

### ReadKeyStroke()

No automatic test case is designed to verify this function.

## EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL Test

Reference Document:

*UEFI Specification*, EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL Section.

UEFI 2.1 Specification, Section 11.3.

### Reset()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.2.1.1 | 0xecaf43c6, 0x6b77, 0x413a, 0x89, 0x8f, 0x28, 0x0e, 0x92, 0x5f, 0xf9, 0x43 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.Reset – Reset() without Extended Verification Mode | 1. Call Reset() with ExtendedVerification as FALSE.  2. Check cursor position. It should be (0,0). |
| 5.6.2.1.2 | 0xc40bba44, 0xcfa3, 0x4494, 0xaf, 0xa5, 0xfa, 0x2f, 0x78, 0xcb, 0x20, 0x20 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.Reset – Reset() without Extended Verification Mode returns EFI\_SUCCESS | 1. Call Reset() with ExtendedVerification as FALSE. The return code should be EFI\_SUCCESS |
| 5.6.2.1.3 | 0x51267bf4, 0x7b3e, 0x46fd, 0xac, 0x6c, 0xff, 0x8e, 0x54, 0x61, 0xd1, 0x7f | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.Reset – Reset() with Extended Verification Mode | 1. Call Reset() with ExtendedVerification as TRUE.  2. Check cursor position. It should be (0,0). |
| 5.6.2.1.4 | 0x1771a342, 0xbbc3, 0x43da, 0x91, 0x4d, 0x7d, 0x59, 0xb7, 0xd8, 0x86, 0x2e | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.Reset – Reset() with Extended Verification Mode returns EFI\_SUCCESS | 1. Call Reset() with ExtendedVerification as TRUE. The return code should be EFI\_SUCCESS |

### OutputString()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.2.2.1 | 0x3e706c2f, 0xc7ee, 0x43de, 0x8f, 0xe7, 0x39, 0x81, 0x33, 0x11, 0x7d, 0x9b | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.OutputString – OutputString() with normal Unicode String. Cursor value in Mode should be assigned to the right position. | 1. Call OutputString() with normal Unicode String.  2. Check cursor position. It should be at the end of the string. In addition, other attributes of output mode remain unchanged. |
| 5.6.2.2.2 | 0xb7c77060, 0xbd1e, 0x4607, 0x85, 0x41, 0xdc, 0xf5, 0x08, 0xe3, 0xff, 0xd4 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.OutputString – OutputString() with normal Unicode string returns EFI\_SUCCESS | 1. Call OutputString() with normal Unicode String. The return code should be EFI\_SUCCESS |
| 5.6.2.2.3 | 0xf3f07bdb, 0x683d, 0x448f, 0xa5, 0x4a, 0xb5, 0x61, 0xf9, 0x86, 0x95, 0xb5 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.OutputString – OutputString() with very long Unicode String. Cursor value in Mode should be assigned to the right position. | 1. Call OutputString() with very long Unicode String.  2. Check cursor position. It should be at the end of the string. In addition, other attributes of output mode remain unchanged. |
| 5.6.2.2.4 | 0xcefd060c, 0x9ed5, 0x4862, 0x96, 0x75, 0xda, 0x26, 0x3b, 0xdc, 0x35, 0x3a | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.OutputString – OutputString() with very long Unicode String returns EFI\_SUCCESS | 1. Call OutputString() with very long Unicode String. The return code should be EFI\_SUCCESS |
| 5.6.2.2.5 | 0x722925c0, 0xf84a, 0x4aa0, 0x9d, 0xe8, 0x04, 0x03, 0x70, 0xe0, 0x69, 0x0f | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.OutputString – OutputString() with cursor control Unicode String. Cursor value in Mode should be assigned to the right position. | 1. Call OutputString() with Drawing Unicode String.  2. Check cursor position. It should be at the end of the string. In addition, other attributes of output mode remain unchanged. |
| 5.6.2.2.6 | 0x6fce5c66, 0xd273, 0x446d, 0x88, 0x54, 0x94, 0x7b, 0x6c, 0xd4, 0xa3, 0x96 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.OutputString – OutputString() with Drawing Unicode String returns EFI\_SUCCESS | 1. Call OutputString() with Drawing Unicode String. The return code should be EFI\_SUCCESS |
| 5.6.2.2.7 | 0xae266668, 0xa3ef, 0x4930, 0x85, 0x64, 0x55, 0x9f, 0x9e, 0x96, 0x14, 0x6b | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.OutputString – OutputString() with cursor control Unicode String. Cursor value in Mode should be assigned to the right position. | 1. Call OutputString() with cursor control Unicode String.  2. Check cursor position. It should be at the appointed postion. In addition, other attributes of output mode remain unchanged. |
| 5.6.2.2.8 | 0x2e40bcfe, 0x7713, 0x4ab1, 0x99, 0x5c, 0xe0, 0x8b, 0x2d, 0xdc, 0x2b, 0x60 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.OutputString – OutputString() with cursor control Unicode String returns EFI\_SUCCESS | 1. Call OutputString() with cursor control Unicode String. The return code should be EFI\_SUCCESS |

### TestString()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.2.3.1 | 0x0317202b, 0x4c09, 0x4f09, 0xa8, 0x9e, 0x17, 0x91, 0x7d, 0x0b, 0xb5, 0x6c | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.TestString – TestString() with cursor control Unicode String. Cursor value in Mode should be assigned to the right position. | 1. Call TestString() with normal Unicode String.  2. Check cursor position. It should be at the end of the string. In addition, other attributes of output mode remain unchanged. |
| 5.6.2.3.2 | 0x92609750, 0x7965, 0x4e08, 0xae, 0xaf, 0xb1, 0xec, 0xa3, 0x61, 0x63, 0x66 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.TestString – TestString() with normal Unicode string returns EFI\_SUCCESS | 1. Call TestString() with normal Unicode String Mode value. The return code should be EFI\_SUCCESS |

### QueryMode()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.2.4.1 | 0x26d95327, 0x008c, 0x4ca1, 0xb6, 0x75, 0x9d, 0x86, 0x20, 0xdf, 0x73, 0x19 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.QueryMode – QueryMode() with supported ModeNumber value remains other attributes unchanged | 1. Call QueryMode() with supported ModeNumber value. Other attributes should remain unchanged. |
| 5.6.2.4.2 | 0xf2b8054e, 0xcfa7, 0x4fcd, 0x9e, 0x6c, 0xc6, 0x07, 0xbe, 0x62, 0xff, 0x27 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.QueryMode – QueryMode() with supported ModeNumber value returns EFI\_SUCCESS | 1. Call QueryMode() with supported ModeNumber value. The return code should be EFI\_SUCCESS |
| 5.6.2.4.3 | 0x4b5c620e, 0x0e2f, 0x4c19, 0xa2, 0x41, 0x25, 0xbd, 0x47, 0x67, 0xbf, 0x3e | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.QueryMode – QueryMode() with unsupported ModeNumber value returns EFI\_UNSUPPORTED | 1. Call QueryMode() with each ModeNumber value less than **MaxMode**. If ModeNumber #1 (80\*50) is unsupported, the return code should be EFI\_UNSUPPORTED |
| 5.6.2.4.4 | 0x5c444cd8, 0x3dce, 0x4be7, 0xb5, 0xcd, 0x39, 0x38, 0xd5, 0x04, 0xac, 0x95 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.QueryMode – ModeNumber #0 is supported and the dimension is 80 \* 25 | 1. Call QueryMode() with each **Mode** value less than **MaxMode**. ModeNumber #0 should be supported and the dimension is 80 \* 25 |
| 5.6.2.4.5 | 0x3b069c23, 0xde80, 0x4eb9, 0x86, 0x57, 0x48, 0x0f, 0x63, 0x81, 0x6c, 0x53 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.QueryMode – If ModeNumber #1 is supported, the dimension is 80 \* 50 | 1. Call QueryMode() with each ModeNumber value less than **MaxMode**. If ModeNumber #1 is supported, the dimension should be 80 \* 50 |
| 5.6.2.4.6 | 0x891cb899, 0xc05e, 0x4160, 0xa9, 0x8c, 0x06, 0x04, 0xc4, 0x0a, 0x44, 0x48 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.QueryMode – QueryMode() with Invalid ModeNumber Fields, EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.Mode does not change before and after | 1. Call QueryMode() with Invalid Mode Fields beyond **MaxMode**. EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.Mode should not change before and after |
| 5.6.2.4.7 | 0x8f0b6ebe, 0xaa65, 0x4aa4, 0x8c, 0xfc, 0x22, 0x08, 0x74, 0xe7, 0x95, 0x63 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.QueryMode – QueryMode() with Invalid Mode Fields returns EFI\_UNSUPPORTED | 1. Call QueryMode() with Invalid Mode Fields beyond **MaxMode**. The return code should be EFI\_UNSUPPORTED |

### SetMode()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.2.5.1 | 0x3680c8c3, 0x8fc6, 0x4fe2, 0xa2, 0xdb, 0x4f, 0xcb, 0xe1, 0x0a, 0x14, 0x87 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.SetMode – SetMode() with supported ModeNumber value sets mode and cursor postion | 1. Call SetMode() with supported ModeNumber value. Cursor position should be set to (0,0). Current mode should be the appointed mode. Other attributes should remain unchanged. |
| 5.6.2.5.2 | 0xcb1c6bc5, 0x6c12, 0x4d3a, 0x91, 0xc4, 0x2e, 0xdb, 0x09, 0xa3, 0x5d, 0x5f | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.SetMode – SetMode() with supported ModeNumber value returns EFI\_SUCCESS | 1. Call SetMode() with supported ModeNumber value. The return code should be EFI\_SUCCESS |
| 5.6.2.5.3 | 0xab044f50, 0xd0d3, 0x44f5, 0x92, 0x34, 0xe0, 0x52, 0xcc, 0x26, 0x47, 0x89 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.SetMode – SetMode() with Invalid Mode Fields, EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.Mode does not change before and after | 1. Call SetMode() with Invalid Mode Fields beyond **MaxMode**. EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.Mode should not change before and after |
| 5.6.2.5.4 | 0x6ce26a46, 0xab4a, 0x44df, 0x86, 0xc0, 0x3a, 0x97, 0xc3, 0xa3, 0x93, 0x0f | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.SetMode – SetMode() with Invalid Mode Fields returns EFI\_UNSUPPORTED | 1. SetMode() with Invalid Mode Fields beyond **MaxMode**. The return code should be EFI\_UNSUPPORTED. |

### SetAttribute()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.2.6.1 | 0xb401e101, 0x5386, 0x49fc, 0x89, 0x64, 0x54, 0x3b, 0xad, 0x90, 0x7b, 0x58 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.SetAttribute – SetAttribute() with supported attributes returns EFI\_SUCCESS | 1. Call SetAttribute() with supported attributes. The return code should be EFI\_SUCCESS. |
| 5.6.2.6.2 | 0x49b1f9ea, 0x085c, 0x4b2b, 0xa8, 0x98, 0x75, 0x6a, 0xa7, 0x61, 0x2f, 0x4a | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.SetAttribute – SetAttribute()should return EFI\_SUCCESS with valid attributes | 1. Check return status of SetAttribute()with valid attribute to set foreground color |
| 5.6.2.6.3 | 0xefa8f25f, 0x60fe, 0x4707, 0x9f,0x2b, 0x66, 0x12, 0xf6, 0x4d, 0x3f, 0x6e | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.SetAttribute – SetAttribute() changes output color and remains other mode fields unchanged | 1. Check all the fields of output mode. The background color and foreground color should be set as appointed value, and other fields should not be changed. |
| 5.6.2.6.4 | 0x3af1e31e, 0x1523, 0x4ad3, 0xa0, 0x77, 0x51, 0xd2, 0x32, 0x8e, 0xdf, 0x80 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.SetAttribute – SetAttribute() with supported attributes returns EFI\_SUCCESS and output color is set as expected | 1. After the multiple calls of SetAttribute(), check all the return codes and changes in output mode fields. |
| 5.6.2.6.5 | 0x42c6876b, 0x46e7, 0x47a5, 0xb4, 0x27, 0x25, 0x06, 0x1e, 0x25, 0xe8, 0xbf | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.SetAttribute – SetAttribute() with Invalid Attribute values, does not change EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.Mode. | 1. Call SetAttribute() with Invalid Attribute values. EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.Mode should not be changed |
| 5.6.2.6.6 | 0x300a1814, 0xd2c8, 0x4a51, 0xa9, 0x37, 0x0b, 0x8c, 0xe9, 0x3f, 0xb4, 0x45 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.SetAttribute – SetAttribute() with Invalid Attribute values returns EFI\_UNSUPPORTED | 1. Call SetAttribute() with Invalid Attribute values. The return code should be EFI\_UNSUPPORTED. |

### ClearScreen()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.2.7.1 | 0xa92ce5f8, 0x89a8, 0x4695, 0xbc, 0xb1, 0x59, 0x3e, 0x0e, 0x88, 0xe2, 0x41 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.ClearScreen – ClearScreen() sets cursor position to (0,0) and remain other attributes unchanged. | 1. Call ClearScreen() in all supported modes. The cursor position should be set to (0,0), and other attributes of output should not be changed. |
| 5.6.2.7.2 | 0xb3a0092f, 0xe768, 0x4359, 0xa9, 0xeb, 0x3d, 0x85, 0x27, 0x78, 0xc4, 0xcb | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.ClearScreen – ClearScreen() returns EFI\_SUCCESS. | 1. Call ClearScreen() in all supported modes. The return code should be EFI\_SUCCESS. |

### SetCursorPosition()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.2.8.1 | 0xe4f9fd56, 0x1e72, 0x44ee, 0xb0, 0x31, 0xae, 0xc6, 0xb4, 0xda, 0xb2, 0x0d | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.SetCursorPosition – SetCursorPosition() moves cursor to appointed position and remain other attributes unchanged. | 1. Call SetCursorPosition() in all supported modes to move cursor to every valid position within dimension boundary. The cursor position should be set to appointed value, and other attributes of output should not be changed. |
| 5.6.2.8.2 | 0xbe56dc0d, 0x8779, 0x4700, 0xb4, 0x4c, 0x6d, 0xf4, 0x39, 0xfb, 0xf6, 0xaa | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.SetCursorPosition – SetCursorPosition() returns EFI\_SUCCESS. | 1. Call SetCursorPosition() in all supported modes to move cursor to every valid position within dimension boundary. The return code should be EFI\_SUCCESS. |
| 5.6.2.8.3 | 0xa125b94f, 0xcbc6, 0x4e25, 0x80, 0x33, 0xfb, 0xf0, 0xde, 0x73, 0x14, 0x65 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.SetCursorPosition – SetCursorPosition() returns EFI\_SUCCESS and moves cursor to appointed position. | 1. Call SetCursorPosition() in all supported modes to move cursor to every valid position within dimension boundary.  2. Check return code and behavior of each call. |
| 5.6.2.8.4 | 0x85e9aabd, 0x1376, 0x4e67, 0xb6, 0x14, 0xce, 0xcf, 0x63, 0x36, 0x9b, 0x31 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.SetCursorPosition – SetCursorPosition() with Invalid Row/Column Numbers does not change cursor position. | 1. Call SetCursorPosition() with Invalid Row/Column Numbers. Mode->**CursorColumn** / **CursorRow** should remain unchanged. |
| 5.6.2.8.5 | 0xbeff2f08, 0xbc3e, 0x4e4f, 0xb8, 0x6f, 0x05, 0xb0, 0xe9, 0xd1, 0x0b, 0xa3 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.SetCursorPosition – SetCursorPosition() with Invalid Row/Column Numbers returns EFI\_UNSUPPORTED. | 1. Call SetCursorPosition() with Invalid Row/Column Numbers. The return code should be EFI\_UNSUPPORTED. |

### EnableCursor()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.2.9.1 | 0xdf85a087, 0xd1c9, 0x4739, 0x97, 0x2c, 0x4e, 0xd8, 0x61, 0x5f, 0x56, 0xd4 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.EnableCursor – EnableCursor() with TRUE returns EFI\_SUCCESS or EFI\_UNSUPPORTED | 1. Call EnableCursor() with TRUE. If EnableCursor() is unsupported, the return code should be EFI\_UNSUPPORTED |
| 5.6.2.9.2 | 0x318fe413, 0xd07d, 0x4aad, 0x9c, 0x62, 0xf8, 0xfe, 0x7f, 0x77, 0xbe, 0xb2 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.EnableCursor – EnableCursor() with TRUE changes cursor status to visible. | 1. Call EnableCursor() with TRUE.  2. If EnableCursor() success, **CursorVisible** should be TRUE |
| 5.6.2.9.3 | 0x07394e57, 0xf2f5, 0x4045, 0x8b, 0x2c, 0x91, 0xbb, 0x2b, 0xe4, 0x3c, 0x4e | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.EnableCursor – EnableCursor() with TRUE returns EFI\_SUCCESS or EFI\_UNSUPPORTED | 1. Call EnableCursor() with TRUE. If EnableCursor() is supported. The return code should be EFI\_SUCCESS |
| 5.6.2.9.4 | 0xb3121d1b, 0xbd25, 0x477d, 0xad, 0xc3, 0x5d, 0xe3, 0x1b, 0x19, 0x43, 0x25 | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.EnableCursor – EnableCursor() with FALSE returns EFI\_SUCCESS or EFI\_UNSUPPORTED | 1. Call EnableCursor() with FALSE. If EnableCursor() is unsupported, the return code should be EFI\_UNSUPPORTED |
| 5.6.2.9.5 | 0xcfd7fe8d, 0x1674, 0x4205, 0xb6, 0x3a, 0xe6, 0x4e, 0x86, 0x15, 0x66, 0x0c | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.EnableCursor – EnableCursor() with FALSE changes cursor status to invisible. | 1. Call EnableCursor() with FALSE.  2. If EnableCursor() success, CursorVisible should be FALSE |
| 5.6.2.9.6 | 0x3f2b2512, 0x91cf, 0x44d9, 0xae, 0xbd, 0x89, 0x76, 0x40, 0xf1, 0xb4, 0x1f | EFI\_SIMPLE\_TEXT\_OUTPUT\_PROTOCOL.EnableCursor – EnableCursor() with FALSE returns EFI\_SUCCESS or EFI\_UNSUPPORTED | 1. Call EnableCursor() with FALSE. If EnableCursor() is supported. The return code should be EFI\_SUCCESS |

## EFI\_SIMPLE\_POINTER\_PROTOCOL\_PROTOCOL Test

Reference Document:

*UEFI Specification*, EFI\_SIMPLE\_POINTER\_PROTOCOL Section.

### Reset()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.4.1.1 | 0x3fcb89c6, 0xe504, 0x4669, 0xbf, 0x31, 0xba, 0x03, 0xb7, 0x66, 0xc8, 0xc2 | EFI\_SIMPLE\_POINTER\_PROTOCOL.Reset - Reset() with an ExtendedVerification value of FALSE returns EFI\_SUCCESS | 1. Call Reset() with an ExtendedVerification value of FALSE. The return code should be EFI\_SUCCESS. |
| 5.6.4.1.2 | 0xd752813f, 0x32dc, 0x4820, 0xb7, 0x59, 0xe8, 0x97, 0x0c, 0xf3, 0x33, 0x89 | EFI\_SIMPLE\_POINTER\_PROTOCOL.Reset - GetState() after Reset() returns 0 for all related movement. | 1. Call Reset() with an ExtendedVerification value of FALSE.  2. Call GetState(). If success, RelativeMovementX, RelativeMovementY and RelativeMovementZ should be 0. |
| 5.6.4.1.3 | 0x716eefc7, 0x8c0a, 0x4636, 0xa0, 0xdb, 0x7e, 0x70, 0x20, 0xce, 0xe8, 0x5d | EFI\_SIMPLE\_POINTER\_PROTOCOL.Reset - GetState() after Reset() returns EFI\_UNSUPPORTED. | 1. Call Reset() with an ExtendedVerification value of FALSE.  2. Call GetState(). The return code maybe EFI\_UNSUPPORTED. |
| 5.6.4.1.4 | 0xce6806f5, 0xe186, 0x4c24, 0x83, 0xaa, 0x00, 0x4f, 0xac, 0xf0, 0x28, 0x65 | EFI\_SIMPLE\_POINTER\_PROTOCOL.Reset - Reset() with an ExtendedVerification value of TRUE returns EFI\_SUCCESS | 1. Call Reset() with an ExtendedVerification value of TRUE. The return code should be EFI\_SUCCESS |
| 5.6.4.1.5 | 0xd3e54374, 0x17b6, 0x417b, 0xae, 0xc7, 0xcc, 0x55, 0xcc, 0x42, 0x35, 0xa2 | EFI\_SIMPLE\_POINTER\_PROTOCOL.Reset - GetState() after Reset() returns 0 for all related movement. | 1. Call Reset() with an ExtendedVerification value of TRUE.  2. Call GetState(). If success, RelativeMovementX, RelativeMovementY and RelativeMovementZ should be 0. |
| 5.6.4.1.6 | 0xd8a03978, 0x7023, 0x4d61, 0x92, 0xbd, 0x15, 0xd3, 0x9b, 0x3f, 0x5d, 0x11 | EFI\_SIMPLE\_POINTER\_PROTOCOL.Reset - GetState() after Reset() returns EFI\_UNSUPPORTED. | 1. Call Reset() with an ExtendedVerification value of TRUE.  2. Call GetState(). The return code maybe EFI\_UNSUPPORTED. |

### GetState()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.4.2.1 | 0x5271062e, 0xdef9, 0x4d30, 0x84, 0x3b, 0x8d, 0x6e, 0x41, 0x33, 0x13, 0xf3 | EFI\_SIMPLE\_POINTER\_PROTOCOL.GetState - GetState() after Reset() returns 0 for all related movement. | 1. Call Reset() with an ExtendedVerification value of FALSE.  2. Call GetState(). If success, RelativeMovementX, RelativeMovementY and RelativeMovementZ should be 0. |
| 5.6.4.2.2 | 0x7614c447, 0x12a0, 0x403d, 0x8a, 0xde, 0x98, 0x97, 0x51, 0x7d, 0xd8, 0x49 | EFI\_SIMPLE\_POINTER\_PROTOCOL.GetState - GetState() returns EFI\_NOT\_READY when there is no move since last call of GetState(). | 1. Call Reset() with an ExtendedVerification value of FALSE.  2. Call GetState().  3. Call GetState() again, the return code should be EFI\_NOT\_READY. |
| 5.6.4.2.3 | 0x2f8f8710, 0x02dd, 0x411f, 0xaa, 0xb5, 0x27, 0xe1, 0x3a, 0x6a, 0xb2, 0x79 | EFI\_SIMPLE\_POINTER\_PROTOCOL.GetState - GetState() after Reset() returns 0 for all related movement. | 1. Call Reset() with an ExtendedVerification value of TRUE.  2. Call GetState(). If success, RelativeMovementX, RelativeMovementY and RelativeMovementZ should be 0. |
| 5.6.4.2.4 | 0x3db7ea18,0xda9d,  0x4760,  0xa7,0x43,  0x04,0xb4,  0x8d,0x14,  0x4e,0x90 | EFI\_SIMPLE\_POINTER\_PROTOCOL.GetState - GetState() returns EFI\_NOT\_READY when there is no move since last call of GetState(). | 1. Call Reset() with an ExtendedVerification value of TRUE.  2. Call GetState().  3. Call GetState() again, the return code should be EFI\_NOT\_READY. |

## EFI\_SERIAL\_IO\_PROTOCOL Test

Reference Document:

*UEFI Specification*, EFI\_SERIAL\_IO\_PROTOCOL Section.

### Reset()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.5.1.1 | 0x2e369256, 0x6c78, 0x49e9, 0x9e, 0xd5, 0xe3, 0xd2, 0x88, 0x34, 0x33, 0xa0 | EFI\_SERIAL\_IO\_PROTOCOL .Reset - Reset() returns EFI\_SUCCESS. | 1. Call Reset(). The return code should be EFI\_SUCCESS. |

### SetAttributes()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.5.2.1 | 0x34260cb2, 0x43ae, 0x4853, 0x87, 0x4b, 0x47, 0x7c, 0xeb, 0x14, 0x42, 0x02 | EFI\_SERIAL\_IO\_PROTOCOL .SetAttributes - SetAttributes() with valid BaudRate returns EFI\_SUCCESS. | 1. Call SetAttributes() with various valid BaudRate. The return code should be EFI\_SUCCESS and the BaudRate field of **Mode** should be equal to the set value. |
| 5.6.5.2.2 | 0x3fd35bee, 0x5013, 0x472f, 0xa0, 0x08, 0xbd, 0xdf, 0x31, 0x9c, 0xe6, 0x6b | EFI\_SERIAL\_IO\_PROTOCOL .SetAttributes - SetAttributes() with valid ReceiveFifoDepth returns EFI\_SUCCESS. | 1. Call SetAttributes() with various valid ReceiveFifoDepth. The return code should be EFI\_SUCCESS and the ReceiveFifoDepth field of Mode should be equal to the set value. |
| 5.6.5.2.3 | 0x8cf74222, 0x7134, 0x47b6, 0xa5, 0x82, 0xf4, 0xd9, 0xad, 0xa7, 0xa3, 0xf4 | EFI\_SERIAL\_IO\_PROTOCOL .SetAttributes - SetAttributes() with valid Timeout returns EFI\_SUCCESS. | 1. Call SetAttributes() with various valid Timeout. The return code should be EFI\_SUCCESS and the Timeout field of Mode should be equal to the set value. |
| 5.6.5.2.4 | 0x68f91273, 0x0078, 0x4e6c, 0xb9, 0xdb, 0x62, 0x59, 0xb5, 0x39, 0xf7, 0x4a | EFI\_SERIAL\_IO\_PROTOCOL .SetAttributes - SetAttributes() with valid Parity returns EFI\_SUCCESS. | 1. Call SetAttributes() with various valid Parity. The return code should be EFI\_SUCCESS and the Parity field of Mode should be equal to the set value. |
| 5.6.5.2.5 | 0xdf6038c2, 0x3752, 0x4e22, 0xab, 0x4c, 0xfe, 0x66, 0x67, 0x0c, 0xa3, 0xdf | EFI\_SERIAL\_IO\_PROTOCOL .SetAttributes - SetAttributes() with valid DataBits returns EFI\_SUCCESS. | 1. Call SetAttributes() with various valid DataBits. The return code should be EFI\_SUCCESS and the DataBits field of Mode should be equal to the set value. |
| 5.6.5.2.6 | 0xdf6f2692, 0x9a0d, 0x4b0f, 0xbc, 0x8e, 0x36, 0x8b, 0x6a, 0x03, 0xe0, 0xb1 | EFI\_SERIAL\_IO\_PROTOCOL .SetAttributes - SetAttributes() with valid StopBits returns EFI\_SUCCESS. | 1. Call SetAttributes() with various valid StopBits. The return code should be EFI\_SUCCESS and the StopBits field of Mode should be equal to the set value. |
| 5.6.5.2.7 | 0xb199d5d2, 0x1143, 0x499e, 0xa5, 0xf8, 0xf0, 0xa7, 0x6f, 0x79, 0xfe, 0xe5 | EFI\_SERIAL\_IO\_PROTOCOL .SetAttributes - SetAttributes() with default attributes returns EFI\_SUCCESS. | 1. Call SetAttributes() with default attributes: BaudRate=115200; FifoDepth=1; Timeout=1000000; Parity=**NoParity**; DataBits=8; StopBits=**OneStopBit**; The return code should be EFI\_SUCCESS |
| 5.6.5.2.8 | 0x3041ec45, 0x00af, 0x4787, 0xb1, 0xe9, 0x15, 0xb8, 0x7a, 0xc5, 0xdd, 0xc8 | EFI\_SERIAL\_IO\_PROTOCOL .SetAttributes - SetAttributes() with nonstandard BaudRate values returns EFI\_SUCCESS and set BaudRate as the nearest standard baud rate value. | 1. Call SetAttributes() with nonstandard BaudRate values. The return code should be EFI\_SUCCESS and the BaudRate field of Mode should be equal to the nearest standard baud rate value. |
| 5.6.5.2.9 | 0x7a5cca70, 0x46c7, 0x4488, 0x87, 0x65, 0x84, 0x33, 0x66, 0x78, 0xa5, 0x01 | EFI\_SERIAL\_IO\_PROTOCOL .SetAttributes - SetAttributes() with unsupported BaudRate returns EFI\_INVALID\_PARAMETER. | 1. Call SetAttributes() with unsupported BaudRate. The return code should be EFI\_INVALID\_PARAMETER. |
| 5.6.5.2.10 | 0x190ca14d, 0xa6c2, 0x4a42, 0x86, 0x29, 0xa5, 0x14, 0x96, 0xc8, 0xe0, 0x52 | EFI\_SERIAL\_IO\_PROTOCOL .SetAttributes - SetAttributes() with unsupported ReceiveFifoDepth returns EFI\_INVALID\_PARAMETER. | 1. Call SetAttributes() with unsupported ReceiveFifoDepth. The return code should be EFI\_INVALID\_PARAMETER. |
| 5.6.5.2.11 | 0xd40c796b, 0xb654, 0x4fb5, 0x88, 0xb0, 0x1e, 0xc8, 0x2a, 0x27, 0x13, 0x50 | EFI\_SERIAL\_IO\_PROTOCOL .SetAttributes - SetAttributes() with unsupported Timeout returns EFI\_INVALID\_PARAMETER. | 1. Call SetAttributes() with unsupported Timeout. The return code should be EFI\_INVALID\_PARAMETER. |
| 5.6.5.2.12 | 0x15dc5ee1, 0x9871, 0x4e25, 0xb2, 0x22, 0xc5, 0x38, 0x5c, 0x9b, 0xf3, 0x6b | EFI\_SERIAL\_IO\_PROTOCOL .SetAttributes - SetAttributes() with unsupported Parity returns EFI\_INVALID\_PARAMETER. | 1. Call SetAttributes() with unsupported Parity. The return code should be EFI\_INVALID\_PARAMETER. |
| 5.6.5.2.13 | 0x0aa15e38, 0xb05c, 0x46cf, 0xb1, 0xf3, 0x1e, 0xb7, 0x41, 0x37, 0xb8, 0xbf | EFI\_SERIAL\_IO\_PROTOCOL .SetAttributes - SetAttributes() with unsupported DataBits returns EFI\_INVALID\_PARAMETER. | 1. Call SetAttributes() with unsupported DataBits. The return code should be EFI\_INVALID\_PARAMETER. |
| 5.6.5.2.14 | 0x174a5c87, 0x74cf, 0x4e88, 0x84, 0x04, 0x68, 0x3e, 0xcb, 0x40, 0xf3, 0x2f | EFI\_SERIAL\_IO\_PROTOCOL .SetAttributes - SetAttributes() with unsupported StopBits returns EFI\_INVALID\_PARAMETER. | 1. Call SetAttributes() with unsupported StopBits. The return code should be EFI\_INVALID\_PARAMETER. |

### SetControl()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.5.3.1 | 0xac56dfb5, 0xce1c, 0x42a6, 0x98, 0xc9, 0xc6, 0xf5, 0xc8, 0xad, 0x83, 0xda | EFI\_SERIAL\_IO\_PROTOCOL .SetControl - SetControl() with valid bits returns EFI\_SUCCESS and GetControl() returns the set bits. | 1. Call SetControl() with valid control bits. The return code should be EFI\_SUCCESS.  2. Call GetControl(). The valid control bits should be set. |
| 5.6.5.3.2 | 0x00605cbc, 0x3965, 0x4b61, 0xa2, 0x54, 0x2b, 0x2b, 0x72, 0x31, 0x72, 0xea | EFI\_SERIAL\_IO\_PROTOCOL .SetControl - SetControl() with unsupported control bits returns EFI\_UNSUPPORTED. | 1. Call SetControl() with unsupported control bits. The return code should be EFI\_UNSUPPORTED. |

### GetControl()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.5.4.1 | 0x131f5894, 0x1613, 0x4f3e, 0xbd, 0x45, 0x2b, 0xdd, 0xb7, 0xed, 0x22, 0xb0 | EFI\_SERIAL\_IO\_PROTOCOL .GetControl - GetControl() returns EFI\_SUCCESS and gets the bits set by SetControl(). | 1. Call SetControl() with valid control bits.  2. Call GetControl().The return code should be EFI\_SUCCESS and the valid control bits should be returned. |
| 5.6.5.4.2 | 0xdd059dc5, 0x6558, 0x4d43, 0xac, 0x65, 0x58, 0xa6, 0x1d, 0x64, 0x8d, 0xb0 | EFI\_SERIAL\_IO\_PROTOCOL .GetControl - GetControl() returns EFI\_SUCCESS and gets the bit of EFI\_SERIAL\_INPUT\_BUFFER\_EMPTY after buffer contents are read out. | 1. Call Read() to read out buffer contents.  2. Call GetControl().The return code should be EFI\_SUCCESS and EFI\_SERIAL\_INPUT\_BUFFER\_EMPTY is set. |

### Write()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.5.5.1 | 0x72c50358, 0xc760, 0x4200, 0x8d, 0xb2, 0x09, 0x4d, 0x96, 0x84, 0x6f, 0x1a | EFI\_SERIAL\_IO\_PROTOCOL .Write - Write() in software-loopback mode returns EFI\_SUCCESS and Read() gets the same contents. | 1. Call Write() in software-loopback mode. The return code should be EFI\_SUCCESS.  2. Call Read() to get buffer. It should return the written contents. |
| 5.6.5.5.2 | 0x688bf990, 0xfd8f, 0x430e, 0x8e, 0x1c, 0x78, 0x07, 0x2d, 0x74, 0xbd, 0x08 | EFI\_SERIAL\_IO\_PROTOCOL .Write - Write() in hardware-loopback mode returns EFI\_SUCCESS and Read() gets the same contents. | 1. Call Write() in hardware-loopback mode. The return code should be EFI\_SUCCESS.  2. Call Read() to get buffer. It should return the written contents. |
| 5.6.5.5.3 | 0x198873b8, 0xe8f2, 0x4bfd, 0xa0, 0x20, 0x36, 0xff, 0xb4, 0x93, 0x72, 0x02 | EFI\_SERIAL\_IO\_PROTOCOL .Write - Write() in non-loopback mode returns EFI\_SUCCESS. | 1. Call Write() in non-loopback mode. The return code should be EFI\_SUCCESS. |

### Read()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.5.6.1 | 0x8ad0312f, 0x4cfc, 0x4611, 0xb7, 0x62, 0x85, 0x3a, 0xa3, 0x9d, 0x2f, 0xd9 | EFI\_SERIAL\_IO\_PROTOCOL .Read - Read() in software-loopback mode returns EFI\_SUCCESS and gets the same contents written. | 1. Call Write() in software-loopback mode.  2. Call Read() in software-loopback to get buffer. It should return the written contents. The return code should be EFI\_SUCCESS. |
| 5.6.5.6.2 | 0x76cb227f, 0x312d, 0x4476, 0x8c, 0x59, 0x6a, 0x98, 0x27, 0x5b, 0x62, 0x3d | EFI\_SERIAL\_IO\_PROTOCOL .Read - Read() in hardware-loopback mode returns EFI\_SUCCESS and gets the same contents written. | 1. Call Write() in hardware-loopback mode.  2. Call Read() in hardware-loopback to get buffer. It should return the written contents. The return code should be EFI\_SUCCESS. |
| 5.6.5.6.3 | 0x3faefba1, 0x4049, 0x4868, 0x8f, 0x34, 0x59, 0xaf, 0x3e, 0x62, 0xdf, 0xb0 | EFI\_SERIAL\_IO\_PROTOCOL .Read - Read() in hardware-loopback mode without any characters in buffer returns EFI\_TIME\_OUT and set buffer size to 0. | 1. Call Read() to read out all contents in buffer.  2. Call Read() again, the return code should be EFI\_TIME\_OUT and BufferSize should be 0. |
| 5.6.5.6.4 | 0xc96db50e, 0xd269, 0x4fb0, 0x88, 0xbd, 0x6a, 0x02, 0x06, 0x66, 0x53, 0xa7 | EFI\_SERIAL\_IO\_PROTOCOL .Read - Read() in hardware-loopback mode with BufferSize=2 returns EFI\_TIME\_OUT when there is only 1 byte contents in serial buffer. | 1. Call Read() to read out all contents in buffer.  2. Call Write() to write 1 byte into serial buffer.  3. Call Read() again with BufferSize = 2. The return code should be EFI\_TIME\_OUT, BufferSize should be 1 and 1 byte should be read. |
| 5.6.5.6.5 | 0xb636572b, 0x7aaa, 0x4146, 0x8d, 0xd4, 0x18, 0xef, 0xac, 0xb4, 0x8a, 0x1a | EFI\_SERIAL\_IO\_PROTOCOL .Read - Read() in software-loopback mode without any characters in buffer returns EFI\_TIME\_OUT and set buffer size to 0. | 1. Call Read() to read out all contents in buffer.  2. Call Read() again, the return code should be EFI\_TIME\_OUT and BufferSize should be 0. |
| 5.6.5.6.6 | 0x48050436, 0xc835, 0x4a24, 0x87, 0x75, 0x4d, 0x2e, 0x47, 0x88, 0xb5, 0x97 | EFI\_SERIAL\_IO\_PROTOCOL .Read - Read() in software-loopback mode with BufferSize=2 returns EFI\_TIME\_OUT when there is only 1 byte contents in serial buffer. | 1. Call Read() to read out all contents in buffer.  2. Call Write() to write 1 byte into serial buffer.  3. Call Read() again with BufferSize = 2. The return code should be EFI\_TIME\_OUT, BufferSize should be 1 and 1 byte should be read. |

## EFI\_GRAPHICS\_OUTPUT\_PROTOCOL Test

Reference Document:

*UEFI Specification*, EFI\_GRAPHICS\_OUTPUT\_PROTOCOL Section.

### QueryMode()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.6.1.1 | 0xd1824539, 0x92cd, 0x434c, 0x81, 0x65, 0x87, 0x2c, 0xc2, 0x1a, 0x5f, 0x9e | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL. QueryMode – returns EFI\_SUCCESS with valid parameter. | For valid graphics mode number from 0 to MaxMode-1:  •Call SetMode() to switch the video device to the specified mode.  •Call QueryMode() with the current mode to get the current info structure and check the content of Info.  The returned status should be EFI\_SUCCESS.  For PixelBlueGreenRedReserved8BitPerColor or PixelRedGreenBlueReserved8BitPerColor, the FrameBufferSize should be PixelsPerScanLine \* VerticalResolution \* PixelElementSize. |
| 5.6.6.1.2 | 0x82dfd41e, 0x49db, 0x4c86, 0x99, 0xbb, 0xc5, 0x74, 0x33, 0x4b, 0xa0, 0xc3 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL. QueryMode – Call QueryMode() with *MaxMode*. | 1. Call QueryMode() with *MaxMode*. The returned status must be EFI\_INVALID\_PARAMETER.  2. Call QueryMode() with a *SizeOfInfo* value of NULL. The returned status must be EFI\_INVALID\_PARAMETER.  3. Call QueryMode() with an Info value of NULL. The returned status must be EFI\_INVALID\_PARAMETER.  4. For valid graphics mode number from 0 to *MaxMode*-1:  •Call QueryMode() with the specified mode number. The returned status should be any value except EFI\_INVALID\_PARAMETER.  •The called allocated buffer that the Info points to should not be NULL. |
| 5.6.6.1.3 | 0x8ebcd9ab, 0x69a9, 0x48a2, 0x9b, 0xbc, 0x8c, 0x47, 0x9e, 0x68, 0x91, 0x56 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL. QueryMode – Call QueryMode() with a *SizeOfInfo* value of NULL. | 1. Call QueryMode() with *MaxMode*. The returned status must be EFI\_INVALID\_PARAMETER.  2. Call QueryMode() with a *SizeOfInfo* value of NULL. The returned status must be EFI\_INVALID\_PARAMETER.  3. Call QueryMode() with an Info *value* of NULL. The returned status must be EFI\_INVALID\_PARAMETER.  4. For valid graphics mode number from 0 to *MaxMode*-1:  •Call QueryMode() with the specified mode number. The returned status should be any value except EFI\_INVALID\_PARAMETER.  •The called allocated buffer that the Info points to should not be NULL |
| 5.6.6.1.4 | 0x394e306b, 0x652a, 0x403a, 0xbd, 0x15, 0xdb, 0x9b, 0x46, 0xc3, 0x44, 0x3b | ConsoleContro. QueryMode – Call QueryMode() with an *Info* value of NULL. | 1. Call QueryMode() with *MaxMode*. The returned status must be EFI\_INVALID\_PARAMETER.  2. Call QueryMode() with a SizeOfInfo value of NULL. The returned status must be EFI\_INVALID\_PARAMETER.  3. Call QueryMode() with an Info value of NULL. The returned status must be EFI\_INVALID\_PARAMETER.  4. For valid graphics mode number from 0 to *MaxMode*-1:  •Call QueryMode() with the specified mode number. The returned status should be any value but EFI\_INVALID\_PARAMETER.  •The called allocated buffer that the Info points to should not be NULL |
| 5.6.6.1.5 | 0xe7782dc5, 0x2b78, 0x460f, 0xb1, 0x02, 0x88, 0xd5, 0x12, 0x06, 0x45, 0x1f | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL. QueryMode – Call QueryMode() with a valid *ModeNumber*. The returned status should be EFI\_SUCCESS. | 1. Call QueryMode() with *MaxMode*. The returned status must be EFI\_INVALID\_PARAMETER.  2. Call QueryMode() with a *SizeOfInfo* value of NULL. The returned status must be EFI\_INVALID\_PARAMETER.  3. Call QueryMode() with an *Info* value of NULL. The returned status must be EFI\_INVALID\_PARAMETER.  4. For valid graphics mode number from 0 to *MaxMode*-1:  •Call QueryMode() with the specified mode number. The returned status should be any value except EFI\_INVALID\_PARAMETER.  •The called allocated buffer that the Info points to should not be NULL |
| 5.6.6.1.6 | 0x486360f1, 0x6b8e, 0x48b5, 0x8b, 0xa8, 0xae, 0x40, 0xeb, 0x3b, 0x07, 0xa2 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL. QueryMode – Call QueryMode() with with valid parameters. | 1. Call QueryMode() with MaxMode. The returned status must be EFI\_INVALID\_PARAMETER.  2. Call QueryMode() with a *SizeOfInfo* value of NULL. The returned status must be EFI\_INVALID\_PARAMETER.  3. Call QueryMode() with an *Info* value of NULL. The returned status must be EFI\_INVALID\_PARAMETER.  4. For valid graphics mode number from 0 to *MaxMode*-1:  •Call QueryMode() with the specified mode number. The returned status should be any value except EFI\_INVALID\_PARAMETER.  •The called allocated buffer that the Info points to should not be NULL |
| 5.6.6.1.7 | 0xdc19ab69, 0x764e, 0x429b, 0xa5, 0x3f, 0xb8, 0x1e, 0xd6, 0x3c, 0xd6, 0xc0 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL. QueryMode – Call QueryMode()to Check the mode structure and dump it. | 1. Call QueryMode() with MaxMode. The returned status must be EFI\_INVALID\_PARAMETER.  2. Call QueryMode() with a SizeOfInfo value of NULL. The returned status must be EFI\_INVALID\_PARAMETER.  3. Call QueryMode() with an *Info* value of NULL. The returned status must be EFI\_INVALID\_PARAMETER.  4. For valid graphics mode number from 0 to MaxMode-1:  •Call QueryMode() with the specified mode number. The returned status should be any value except EFI\_INVALID\_PARAMETER.  •The called allocated buffer that the Info points to should not be NULL. |

### SetMode()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.6.2.1 | 0xb3a4939b, 0xd00a, 0x4da7, 0xaf, 0x6d, 0xf3, 0xee, 0xcb, 0xf9, 0x99, 0x0c | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL. SetMode –SetMode() returns EFI\_SUCCESS when setting the graphics device and the set of active video output devices to the video mode specified by ModeNumber. | Call SetMode() with valid mode numbers from 0 to *MaxMode*-1.The returned status should be EFI\_SUCCESS. |
| 5.6.6.2.2 | 0x128e953b, 0xe6ec, 0x4f93, 0xa8, 0xec, 0x72, 0xc5, 0x9b, 0x8a, 0x40, 0x43 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL. SetMode – Call SetMode()with valid *ModeNumber*. | 1. For valid graphics mode number from 0 to *MaxMode*-1:  •Call **SetMode()** with valid mode, the returned status should not be **EFI\_UNSUPPORTED** and should be EFI\_SUCCESS.  •Call **QueryMode()** with the same mode number as **SetMode()**.  •The Info structure the **QueryMode()** returns and current mode of graphic device should be the same.  4. Call **SetMode()** with *MaxMode*. The returned status should be **EFI\_UNSUPPORTED**. |
| 5.6.6.2.3 | 0x4f13e7ba, 0xb35a, 0x4bf7, 0xb1, 0xc0, 0xfe, 0x39, 0x9c, 0x49, 0x97, 0xfe | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL. SetMode –Call QueryMode() with the *ModeNumber* the SetMode() set , then  compare the Info structure QueryMode() returns with current mode of graphic device in order to verify whether they are same, and at last dump the Info structure. | 1. For valid graphics mode number from 0 to *MaxMode*-1:  •Call **SetMode()** with valid mode, the returned status should not be **EFI\_UNSUPPORTED** and should be EFI\_SUCCESS.  •Call **QueryMode()** with the same mode number as **SetMode()**.  •The Info structure the **QueryMode()** returns and current mode of graphic device should be the same.  2. Call **SetMode()** with MaxMode. The returned status should be **EFI\_UNSUPPORTED**. |
| 5.6.6.2.4 | 0x8776b9dc, 0x711e, 0x4e36, 0x99, 0x21, 0x7e, 0xa7, 0xc4, 0xc7, 0xee, 0x6d | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL. SetMode – Call SetMode()with valid *MaxMode*. | 1. For valid graphics mode number from 0 to *MaxMode*-1:  •Call **SetMode()** with valid mode, the returned status should not be **EFI\_UNSUPPORTED** and should be EFI\_SUCCESS.  •Call **QueryMode()** with the same mode number as **SetMode()**.  •The Info structure the **QueryMode()** returns and current mode of graphic device should be the same.  2. Call **SetMode()** with *MaxMode*. The returned status should be EFI\_UNSUPPORTED. |

### Blt()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.6.3.1 | 0x95a44702, 0xcea0, 0x480f, 0x9f, 0x84, 0xe2, 0x4c, 0x17, 0xbf, 0x47, 0x79 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt – EfiBltVideoFill operation should fill graphic screen with pixels. | For valid graphics mode number from 0 to *MaxMode*-1:  •Call Blt() with a BltOperation value of *EfiBltVideoFill* to write data from the *BltBuffer* pixel (0, 0) directly to every pixel of the video display rectangle (*DestinationX*, *DestionationY*) (*DestionationX*+*Width*, *DestionationY*+*Height*). The returned status should be EFI\_SUCCESS.  •Call Blt() with a *BltOperation* value of *EfiBltVideoToBltBuffer* to retrieve the rectangles drawn by the last *EfiBltVideoFill* operation. All the retrieved rectangles should be the same pixel used in the last *EfiBltVideoFill* operation. If pixels verification passes, the return status should be EFI\_SUCCESS. |
| 5.6.6.3.2 | 0x699c30b0, 0xab3f, 0x45d9, 0xbd, 0x69, 0x6b, 0x93, 0x96, 0xb7, 0x7e, 0x66 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt – EfiBltVideoFill operation should fill graphic screen with pixels. | For valid graphics mode number from 0 to MaxMode-1:  •Call Blt() with a *BltOperation* value of *EfiBltVideoFill* to write data from the BltBuffer pixel (0, 0) directly to every pixel of the video display rectangle (*DestinationX*, *DestionationY*) (*DestinationX* +*Width*, *DestionationY* +*Height*). The returned status should be EFI\_SUCCESS.  •Call Blt() with a *BltOperation* value of *EfiBltVideoToBltBuffer* to retrieve the rectangles drawn by the last *EfiBltVideoFill* operation. All the retrieved rectangles should be the same pixel used in the last *EfiBltVideoFill* operation. If pixels verification passes, the return status should be EFI\_SUCCESS. |
| 5.6.6.3.3 | 0xc34c3fa4, 0xa61e, 0x4598, 0x9f, 0x80, 0x2d, 0xee, 0x8e, 0x2c, 0x9b, 0x57 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt – EfiBltBufferToVideo  operation should write data to video screen and EfiBltVideoToBltBuffer operation should read data from video display rectangle. | For valid graphics mode number from 0 to *MaxMode*-1:  •Call Blt() with a *BltOperation* value of *EfiBltVideoToBltBuffer* to read data from the video display rectangle (*SourceX*, *SourceY*)  (SourceX+Width, SourceY+Height) and place it in the BltBuffer rectangle (DestinationX, DestionationY) (DestionationX+Width,  *DestionationY*+*Height*). The returned status should be EFI\_SUCCESS.  •Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to write data from the *BltBuffer* rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) directly to the video display rectangle (*DestinationX*, *DestionationY*) (*DestionationX*+*Width*,  *DestionationY*+*Height*). – Source and Destination should reverse with that of EfiBltVideoToBltBuffer operation. The returned status should be EFI\_SUCCESS.  •Call Blt() with a *BltOperation* value of **EfiBltVideoToBltBuffer** to read data from the video display rectangle (*SourceX*, *SourceY*)  (*SourceX*+*Width*, *SourceY*+*Height*) and place it in the BltBuffer2 rectangle (0, 0) (*Width*, *Height*) with a delta value of 0.The returned status should be EFI\_SUCCESS.  •The *BltBuffer* and BltBuffer2 should be the same. |
| 5.6.6.3.4 | 0x33a341ea, 0xc6a2, 0x4037, 0x8a, 0x2d, 0x19, 0xea, 0x1f, 0xe2, 0xf2, 0xa6 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt – EfiBltBufferToVide**o o**peration should write data to video screen and EfiBltVideoToBltBuffer operation should read data from video display rectangle. | For valid graphics mode number from 0 to *MaxMode*-1:  •Call Blt() with a *BltOperation* value of **EfiBltVideoToBltBuffer** to read data from the video display rectangle (*SourceX*, *SourceY*)  (SourceX+Width, SourceY+Height) and place it in the BltBuffer rectangle (DestinationX, DestionationY) (DestionationX+Width,  *DestionationY*+*Height*). The returned status should be EFI\_SUCCESS.  •Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to write data from the *BltBuffer* rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) directly to the video display rectangle (*DestinationX*, *DestionationY*) (*DestionationX*+*Width*,  *DestionationY*+*Height*). – Source and Destination should reverse with that of EfiBltVideoToBltBuffer operation. The returned status should be EFI\_SUCCESS.  •Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer to read data from the video display rectangle (*SourceX*, *SourceY*)  (*SourceX*+*Width*, *SourceY*+*Height*) and place it in the BltBuffer2 rectangle (0, 0) (*Width*, *Height*) with a delta value of 0.The returned status should be EFI\_SUCCESS.  •The *BltBuffer* and BltBuffer2 should be the same. |
| 5.6.6.3.5 | 0x13f113dc, 0xafd0, 0x4658, 0xb7, 0xfb, 0x83, 0xd5, 0xae, 0x6f, 0x10, 0x58 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt – EfiBltBufferToVide operation should write data to video screen and EfiBltVideoToBltBuffer operation should read data from video display rectangle. | For valid graphics mode number from 0 to *MaxMode*-1:  •Call Blt() with a *BltOperation* value of *EfiBltVideoToBltBuffer* to read data from the video display rectangle (*SourceX*, *SourceY*)  (*SourceX*+*Width*, *SourceY*+*Height*) and place it in the *BltBuffer* rectangle (*DestinationX*, *DestionationY*) (*DestionationX*+*Width*,  *DestionationY*+*Height*). The returned status should be EFI\_SUCCESS.  •Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to write data from the *BltBuffer* rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) directly to video display rectangle (*DestinationX*, *DestionationY*) (*DestionationX*+*Width*,  *DestionationY*+*Height*). – Source and Destination should reverse with that of EfiVideoToBltBuffer operation. The returned status should be EFI\_SUCCESS.  •Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer to read data from the video display rectangle (*SourceX*, *SourceY*)  (*SourceX*+*Width*, *SourceY*+*Height*) and place it in the BltBuffer2 rectangle (0, 0) (*Width*, *Height*) with a delta value of 0.The returned status should be EFI\_SUCCESS.  •The *BltBuffer* and BltBuffer2 should be the same. |
| 5.6.6.3.6 | 0x5ca291cc, 0x84a0, 0x489d, 0x9b, 0x2a, 0x0f, 0x2f, 0xcc, 0xc6, 0x0b, 0x29 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt – EfiBltBufferToVide operation should write data to video screen and EfiBltVideoToBltBuffer operation should read data from video display rectangle. | For valid graphics mode number from 0 to *MaxMode*-1:  •Call Blt() with a *BltOperation* value of *EfiBltVideoToBltBuffer* to read data from the video display rectangle (*SourceX*, *SourceY*)  (*SourceX*+*Width*, *SourceY*+*Height*) and place it in the BltBuffer rectangle (*DestinationX*, *DestionationY*) (*DestionationX*+*Width*,  *DestionationY*+*Height*). The returned status should be EFI\_SUCCESS.  •Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to write data from the *BltBuffer* rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) directly to video display rectangle (*DestinationX*, *DestionationY*) (*DestionationX*+*Width*,  *DestionationY*+*Height*). – Source and Destination should reverse with that of EfiBltVideoToBltBuffer operation. The returned status should be EFI\_SUCCESS.  •Call Blt() with a *BltOperation* value of *EfiBltVideoToBltBuffer* to read data from the video display rectangle (*SourceX*, *SourceY*)  (*SourceX*+*Width*, *SourceY*+*Height*) and place it in the BltBuffer2 rectangle (0, 0) (Width, Height) with a delta value of 0.The returned status should be EFI\_SUCCESS.  •The BltBuffer and BltBuffer2 should be the same. |
| 5.6.6.3.7 | 0x6c2632c0, 0xe3de, 0x4afc, 0xb3, 0xa1, 0xbe, 0x50, 0x75, 0xab, 0x2d, 0x7a | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltVideoToVideo operation should copy data from one video display rectangle to another. | For valid graphics mode number from 0 to *MaxMode*-1:  •Call Blt() with a *BltOperation* value of  *EfiBltVideoToBltBuffer* to save original screen and read data from the video display rectangle  (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) and place it in *BltBuffer* rectangle (0, 0) (*Width*, *Height*) with a delta value of 0.  •Call Blt() with a *BltOperation* value of EfiBltVideoToVideo to copy data from the video display rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) to another display rectangle (*DestinationX*, *DestinationY*) (*DestinationX*+*Width*, *DestionationY*+*Height*). The returned status should be EFI\_SUCCESS.  •Call Blt() with a *BltOperation* value of *EfiBltVideoToBltBuffer* again to retrieve the area from the video display and read data from the video display rectangle (*DestinationX*, *DestinationY*) (*DestinationX*+*Width*, *DestinationY*+*Height*) and place it in the BltBuffer2  rectangle (0, 0) (*Width*, *Height*) with a delta value of 0.  •The BltBuffer and BltBuffer2 should be the same. |
| 5.6.6.3.8 | 0x07d1d0c1, 0x3884, 0x4310, 0x97, 0xbc, 0x16, 0xd6, 0xaa, 0x1a, 0x21, 0x80 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltVideoToVideo operation should copy data from video display rectangle to another. | For valid graphics mode number from 0 to *MaxMode*-1:  •Call Blt() with a *BltOperation* value of  *EfiBltVideoToBltBuffer* to save original screen and read data from the video display rectangle  (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) and place it in *BltBuffer* rectangle (0, 0) (*Width*, *Height*) with a delta value of 0.  •Call Blt() with a *BltOperation* value of EfiBltVideoToVideo to copy data from the video display rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) to another display rectangle (*DestinationX*, *DestinationY*) (*DestinationX*+*Width*, *DestionationY*+*Height*). The returned status should be EFI\_SUCCESS.  •Call Blt() with a *BltOperation* value of *EfiBltVideoToBltBuffer* again to retrieve the area from the video display and read data from the video display rectangle (*DestinationX*, *DestinationY*) (*DestinationX*+*Width*, *DestinationY*+*Height*) and place it in the BltBuffer2  rectangle (0, 0) (*Width*, *Height*) with a delta value of 0.  •The BltBuffer and BltBuffer2 should be the same. |
| 5.6.6.3.9 | 0x11af616a, 0xbef5, 0x4590, 0xbe, 0x85, 0x19, 0x52, 0xa0, 0x0d, 0xe1, 0xaf | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –Call Blt()with invalid *BltOperation*. | Repeat the following step 6 times:  •Call Blt() with a *BltOperation* value other than  EfiBltVideoFill/EfiBltVideoToBltBuffer/EfiBltBufferToVideo/EfiBltVideoToVideo.The returned status should be EFI\_INVALID\_PARAMETER.  3. Restore the screen mode. |
|  |  |  |  |
| 5.6.6.3.11 | 0xe967bdc7, 0xa0ea, 0x4fd7, 0xab, 0xba, 0x52, 0xf3, 0xef, 0x53, 0x22, 0x3e | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –Call Blt() to verify that the pixels EfiBltVideoToBltBuffer retrieves are the same as the pixels EfiBltVideoFill fills. | For valid graphics mode number from 0 to *MaxMode*-1:  •To select a different valid parameter (*SourceX*, *SourceY*, *DestinationX*, *DestinationY*, *Width*, *Height*, *Delta*):  Call Blt() with a *BltOperation* value of EfiBltVideoFill to write data from the *BltBuffer* pixel (0, 0) directly to every pixel of the video display rectangle (*DestinationX*, *DestionationY*) (*DestionationX*+*Width*, *DestionationY*+*Height*). The returned status should be EFI\_SUCCESS.  Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer to retrieve the rectangles drawn by the last EfiBltVideoFill operation. The returned status should be EFI\_SUCCESS.  All of the retrieved rectangles should be the same pixel used in the last EfiBltVideoFill operation. |
| 5.6.6.3.12 | 0x1fc521b0, 0x63c1, 0x4f42, 0xb8, 0x14, 0x06, 0x8a, 0x6c, 0x9c, 0x3e, 0x29 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltVideoFill operation should fill the graphic screen with pixels. | For valid graphics mode number from 0 to *MaxMode*-1:  •To select a different valid parameter (*SourceX*, *SourceY*, *DestinationX*, *DestinationY*, *Width*, *Height*, *Delta*):  Call Blt()with a *BltOperation* value of EfiBltVideoFill to write data from the *BltBuffer* pixel (0, 0) directly to every pixel of the video display rectangle (*DestinationX*, *DestionationY*) (*DestionationX*+*Width*, *DestionationY*+*Height*). The returned status should be EFI\_SUCCESS.  Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer to retrieve the rectangles drawn by the last EfiBltVideoFill operation. The returned status should be EFI\_SUCCESS.  All of the retrieved rectangles should be the same pixel used in the last EfiBltVideoFill operation. |
| 5.6.6.3.13 | 0x04fd0571, 0xf3eb, 0x4d69, 0xb2, 0xd2, 0x5c, 0x4f, 0xfb, 0x10, 0x5a, 0xc3 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltVideoToBltBuffer operation should retrieve the pixels from the video memory to buffer. | For valid graphics mode number from 0 to *MaxMode*-1:  •To select a different valid parameter (*SourceX*, *SourceY*, *DestinationX*, *DestinationY*, *Width*, *Height*, *Delta*):  Call Blt() with a *BltOperation* value EfiBltVideoFill to write data from the *BltBuffer* pixel (0, 0) directly to every pixel of the video display rectangle (*DestinationX*, *DestionationY*) (*DestionationX*+*Width*, *DestionationY*+*Height*). The returned status should be EFI\_SUCCESS.  Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer to retrieve the rectangles drawn by the last EfiBltVideoFill operation. The returned status should be EFI\_SUCCESS.  All of the retrieved rectangles should be the same pixel used in the last EfiBltVideoFill operation. |
| 5.6.6.3.14 | 0x5bee154c, 0xe519, 0x4be4, 0xaf, 0x8c, 0xb4, 0x18, 0x8e, 0x79, 0xb4, 0xbf | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltBufferToVideo operation should draw the bitmap from the specified buffer to the video screen. | For valid graphics mode number from 0 to *MaxMode*-1:  • To select a different valid parameter (*SourceX*, *SourceY*, *DestinationX*, *DestinationY*, *Width*, *Height*, *Delta*):  Load a bitmap from the prepared buffer and call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display it in the video. The returned status should be EFI\_SUCCESS.  Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer to read data from the video display rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) and place it in the BltBuffer rectangle (*DestinationX*, *DestionationY*) (*DestionationX*+*Width*, *DestionationY*+*Height*). The returned status should be EFI\_SUCCESS. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to write data from the BltBuffer rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) directly to video display rectangle (*DestinationX*, *DestionationY*) (*DestionationX*+*Width*, *DestionationY*+*Height*). – Source and Destination should reverse with that of EfiBltVideoToBltBuffer operation. The returned status should be EFI\_SUCCESS.  Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer to read data from the video display rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) and place it in BltBuffer2 rectangle (0, 0) (*Width*, *Height*) with a delta value of 0.The returned status should be EFI\_SUCCESS.  The BltBuffer and BltBuffer2 should be the same. |
| 5.6.6.3.15 | 0xf9e726c1, 0x1346, 0x419e, 0x90, 0x8a, 0x66, 0xc4, 0x49, 0x8c, 0xfd, 0x71 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltVideoToBltBuffer operation should retrieve the pixels from the video to the buffer *BltBuffer*. | For valid graphics mode number from 0 to *MaxMode*-1:  •To select a different valid parameter (*SourceX*, *SourceY*, *DestinationX*, *DestinationY*, *Width*, *Height*, *Delta*):  Load a bitmap from the prepared buffer and call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display it in the video. The returned status should be EFI\_SUCCESS.  Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer to read data from the video display rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) and place it in the *BltBuffer* rectangle (*DestinationX*, *DestionationY*) (*DestionationX*+*Width*, *DestionationY*+*Height*). The returned status should be EFI\_SUCCESS. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to write data from the *BltBuffer* rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) directly to video display rectangle (*DestinationX*, *DestionationY*) (*DestionationX*+*Width*, *DestionationY*+*Height*). – Source and Destination should be the reverse of the EfiBltVideoToBltBuffer operation. The returned status should be EFI\_SUCCESS.  Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer to read data from the video display rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) and place it in the BltBuffer2 rectangle (0, 0) (*Width*, *Height*) with a delta value of 0.The returned status should be EFI\_SUCCESS.  The BltBuffer and BltBuffer2 should be the same. |
| 5.6.6.3.16 | 0x00f74a1b, 0x4599, 0x45b7, 0xb6, 0xf7, 0x13, 0xf2, 0xcb, 0xd8, 0x6c, 0xe6 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltBufferToVideo operation should write data to the video screen. | For valid graphics mode number from 0 to *MaxMode*-1:  •To select a different valid parameter (*SourceX*, *SourceY*, *DestinationX*, *DestinationY*, *Width*, *Height*, *Delta*):  Load a bitmap from the prepared buffer and call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display it in the video. The returned status should be EFI\_SUCCESS.  Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer to read data from the video display rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) and place it in the *BltBuffer* rectangle (*DestinationX*, *DestionationY*) (*DestionationX*+*Width*, *DestionationY*+*Height*). The returned status should be EFI\_SUCCESS. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to write data from the *BltBuffer* rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) directly to the video display rectangle (*DestinationX*, *DestionationY*) (*DestionationX*+*Width*, *DestionationY*+*Height*). – Source and Destination should be the reverse of the FeiBltVideoToBltBuffer operation. The returned status should be EFI\_SUCCESS.  Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer to read data from the video display rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) and place it in the BltBuffer2 rectangle (0, 0) (*Width*, *Height*) with a *Delta* value of 0.The returned status should be EFI\_SUCCESS.  The BltBuffer and BltBuffer2 should be the same. |
| 5.6.6.3.17 | 0x26da6582, 0x8b82, 0x4bd2, 0xac, 0x3a, 0x6e, 0x37, 0x85, 0x4f, 0xd8, 0x21 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltVideoToBltBuffer operation should retrieve the pixels from the video to another buffer BltBuffer2. | For valid graphics mode number from 0 to *MaxMode*-1:  •To select a different valid parameter (*SourceX*, *SourceY*, *DestinationX*, *DestinationY*, *Width*, *Height*, *Delta*):  Load a bitmap from the prepared buffer and call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display it in the video. The returned status should be EFI\_SUCCESS.  Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer to read data from the video display rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) and place it in the *BltBuffer* rectangle (*DestinationX*, *DestionationY*) (*DestionationX*+*Width*, *DestionationY*+*Height*). The returned status should be EFI\_SUCCESS. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to write data from the BltBuffer rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) directly to the video display rectangle (*DestinationX*, *DestionationY*) (*DestionationX*+*Width*, *DestionationY*+*Height*). – Source and Destination should be the reverse of the EfiBltVideoToBltBuffer operation. The returned status should be EFI\_SUCCESS.  Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer to read data from the video display rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) and place it in the BltBuffer2 rectangle (0, 0) (*Width*, *Height*) with a *Delta* value of 0.The returned status should be EFI\_SUCCESS.  The BltBuffer and BltBuffer2 should be the same. |
| 5.6.6.3.18 | 0x0aaf7f4e, 0x1794, 0x403c, 0xb3, 0xb0, 0x18, 0xf5, 0xe4, 0xd3, 0xc4, 0xea | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –Verify if the pixels retrieved from the first operation of EfiBltVideoToBltBuffer are the same as the pixels retrieved from the second operation. | For valid graphics mode number from 0 to *MaxMode*-1:  •To select a different valid parameter (*SourceX*, *SourceY*, *DestinationX*, *DestinationY*, *Width*, *Height*, *Delta*):  Load a bitmap from the prepared buffer and call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display it in the video. The returned status should be EFI\_SUCCESS.  Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer to read data from the video display rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) and place it in the *BltBuffer* rectangle (*DestinationX*, *DestionationY*) (*DestionationX*+*Width*, *DestionationY*+*Height*). The returned status should be EFI\_SUCCESS. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to write data from the BltBuffer rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) directly to the video display rectangle (*DestinationX*, *DestionationY*) (*DestionationX*+*Width*, *DestionationY*+*Height*). – Source and Destination should be the reverse of the EfiBltVideoToBltBuffer operation. The returned status should be EFI\_SUCCESS.  Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer to read data from the video display rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) and place it in the BltBuffer2 rectangle (0, 0) (*Width*, *Height*) with a delta value of 0.The returned status should be EFI\_SUCCESS.  The BltBuffer and BltBuffer2 should be the same. |
| 5.6.6.3.19 | 0x2a79335b, 0xafc3, 0x4ccf, 0x9b, 0xa4, 0x91, 0x9b, 0xe4, 0xb8, 0xbe, 0xfc | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltBufferToVideo operation should draw the bitmap from the specified buffer to the video screen. | For valid graphics mode number from 0 to *MaxMode*-1:  •To select a different valid parameter (*SourceX*, *SourceY*, *DestinationX*, *DestinationY*, *Width*, *Height*, *Delta*):  Load a bitmap from the prepared buffer and call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display it in the video. The returned status should be EFI\_SUCCESS.  Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer to save the original screen and read data from the video display rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) and place it in the *BltBuffer* rectangle (0, 0) (*Width*, *Height*) with a delta value of 0.The returned status should be EFI\_SUCCESS. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltVideoToVideo to copy data from the video display rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) to another display rectangle (*DestinationX*, *DestinationY*) (*DestinationX*+*Width*, *DestionationY*+*Height*). The returned status should be EFI\_SUCCESS.  Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer again to retrieve the area from the video display and read data from the video display rectangle (*DestinationX*, *DestinationY*) (*DestinationX*+*Width*, *DestinationY*+*Height*) and place it in BltBuffer2 rectangle (0, 0) (*Width*, *Height*) with a delta value of 0. The returned status should be EFI\_SUCCESS.  The BltBuffer and BltBuffer2 should be the same. |
| 5.6.6.3.20 | 0x3f4c2c88, 0xa1f8, 0x46f5, 0x9e, 0x5e, 0x67, 0x50, 0xb4, 0xae, 0x2b, 0x6f | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltVideoToBltBuffer operation should retrieve the pixels from the video display rectangle to the buffer *BltBuffer*. | For valid graphics mode number from 0 to *MaxMode*-1:  •To select a different valid parameter (*SourceX*, *SourceY*, *DestinationX*, *DestinationY*, *Width*, *Height*, *Delta*):  Load a bitmap from the prepared buffer and call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display it in the video. The returned status should be EFI\_SUCCESS.  Call Blt() with a BltOperation value of EfiBltVideoToBltBuffer to save the original screen and read data from the video display rectangle (SourceX, SourceY) (SourceX+Width, SourceY+Height) and place it in the BltBuffer rectangle (0, 0) (Width, Height) with a delta value of 0.The returned status should be EFI\_SUCCESS. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltVideoToVideo to copy data from the video display rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) to another display rectangle (*DestinationX*, *DestinationY*) (*DestinationX*+*Width*, *DestionationY*+*Height*). The returned status should be EFI\_SUCCESS.  Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer again to retrieve the area from the video display and read data from the video display rectangle (*DestinationX*, *DestinationY*) (*DestinationX*+*Width*, *DestinationY*+*Height*) and place it in the BltBuffer2 rectangle (0, 0) (*Width*, *Height*) with a delta value of 0. The returned status should be EFI\_SUCCESS.  The BltBuffer and BltBuffer2 should be the same. |
| 5.6.6.3.21 | 0xa11dd47e, 0xf144, 0x460c, 0x9e, 0x18, 0x7e, 0xb7, 0xed, 0xda, 0xc0, 0x18 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltVideoToVideo operation should copy data from one video display rectangle to another. | For valid graphics mode number from 0 to *MaxMode*-1:  •To select a different valid parameter (*SourceX*, *SourceY*, *DestinationX*, *DestinationY*, *Width*, *Height*, *Delta*):  Load a bitmap from the prepared buffer and call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display it in the video. The returned status should be EFI\_SUCCESS.  Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer to save the original screen and read data from the video display rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) and place it in the *BltBuffer* rectangle (0, 0) (*Width*, *Height*) with a delta value of 0.The returned status should be EFI\_SUCCESS. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltVideoToVideo to copy data from the video display rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) to another display rectangle (*DestinationX*, Dest*i*nationY) (*DestinationX*+*Width*, *DestionationY*+*Height*). The returned status should be EFI\_SUCCESS.  Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer again to retrieve the area from the video display and read data from the video display rectangle (*DestinationX*, *DestinationY*) (*DestinationX*+*Width*, *DestinationY*+*Height*) and place it in the BltBuffer2 rectangle (0, 0) (*Width*, *Height*) with a delta value of 0. The returned status should be EFI\_SUCCESS.  The BltBuffer and BltBuffer2 should be the same. |
| 5.6.6.3.22 | 0xbe3e3046, 0x5aea, 0x48d0, 0x91, 0xc4, 0x62, 0xce, 0xff, 0x61, 0x3c, 0xec | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltVideoToBltBuffer operation should retrieve the pixels from the video display rectangle to another buffer BltBuffer2. | For valid graphics mode number from 0 to *MaxMode*-1:  •To select a different valid parameter (*SourceX*, *SourceY*, *DestinationX*, *DestinationY*, *Width*, *Height*, *Delta*):  Load a bitmap from the prepared buffer and call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display it in the video. The returned status should be EFI\_SUCCESS.  Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer to save the original screen and read data from the video display rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) and place it in the *BltBuffer* rectangle (0, 0) (*Width*, *Height*) with a delta value of 0.The returned status should be EFI\_SUCCESS. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltVideoToVideo to copy data from the video display rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) to another display rectangle (*DestinationX*, *DestinationY*) (*DestinationX*+*Width*, *DestionationY*+*Height*). The returned status should be EFI\_SUCCESS.  Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer again to retrieve the area from the video display and read data from the video display rectangle (*DestinationX*, *DestinationY*) (*DestinationX*+*Width*, *DestinationY*+*Height*) and place it in the BltBuffer2 rectangle (0, 0) (*Width*, *Height*) with a delta value of 0. The returned status should be EFI\_SUCCESS.  The BltBuffer and BltBuffer2 should be the same. |
| 5.6.6.3.23 | 0xed4e402a, 0x403c, 0x4071, 0x86, 0x93, 0x9d, 0x8d, 0x28, 0xf7, 0x83, 0xd9 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –Verify that the pixels EfiBltVideoToBltBuffer retrieves are the same as the ones the second operation retrieves. | For valid graphics mode number from 0 to *MaxMode*-1:  •To select a different valid parameter (*SourceX*, *SourceY*, *DestinationX*, *DestinationY*, *Width*, *Height*, *Delta*):  Load a bitmap from the prepared buffer and call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display it in the video. The returned status should be EFI\_SUCCESS.  Call Blt() with a *BltOperation* value EfiBltVideoToBltBuffer to save the original screen and read data from the video display rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) and place it in the *BltBuffer* rectangle (0, 0) (*Width*, *Height*) with a delta value of 0.The returned status should be EFI\_SUCCESS. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltVideoToVideo to copy data from the video display rectangle (*SourceX*, *SourceY*) (*SourceX*+*Width*, *SourceY*+*Height*) to another display rectangle (*DestinationX*, *DestinationY*) (*DestinationX*+*Width*, *DestionationY*+*Height*). The returned status should be EFI\_SUCCESS.  Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer again to retrieve the area from the video display and read data from the video display rectangle (*DestinationX*, *DestinationY*) (*DestinationX*+*Width*, *DestinationY*+*Height*) and place it in the BltBuffer2 rectangle (0, 0) (*Width*, *Height*) with a delta value of 0. The returned status should be EFI\_SUCCESS.  The BltBuffer and BltBuffer2 should be the same. |
| 5.6.6.3.24 | 0x3b54894e, 0x6383, 0x4dd5, 0x9e, 0x53, 0xbe, 0x6b, 0xc1, 0x1b, 0xd8, 0x94 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltBufferToVideo operation should draw the bitmap from the specified buffer to the video screen. | For valid graphics mode number from 0 to *MaxMode*-1:  •Prepare BltBuffer from a small BMP file.  •Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the display.  •Repeat the following actions with 2 times:  change the (*DestinationX*,*DestinationY*) from the upper left-hand corner to the upper right-hand corner, then down to the lower right-hand corner, then to the lower left-hand corner, then to the upper left-hand corner, i.e., rotate the (*DestinationX*,*DestinationY*) clockwise.  Call Blt()with a *BltOperation* value EfiBltBufferToVideo to display the BMP file stored in *BltBuffer*. The returned status should be EFI\_SUCCESS. |
|  |  |  | Call Blt() with a *BltOperation* value EfiBltVideoFill to clear the rectangle in the last EfiBltBufferToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** The rotation of (*DestinationX*, *DestinationY*) may have some distance between the two coordinates. In addition, the (*DestinationX*, *DestinationY*, *Width*, *Height*) should not exceed the boundary of the current screen mode.  •Prompt the user to judge whether the BMP file rotates correctly. |
| 5.6.6.3.25 | 0xd0869ac8, 0x1d16, 0x4657, 0xae, 0xf2, 0x06, 0xc3, 0x49, 0x82, 0x1d, 0x55 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltVideoFill operation should fill the graphic screen with pixels. | For valid graphics mode number from 0 to *MaxMode*-1:  •Prepare *BltBuffer* from a small BMP file.  •Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the display.  •Repeat following actions with 2 times:  change the (*DestinationX*,*DestinationY*) from the upper left-hand corner to the upper right-hand corner, then down to the lower right-hand corner, then to the lower left-hand corner, then to the upper left-hand corner, i.e., rotate the (*DestinationX*,*DestinationY*) clockwise.  Call Blt() with a BltOperation value of EfiBltBufferToVideo to display the BMP file stored in *BltBuffer*. The returned status should be EFI\_SUCCESS. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the rectangle in the last EfiBltBufferToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** the rotation of (*DestinationX*, *DestinationY*) may have some distance between two coordinate. In addition, the (*DestinationX*, *DestinationY*, *Width*, *Height*) should avoid exceeding the boundary of the current screen mode.  •Prompt the user to judge whether the BMP file rotates correctly. |
| 5.6.6.3.26 | 0x1f026b26, 0x36fd, 0x4f1c, 0x95, 0x4c, 0x16, 0x0f, 0x9f, 0x98, 0x49, 0xd1 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltBufferToVideo operation should draw the bitmap from the specified buffer to the video screen. | For valid graphics mode number from 0 to *MaxMode*-1:  •Prepare *BltBuffer* from a small BMP file.  •Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the display.  •Repeat following actions with 2 times:  change the (*DestinationX*,*DestinationY*) from the upper left-hand corner to the upper right-hand corner, then down to the lower right-hand corner, then to the lower left-hand corner, then to the upper left-hand corner, i.e., rotate the (*DestinationX*,*DestinationY*) clockwise.  Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display the BMP file stored in *BltBuffer*. The returned status should be EFI\_SUCCESS. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the rectangle in the last EfiBltBufferToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** the rotation of (*DestinationX*, *DestinationY*) may have some distance between two coordinate. In addition, the (*DestinationX*, *DestinationY*, *Width*, *Height*) should avoid exceeding the boundary of the current screen mode.  •Prompt the user to judge whether the BMP file rotates correctly. |
| 5.6.6.3.27 | 0xd0bfb3c3, 0x54df, 0x4c07, 0x8e, 0x5c, 0x7a, 0x19, 0xa3, 0x5b, 0x5c, 0x0c | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltVideoFill operation should fill the graphic screen with pixels. | For valid graphics mode number from 0 to *MaxMode*-1:  •Prepare *BltBuffer* from a small BMP file.  •Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the display.  •Repeat following actions with 2 times:  change the (*DestinationX*,*DestinationY*) from the upper left-hand corner to the upper right-hand corner, then down to the lower right-hand corner, then to the lower left-hand corner, then to the upper left-hand corner, i.e., rotate the (*DestinationX*,*DestinationY*) clockwise.  Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display the BMP file stored in *BltBuffer*. The returned status should be EFI\_SUCCESS. |
|  |  |  | Call Blt()with a *BltOperation* value of EfiBltVideoFill to clear the rectangle in the last EfiBltBufferToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** the rotation of (*DestinationX*, *DestinationY*) may have some distance between two coordinate. In addition, the (*DestinationX*, *DestinationY*, *Width*, *Height*) should avoid exceeding the boundary of the current screen mode.  •Prompt the user to judge whether the BMP files rotates correctly. |
| 5.6.6.3.28 | 0xfde7edd9, 0x1486, 0x45e9, 0xae, 0x06, 0x31, 0xe8, 0xcb, 0x3f, 0xf3, 0x46 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltBufferToVideo operation should draw the bitmap from the specified buffer to the video screen. | For valid graphics mode number from 0 to *MaxMode*-1:  •Prepare *BltBuffer* from a small BMP file.  •Call Blt() with a *BltOperation* Value of EfiBltVideoFill to clear the display.  •Repeat following actions with 2 times:  change the (*DestinationX*,*DestinationY*) from the upper left-hand corner to the upper right-hand corner, then down to the lower right-hand corner, then to the lower left-hand corner, then to the upper left-hand corner, i.e., rotate the (*DestinationX*,*DestinationY*) clockwise.  Call Blt() with a *BltOperation* value EfiBltBufferToVideo to display the BMP file stored in *BltBuffer*. The returned status should be EFI\_SUCCESS. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the rectangle in the last EfiBltBufferToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** the rotation of (*DestinationX*, *DestinationY*) may have some distance between two coordinate. In addition, the (*DestinationX*, *DestinationY*, *Width*, *Height*) should avoid exceeding the boundary of the current screen mode.  •Prompt the user to judge whether the BMP file rotates correctly. |
| 5.6.6.3.29 | 0x538471f3, 0x8828, 0x4d1b, 0x8c, 0x2b, 0x01, 0x37, 0xe9, 0x4f, 0xae, 0xc9 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltVideoFill operation should fill the graphic screen with pixels. | For valid graphics mode number from 0 to *MaxMode*-1:  •Prepare BltBuffer from a small BMP file.  •Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the display.  •Repeat following actions with 2 times:  change the (*DestinationX*,*DestinationY*) from the upper left-hand corner to the upper right-hand corner, then down to the lower right-hand corner, then to the lower left-hand corner, then to the upper left-hand corner, i.e., rotate the (*DestinationX*,*DestinationY*) clockwise.  Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display the BMP file stored in *BltBuffer*. The returned status should be EFI\_SUCCESS. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the rectangle in the last EfiBltBufferToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** the rotation of (*DestinationX*, *DestinationY*) may have some distance between two coordinate. In addition, the (*DestinationX*, *DestinationY*, *Width*, *Height*) should avoid exceeding the boundary of the current screen mode.  •Prompt the user to judge whether the BMP file rotates correctly. |
| 5.6.6.3.30 | 0x30ef55c6, 0x62a2, 0x4f90, 0xb3, 0xf8, 0xf4, 0xf9, 0x1b, 0x94, 0xbf, 0x91 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltBufferToVideo operation should draw the bitmap from the specified buffer to the video screen. | For valid graphics mode number from 0 to *MaxMode*-1:  •Prepare BltBuffer from a small BMP file.  •Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the display.  •Repeat following actions with 2 times:  change the (*DestinationX*,*DestinationY*) from the upper left-hand corner to the upper right-hand corner, then down to the lower right-hand corner, then to the lower left-hand corner, then to the upper left-hand corner, i.e., rotate the (*DestinationX*,*DestinationY*) clockwise.  Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display the BMP file stored in *BltBuffer*. The returned status should be EFI\_SUCCESS. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the rectangle in the last EfiBltBufferToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** the rotation of (*DestinationX*, *DestinationY*) may have some distance between two coordinate. In addition, the (*DestinationX*, *DestinationY*, *Width*, *Height*) should avoid exceeding the boundary of the current screen mode.  •Prompt the user to judge whether the BMP file rotates correctly. |
| 5.6.6.3.31 | 0x2bb7feeb, 0x9b15, 0x4b27, 0x92, 0x61, 0xff, 0xa6, 0x9e, 0xcf, 0x0a, 0x00 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltVideoFill operation should fill the graphic screen with pixels. | For valid graphics mode number from 0 to MaxMode-1:  •Prepare BltBuffer from a small BMP file.  •Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the display.  •Repeat following actions, change the (*DestinationX*,*DestinationY*) from the upper left-hand corner to the upper right-hand corner, then down to the lower right-hand corner, then to the lower left-hand corner, then to the upper left-hand corner, i.e., rotate the (*DestinationX*,*DestinationY*) clockwise:  Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display the BMP file stored in *BltBuffer*. The returned status should be EFI\_SUCCESS. |
|  |  |  | Call Blt() with a *BltOperation* value EfiBltVideoFill to clear the rectangle in the last call of EfiBltBufferToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** The rotation of (*DestinationX*, *DestinationY*) may have some distance between two coordinate. In addition, the (*DestinationX*, *DestinationY*, *Width*, *Height*) should avoid exceeding the boundary of the current screen mode.  •Prompt the user to judge whether the BMP file rotates correctly. |
| 5.6.6.3.32 | 0x3bb9ebcc, 0x370a, 0x4c02, 0xb2, 0x0d, 0x1f, 0x86, 0x5a, 0x98, 0xaa, 0x15 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltBufferToVideo operation should draw the bitmap from the specified buffer to the video screen. | For valid graphics mode number from 0 to *MaxMode*-1:  •Prepare some *BltBuffer* from a small BMP file.  •Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the display.  •Repeat following actions, change the (*DestinationX*,*DestinationY*) from the upper left-hand corner to the upper right-hand corner, then down to the lower right-hand corner, then to the lower left-hand corner, then to the upper left-hand corner, i.e., rotate the (*DestinationX*,*DestinationY*) clockwise:  Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display the BMP file stored in *BltBuffer*. The returned status should be EFI\_SUCCESS. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the rectangle in the last EfiBltBufferToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** The rotation of (*DestinationX*, *DestinationY*) may have some distance between two coordinate. In addition, the (*DestinationX*, *DestinationY*,*Width*, *Height*) should avoid exceeding the boundary of the current screen mode.  •Prompt the user to judge whether the BMP file rotates correctly. |
| 5.6.6.3.33 | 0xb904f2be, 0x720e, 0x4d9b, 0x86, 0x72, 0xd7, 0x84, 0x6b, 0xbc, 0x53, 0xea | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltBufferToVideo operation should draw the bitmap from the specified buffer to the video screen. | For valid graphics mode number from 0 to *MaxMode*-1:  •Prepare some *BltBuffer* from a small BMP file.  •Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the display. The returned status should be EFI\_SUCCESS, and the display should be cleaned.  •Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display the BMP file.  •Repeat following actions, change the  (*DestinationX*,*DestinationY*) from the upper left-hand corner to the upper right-hand corner, then down to the lower right-hand corner, then to the lower left-hand corner, then to the upper left-hand corner, i.e., rotate the (*DestinationX*,*DestinationY*) clockwise:  Call Blt() with a *BltOperation* value of EfiBltVideoToVideo to copy the video drawn by the last EfiBltVideoToVideo or fiBltBufferToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** The two rectangles from the EfiBltVideoToVideo operation should not overlap. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the source rectangle in the last EfiBltVideoToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** The rotation of (*DestinationX*, *DestinationY*) may have some distance between two coordinate. In addition, the (*DestinationX*, *DestinationY*, *Width*, *Height*) should avoid exceeding the boundary of the current screen mode.  •Prompt the user to judge whether the BMP file rotates correctly. |
| 5.6.6.3.34 | 0x53748ffc, 0xaff8, 0x4cc9, 0x83, 0xab, 0xc7, 0x09, 0xe1, 0x59, 0x1c, 0xed | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltVideoToVideo operation should copy data from one video display rectangle to another. | For valid graphics mode number from 0 to *MaxMode*-1:  •Prepare some *BltBuffer* from a small BMP file.  •Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the display. The returned status should be EFI\_SUCCESS, and the display should be cleaned.  •Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display the BMP file.  •Repeat following actions, change the  (*DestinationX*,*DestinationY*) from the upper left-hand corner to the upper right-hand corner, then down to the lower right-hand corner, then to the lower left-hand corner, then to the upper left-hand corner, i.e., rotate the (*DestinationX*,*DestinationY*) clockwise:  Call Blt() with a *BltOperation* value of EfiBltVideoToVideo to copy the video drawn by the last EfiBltVideoToVideo or EfiBltBufferToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** The two rectangles from the EfiBltVideoToVideo operation should not overlap. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the source rectangle in the last call of EfiBltVideoToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** the rotation of (*DestinationX*, *DestinationY*) may have some distance between two coordinate. In addition, the (*DestinationX*, *DestinationY*, *Width*, *Height*) should avoid exceeding the boundary of the current screen mode.  •Prompt the user to judge whether the BMP file rotates correctly. |
| 5.6.6.3.35 | 0x4acd2d08, 0x01dd, 0x411f, 0xa6, 0xe2, 0xf3, 0x6f, 0x9f, 0x4b, 0x03, 0xb0 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltVideoFill operation should fill the graphic screen with pixels. | For valid graphics mode number from 0 to *MaxMode*-1:  •Prepare some *BltBuffer* from a small BMP file.  •Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the display. The returned status should be EFI\_SUCCESS, and the display should be cleaned.  •Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display the BMP file.  •Repeat following actions, change the  (*DestinationX*,*DestinationY*) from the upper left-hand corner to the upper right-hand corner, then down to the lower right-hand corner, then to the lower left-hand corner, then to the upper left-hand corner, i.e., rotate the (*DestinationX*,*DestinationY*) clockwise:  Call Blt() with a *BltOperation* value of EfiBltVideoToVideo to copy the Video drawn by the last EfiBltVideoToVideo or EfiBltBufferToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** The two rectangles from the EfiBltVideoToVideo operation should not overlap. |
|  |  |  | Call Blt() with a *BltOperation* value of *EfiBltVideoFill* to clear the source rectangle in the last EfiBltVideoToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** The rotation of (DestinationX, DestinationY) may have some distance between two coordinate. In addition, the (DestinationX, DestinationY, Width, Height) should avoid exceeding the boundary of the current screen mode.  •Prompt the user to judge whether the BMP file rotates correctly. |
| 5.6.6.3.36 | 0xb11e8ade, 0x0c54, 0x4963, 0x89, 0x66, 0xa0, 0x4a, 0x50, 0x40, 0x1c, 0x7b | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltVideoToVideo operation should copy data from one video display rectangle to another. | For valid graphics mode number from 0 to *MaxMode*-1:  •Prepare some *BltBuffer* from a small BMP file.  •Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the display. The returned status should be EFI\_SUCCESS, and the display should be cleaned.  •Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display the BMP file.  •Repeat following actions, change the  (*DestinationX*,*DestinationY*) from the upper left-hand corner to the upper right-hand corner, then down to the lower right-hand corner, then to lower the left-hand corner, then to upper the left-hand corner, i.e., rotate the (*DestinationX*,*DestinationY*) clockwise:  Call Blt() with a *BltOperation* value of EfiBltVideoToVideo to copy the Video drawn by the last EfiBltVideoToVideo or EfiBltBufferToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** The two rectangles from the EfiBltVideoToVideo operation should not overlap. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the source rectangle in the last EfiBltVideoToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** the rotation of (*DestinationX*, *DestinationY*) may have some distance between two coordinate. In addition, the (*DestinationX*, *DestinationY*, *Width*, *Height*) should avoid exceeding the boundary of the current screen mode.  •Prompt the user to judge whether the BMP rotates correctly. |
| 5.6.6.3.37 | 0xfa43d810, 0x7501, 0x481f, 0xbd, 0xcd, 0xc1, 0x06, 0x57, 0x94, 0x84, 0x9a | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltVideoFill operation should fill the graphic screen with pixels. | For valid graphics mode number from 0 to *MaxMode*-1:  •Prepare some *BltBuffer* from a small BMP file.  •Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the display. The returned status should be EFI\_SUCCESS, and the display should be cleaned.  •Call Blt() with a *BltOperation* vlue of EfiBltBufferToVideo to display the BMP file.  •Repeat following actions, change the  (*DestinationX*,*DestinationY*) from the upper left-hand corner to the upper right-hand corner, then down to the lower right-hand corner, then to the lower left-hand corner, then to upper the left-hand corner, i.e., rotate the (*DestinationX*,*DestinationY*) clockwise:  Call Blt() with a *BltOperation* value of EfiBltVideoToVideo to copy the Video drawn by the last EfiBltVideoToVideo or EfiBltBufferToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** The two rectangles from the EfiBltVideoToVideo operation should not overlap. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the source rectangle in the last EfiBltVideoToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** the rotation of (*DestinationX*, *DestinationY*) may have some distance between two coordinate. In addition, the (*DestinationX*, *DestinationY*, *Width*, *Height*) should avoid exceeding the boundary of the current screen mode.  •Prompt the user to judge whether the BMP file rotates correctly. |
| 5.6.6.3.38 | 0x94989a37, 0x3941, 0x4cd8, 0x97, 0x0b, 0x14, 0xfa, 0x46, 0xb6, 0x07, 0x16 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltVideoToVideo operation should copy data from one video display rectangle to another. | For valid graphics mode number from 0 to *MaxMode*-1:  •Prepare some *BltBuffer* from a small BMP file.  •Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the display. The returned status should be EFI\_SUCCESS, and the display should be cleaned.  •Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display the BMP file.  •Repeat following actions, change the  (*DestinationX*,*DestinationY*) from the upper left-hand corner to the upper right-hand corner, then down to the lower right-hand corner, then to the lower left-hand corner, then to the upper left-hand corner, i.e., rotate the (*DestinationX*,*DestinationY*) clockwise:  Call Blt() with a *BltOperation* value of EfiBltVideoToVideo to copy the Video drawn by the last EfiBltVideoToVideo or EfiBltBufferToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** The two rectangles from the EfiBltVideoToVideo operation should not overlap. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the source rectangle in the last call of EfiBltVideoToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** the rotation of (*DestinationX*, *DestinationY*) may have some distance between two coordinate. In addition, the (*DestinationX*, *DestinationY*, *Width*, *Height*) should avoid exceeding the boundary of the current screen mode.  •Prompt the user to judge whether the BMP file rotates correctly. |
| 5.6.6.3.39 | 0x4dde309d, 0xaf32, 0x4a35, 0x91, 0x5a, 0x41, 0xcb, 0xb0, 0x18, 0x7c, 0x29 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltVideoFill operation should fill the graphic screen with pixels. | For valid graphics mode number from 0 to *MaxMode*-1:  •Prepare some *BltBuffer* from a small BMP file.  •Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the display. The returned status should be EFI\_SUCCESS, and the display shoul be cleaned.  •Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display the BMP file.  •Repeat following actions, change the  (*DestinationX*,*DestinationY*) from the upper left-hand corner to the upper right-hand corner, then down to the lower right-hand corner, then to the lower left-hand corner, then to the upper left-hand corner, i.e., rotate the (*DestinationX*,*DestinationY*) clockwise:  Call Blt() with a *BltOperation* value of EfiBltVideoToVideo to copy the Video drawn by the last EfiBltVideoToVideo or EfiBltBufferToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** The two rectangles from the EfiBltVideoToVideo operation should not overlap. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the source rectangle in the last EfiBltVideoToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** the rotation of (*DestinationX*, *DestinationY*) may have some distance between two coordinate. In addition, the (*DestinationX*, *DestinationY*, *Width*, *Height*) should avoid exceeding the boundary of the current screen mode.  •Prompt the user to judge whether the BMP file rotates correctly. |
| 5.6.6.3.40 | 0xaa6b7386, 0x0537, 0x4762, 0xa1, 0x43, 0xca, 0xde, 0xb7, 0x55, 0x15, 0xc7 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltVideoToVideo operation should copy data from one video display rectangle to another. | For valid graphics mode number from 0 to *MaxMode*-1:  •Prepare some *BltBuffer* from a small BMP file.  •Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the display. The returned status should be EFI\_SUCCESS, and the display should be cleaned.  •Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display the BMP file.  •Repeat following actions, change the  (*DestinationX*,*DestinationY*) from the upper left-hand corner to the upper right-hand corner, then down to the lower right-hand corner, then to the lower left-hand corner, then to the upper left-hand corner, i.e., rotate the (*DestinationX*,*DestinationY*) clockwise:  Call Blt() with a *BltOperation* value of EfiBltVideoToVideo to copy the Video drawn by the last EfiBltVideoToVideo or EfiBltBufferToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** The two rectangles from the EfiBltVideoToVideo operation should not overlap. |
|  |  |  | Call Blt() with a *BltOperation* value EfiBltVideoFill to clear the source rectangle in the last EfiBltVideoToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** the rotation of (*DestinationX*, *DestinationY*) may have some distance between two coordinate. In addition, the (*DestinationX*, *DestinationY*, *Width*, *Height*) should avoid exceeding the boundary of the current screen mode.  •Prompt the user to judge whether the BMP file rotates correctly. |
| 5.6.6.3.41 | 0xb751208f, 0x10eb, 0x47eb, 0x9c, 0x73, 0x15, 0x08, 0xb8, 0xc9, 0xcd, 0xbe | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltVideoFill operation should fill the graphic screen with pixels. | For valid graphics mode number from 0 to *MaxMode*-1:  •Prepare some *BltBuffer* from a small BMP file.  •Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the display. The returned status should be EFI\_SUCCESS, and the display should be cleaned.  •Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display the BMP file.  •Repeat following actions, change the  (*DestinationX*,*DestinationY*) from the upper left-hand corner to the upper right-hand corner, then down to the lower right-hand corner, then to the lower left-hand corner, then to the upper left-hand corner, i.e., rotate the (*DestinationX*,*DestinationY*) clockwise:  Call Blt() with a *BltOperation* value EfiBltVideoToVideo to copy the Video drawn by the last EfiBltVideoToVideo or EfiBltBufferToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** The two rectangles from theEfiBltVideoToVideo operation should not overlap. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the source rectangle in the last EfiBltVideoToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** the rotation of (*DestinationX*, *DestinationY*) may have some distance between two coordinate. In addition, the (*DestinationX*, *DestinationY*, *Width*, *Height*) should avoid exceeding the boundary of the current screen mode.  •Prompt the user to judge whether the BMP file rotates correctly. |
| 5.6.6.3.42 | 0x57b7debf, 0xb831, 0x40d1, 0x8b, 0xa0, 0xa6, 0x57, 0x7b, 0x92, 0xe2, 0x53 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –check logo rotatation correctly from user’s view. | For valid graphics mode number from 0 to *MaxMode*-1:  •Prepare some *BltBuffer* from a small BMP file.  •Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the display. The returned status should be EFI\_SUCCESS, and the display should be cleaned.  •Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display the BMP file.  •Repeat following actions, change the  (*DestinationX*,*DestinationY*) from the upper left-hand corner to the upper right-hand corner, then down to the lower right-hand corner, then to the lower left-hand corner, then to the upper left-hand corner, i.e., rotate the (*DestinationX*,*DestinationY*) clockwise:  Call Blt() with a *BltOperation* value of EfiBltVideoToVideo to copy the Video drawn by the last EfiBltVideoToVideo or EfiBltBufferToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** The two rectangles from the EfiBltVideoToVideo operation should not overlap. |
|  |  |  | Call Blt() with a *BltOperation* value of EfiBltVideoFill to clear the source rectangle in the last call of EfiBltVideoToVideo operation. The returned status should be EFI\_SUCCESS.  **Note:** the rotation of (*DestinationX*, *DestinationY*) may have some distance between two coordinate. In addition, the (*DestinationX*, *DestinationY*, *Width*, *Height*) should avoid exceeding the boundary of the current screen mode.  •Prompt user to judge whether the rotation of the BMP file correctly. |
| 5.6.6.3.43 | 0x8971c5fe, 0x02c6, 0x4ada, 0xab, 0x30, 0x36, 0xc5, 0xa7, 0xd9, 0xdc, 0x01 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltBufferToVideo operation should draw the bitmap from the specified buffer to the video screen. | For valid graphics mode number from 0 to *MaxMode*-1:  •Prepare some *BltBuffer* from a small BMP file.  •Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display BMP file to some position of the screen.  •Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer to retrieve the whole screen to a large *BltBuffer*. The return status  should be EFI\_SUCCESS.  •Change *SourceX* from 0 to HorizontalResolution step by step, and change *SourceY* from 0 to VerticalResolution step by step, carry out following action:  For small *Width*, *Height*, and *BltBuffer*, call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer. The returned status should be EFI\_SUCCESS. The small *BltBuffer* retrieved should be the same as the corresponding segment in the large *BltBuffer* standing for the whole screen buffer. |
| 5.6.6.3.44 | 0x03093b96, 0x2b15, 0x4008, 0xb7, 0xbf, 0x9f, 0x8c, 0x17, 0x41, 0x2d, 0xb3 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –EfiBltVideoToBltBuffer operation should retrieve the pixels from the video display rectangle to the specified buffer. | For valid graphics mode number from 0 to *MaxMode*-1:  •Prepare some *BltBuffer* from a small BMP file.  •Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display BMP file to some position of the screen.  •Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer to retrieve the whole screen to a large *BltBuffer*. The return status  should be EFI\_SUCCESS  •Change *SourceX* from 0 to HorizontalResolution step by step, and change *SourceY* from 0 to VerticalResolution step by step, carry out following action:  For small *Width*, *Height*,and *BltBuffer*, Call Blt()with a *BltOperation* value of EfiBltVideoToBltBuffer. The returned status should be EFI\_SUCCESS. The small *BltBuffer* retrieved should be the same as the corresponding segment in the large *BltBuffer* standing for the whole screen buffer. |
| 5.6.6.3.45 | 0x1ef36d93, 0x8591, 0x4172, 0x94, 0xfd, 0x93, 0x08, 0x54, 0x6e, 0x73, 0x11 | EFI\_GRAPHICS\_OUTPUT\_PROTOCOL.Blt –Blt/  EfiBltVideoToBltBuffe, Pixel verification. | For valid graphics mode number from 0 to *MaxMode*-1:  •Prepare some *BltBuffer* from a small BMP file.  •Call Blt() with a *BltOperation* value of EfiBltBufferToVideo to display BMP file to some position of the screen.  •Call Blt() with a *BltOperation* value of EfiBltVideoToBltBuffer to retrieve the whole screen to a large *BltBuffer*. The return status  should be EFI\_SUCCESS  •Change *SourceX* from 0 to HorizontalResolution step by step, and change *SourceY* from 0 to VerticalResolution step by step, carry out following action:  For mall *Width*, *Height*, and *BltBuffer*, call Blt()with a *BltOperation* value of EfiBltVideoToBltBuffer. The returned status should be EFI\_SUCCESS. The small *BltBuffer* retrieved should be the same as the corresponding segment in the large *BltBuffer* standing for the whole screen buffer. |

## EFI\_SIMPLE\_ TEXT\_INPUT\_EX\_PROTOCOL Test

Reference Document:

*UEFI Specification*, EFI\_SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL Section.

### Reset()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.7.1.1 | 0xc969bba7, 0xed63, 0x4235, 0x80, 0x46, 0xa1, 0x8c, 0xa2, 0x8a, 0x3f, 0x6a | SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL.Reset - Reset() returns EFI\_SUCCESS and ReadKeyStrokeEx return EFI\_NOT\_READY | It is a auto test. Call Reset(). EFI\_SUCCESS should be returned. Call ReadKeyStrokeEx(). EFI\_NOT\_READY should be returned. |
| 5.6.7.1.2 | 0x35381b6c, 0x1035, 0x4241, 0x95, 0x80, 0x21, 0x25, 0x3b, 0x78, 0x60, 0x8d | SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL.Reset - Reset() returns EFI\_SUCCESS and ReadKeyStrokeEx return EFI\_NOT\_READY | It is a manual test. Press a key. Call Reset(). EFI\_SUCCESS should be returned. Call ReadKeyStrokeEx(). EFI\_NOT\_READY should be returned. |

### ReadKeyStrokeEx ()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.7.2.1 | 0x89854ccd, 0xa672, 0x4856, 0xb7, 0x6c, 0xb1, 0x66, 0xc5, 0x64, 0x2f, 0x9a | SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL.ReadKeyStorkeEx - ReadKeyStorkeEx() returns EFI\_INVALID\_PARAMETER with KeyData being NULL. | Call Reset() first, and EFI\_SUCCESS should be returned.  Call ReadKeyStrokeEx() with KeyData being NULL. EFI\_INVALID\_PARAMETER should be returned. |
| 5.6.7.2.2 | 0x5d141dc0, 0xded6, 0x4e01, 0xa9, 0x8b, 0x55, 0x1f, 0x3e, 0xe3, 0x59, 0x4d | SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL.ReadKeyStorkeEx - ReadKeyStorkeEx() returns EFI\_NOT\_READY with console just been reseted. | Call Reset() first, and EFI\_SUCCESS should be returned.  Call ReadKeyStrokeEx() with valid parameter. EFI\_NOT\_READY should be returned. |
| 5.6.7.2.3 | 0x5eed7df1, 0x4630, 0x44e1, 0x97, 0xaa, 0xd3, 0x26, 0x82, 0x24, 0xc4, 0x30 | EFI\_SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL.ReadKeyStroke - ReadKeyStrokeEx() return EFI\_SUCCESS with key input | It is a manual test. Part 1:  Call Reset(). EFI\_SUCCESS should be returned. Press a key. Call ReadKeyStrokeEx() with valid parameter. EFI\_SUCCESS should be returned. |
| 5.6.7.2.4 | 0x3032721e, 0x8089, 0x49d4, 0x94, 0x5a, 0x46, 0x07, 0xdc, 0x05, 0xcf, 0x8d | EFI\_SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL.ReadKeyStroke - ReadKeyStroke() with key input, user's view | Part 2: Echo the key which is pressed. Tester decides the SUCCESS or Failure. |

### SetState()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.7.3.1 | 0x6647a0e7, 0x483c, 0x4777, 0xa9, 0x4b, 0xc8, 0xbc, 0xa3, 0xdf, 0xc7, 0x9c | SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL.SetState - SetState() returns EFI\_INVALID\_PARAMETER with KeyToggleState being NULL. | Call SetState() with KeyToggleState being NULL. Return status should be EFI\_INVALID\_PARAMETER. |
| 5.6.7.3.2 | 0x4c766c77, 0xdbf3, 0x4b3d, 0x82, 0x59, 0x81, 0xf8, 0xb8, 0xaa, 0x17, 0x75 | SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL.SetState - SetState() returns EFI\_UNSUPPORTED with KeyToggleState being a unsupported bit set. | Call SetState() with unsupported KeyToggleState. Return status should be EFI\_UNSUPPORTED. |
| 5.6.7.3.3 | 0x44bf142c, 0x72a9, 0x445e, 0xaf, 0x84, 0xaa, 0xc5, 0x96, 0xc6, 0x3f, 0xc8 | SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL.SetState - SetState() returns EFI\_UNSUPPORTED or EFI\_SUCCESS with a valid bit set | Call SetState() with valid KeyToogleState. The return status should be EFI\_UNSUPPORTED or EFI\_SUCCESS.  Press a key and call ReadKeyStrokeEx(). The KeyToggleState should be same as the State which be set. |

### 

### RegisterKeyNotify()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.7.4.1 | 0x27a40c7e, 0x119e, 0x451d, 0x84, 0x70, 0x1d, 0xc4, 0x52, 0x09, 0x2b, 0x0a | SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL.RegisterKeyNotify - RegisterKeyNotify() returns EFI\_INVALID\_PARAMETER with KeyData being NULL | Call RefisterKeyNotify() with NULL KeyData. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.6.7.4.2 | 0xb03a561d, 0x6339, 0x4035, 0xaf, 0xd5, 0xfa, 0x2e, 0xce, 0x16, 0x4b, 0xf9 | SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL.RegisterKeyNotify - RegisterKeyNotify() returns EFI\_INVALID\_PARAMETER with KeyNotificationFunction being NULL | Call RefisterKeyNotify() with NULL KeyNotificationFunction. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.6.7.4.3 | 0x5b22932e, 0xc24d, 0x45fe, 0x8b, 0xbd, 0x2e, 0x0e, 0x56, 0xfa, 0xc3, 0x16 | SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL.RegisterKeyNotify - RegisterKeyNotify() returns EFI\_INVALID\_PARAMETER with NotifyHandle being NULL. | Call RefisterKeyNotify() with NULL NotifyHandle. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.6.7.4.4 | 0x69a1c06c, 0x516e, 0x4595, 0xbe, 0x4f, 0x6b, 0x18, 0x58, 0xcc, 0x82, 0x3d | SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL.RegisterKeyNotify - RegisterKeyNotify() returns EFI\_SUCCESS. | It is a manual test. Part 1:  Call Reset(). The return status should be EFI\_SUCCESS.  Press a key, call ReadKeyStrokeEx() to get the key value. The return status should be EFI\_SUCCESS.  Call RegisterKeyNotify() with the key. The return status should EFI\_SUCCESS. |
| 5.6.7.4.5 | 0x6f509a8c, 0x0df2, 0x499d, 0x97, 0x56, 0x35, 0xbe, 0x3c, 0xcb, 0x21, 0xc4 | SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL.RegisterKeyNotify - RegisterKeyNotify() returns EFI\_SUCCESS and notify function has been invoked. | Part 2:  Call Reset(). The return status should be EFI\_SUCCESS.  Press the same key and check the the result of the notify function..  Call UnregisterKeyNotify(). |

### UnregisterKeyNotify()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.7.5.1 | 0xa5244802, 0xf4bf, 0x4e8a, 0xaa, 0x76, 0xe4, 0x87, 0x37, 0x43, 0xd8, 0xd1 | SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL.UnregisterKeyNotify - UnregisterKeyNotify() returns EFI\_INVALID\_PARAMETER with NotifyHandle being NULL | Call UnrefisterKeyNotify with NULL NotifyHandle. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.6.7.5.2 | 0x5d1c8b11, 0x326f, 0x4cf5, 0xb0, 0x3d, 0x89, 0xaa, 0x2f, 0xaf, 0x66, 0x42 | SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL.UnregisterKeyNotify - UnregisterKeyNotify() returns EFI\_INVALID\_PARAMETER with NotifyHandle not refer to a register notify function anymore. | Call RegisterKeyNotify() with valid parameter. Return status should be EFI\_SUCCESS.  Call UnregisterKeyNotify() twice. Return status should be EFI\_INVALID\_PARAMETER. |
| 5.6.7.5.3 | 0x5fe62478, 0x4614, 0x4430, 0xb9, 0xe9, 0x30, 0xe2, 0x12, 0x19, 0xeb, 0x35 | SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL.UnregisterKeyNotify - UnregisterKeyNotify() returns EFI\_INVALID\_PARAMETER with NotifyHandle being illegal format. | Call UnrefisterKeyNotify() with invalid NotifyHandle. The return status should be EFI\_INVALID\_PARAMETER. |
| 5.6.7.5.4 | 0xe305a4b5, 0x03c3, 0x43c4, 0x93, 0x16, 0x7d, 0x7a, 0xb3, 0x6a, 0x13, 0xa5 | SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL.UnregisterKeyNotify - UnregisterKeyNotify() returns EFI\_SUCCESS. | It is a manual test. Part 1:  Call Reset(). The return status should be EFI\_SUCCESS.  Press a key, call ReadKeyStrokeEx() to get the key value. The return status should be EFI\_SUCCESS.  Call RegisterKeyNotify() with the key. The return status should EFI\_SUCCESS.  Call UnregisterKeyNotify() with the valid NotifyHandle. The return status should be EFI\_SUCCESS. |
| 5.6.7.5.5 | 0x534369f7, 0x8399, 0x4353, 0x94, 0xad, 0xc4, 0x48, 0xfa, 0xda, 0xeb, 0x84 | SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL.UnregisterKeyNotify - UnregisterKeyNotify() returns EFI\_SUCCESS and notify function has not been invoked. | Part 2:  Call Reset(). The return status should be EFI\_SUCCESS.  Press the same key and check the the result of the notify function. |

## EFI\_ABSOLUTE\_POINTER\_PROTOCOL Test

Reference Document:

*UEFI Specification*, EFI\_ABSOLUTE\_POINTER\_PROTOCOL Section.

### Reset()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.8.1.1 | 0xa4e0b129, 0x4bd4, 0x4446, 0x8d, 0x32, 0xaa, 0x45, 0x64, 0xb2, 0x74, 0x6e | EFI\_ABSOLUTE\_POINTER\_PROTOCOL.Reset - Reset() with an ExtendedVerification value of FALSE returns EFI\_SUCCESS. If device error, should return EFI\_DEVICE\_ERROR | 1. Call Reset() with an ExtendedVerification value of FALSE. The return code should be EFI\_SUCCESS or EFI\_DEVICE\_ERROR. |
| 5.6.8.1.2 | 0xc246b3ff, 0xc1d5, 0x499b, 0x92, 0x87, 0x73, 0xf5, 0x88, 0xf6, 0xa9, 0x9f | EFI\_ABSOLUTE\_POINTER\_PROTOCOL.Reset - Reset() with an ExtendedVerification value of TRUE returns EFI\_SUCCESS. If device error, should return EFI\_DEVICE\_ERROR. | 1. Call Reset() with an ExtendedVerification value of TRUE. The return code should be EFI\_SUCCESS or EFI\_DEVICE\_ERROR. |
| 5.6.8.1.3 | 0xab689092, 0xc9e2, 0x4618, 0x90, 0xa8, 0x4, 0x74, 0xde, 0x94, 0x7c, 0x4e | EFI\_ABSOLUTE\_POINTER\_PROTOCOL.Reset - GetState() after Reset () return EFI\_UNSUPPORTED | 1. Call Reset() with an ExtendedVerification value of FALSE.  2. Call GetState(),the return code maybe EFI\_UNSUPPORTED. |
| 5.6.8.1.4 | 0x5c250202, 0xe791, 0x4cee, 0x86, 0x74, 0x4e, 0x3a, 0x43, 0xbc, 0x18, 0x15 | EFI\_ABSOLUTE\_POINTER\_PROTOCOL.Reset - Reset() with an ExtendedVerification value of TRUE returns EFI\_SUCCESS. | 1. Call Reset() with an ExtendedVerification value of TRUE. The return code should be EFI\_SUCCESS. |
| 5.6.8.1.5 | 0x39c1417d, 0xf6a2, 0x4a77, 0xbb, 0xcd, 0xe, 0xb8, 0xeb, 0x41, 0xe7, 0x52 | EFI\_ABSOLUTE\_POINTER\_PROTOCOL.Reset - GetState() after Reset () return 0 for all current movement. | 1. Call Reset() with an ExtendedVerification value of TRUE.  2. Call GetState(). if success, Current movement in X,Y and Z should be 0. |
| 5.6.8.1.6 | 0xd909148a, 0xd05a, 0x4694, 0xb4, 0xc4, 0xfc, 0x27, 0x87, 0x40, 0xce, 0x78 | EFI\_ABSOLUTE\_POINTER\_PROTOCOL.Reset - GetState() after Reset () return EFI\_UNSUPPORTED | 1. Call Reset() with an ExtendedVerification value of TRUE.  2. Call GetState(),the return code maybe EFI\_UNSUPPORTED. |

### GetState()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.6.8.2.1 | 0x5271062f, 0xdef9, 0x4d30, 0x84,0x3b, 0x8d,0x6e, 0x41,0x33, 0x13,0xf3 | EFI\_ABSOLUTE\_POINTER\_PROTOCOL.GetState - GetState() after Reset() returns 0 for all current movement. | 1. Call Reset() with an ExtendedVerification value of FALSE.  2. Call GetState(). if success, Current movement in X,Y and Z should be 0. |
| 5.6.8.2.2 | 0x7614c448, 0x12a0, 0x403d, 0x8a,0xde, 0x98,0x97, 0x51,0x7d, 0xd8,0x49 | EFI\_ABSOLUTE\_POINTER\_PROTOCOL.GetState - GetState() returns EFI\_NOT\_READY when there is no move since last call of GetState()or returns EFI\_DEVICE\_ERROR while a device error occurred.. | 1. Call Reset() with an ExtendedVerification value of FALSE.  2. Call GetState() .  3. Call GetState() again, the return code should be EFI\_NOT\_READY or EFI\_DEVICE\_ERROR. |
| 5.6.8.2.3 | 0x2f8f8711, 0x02dd, 0x411f, 0xaa,0xb5, 0x27,0xe1, 0x3a,0x6a, 0xb2,0x79 | EFI\_ABSOLUTE\_POINTER\_PROTOCOL.GetState - GetState() after Reset() returns 0 for all current movement. | 1. Call Reset() with an ExtendedVerification value of TRUE.  2. Call GetState(). if success, Current movement in X,Y and Z should be 0. |
| 5.6.8.2.4 | 0x3db7ea19, 0xda9d, 0x4760, 0xa7,0x43, 0x04,0xb4, 0x8d,0x14, 0x4e,0x90 | EFI\_ABSOLUTE\_POINTER\_PROTOCOL.GetState - GetState() returns EFI\_NOT\_READY when there is no move since last call of GetState()or returns EFI\_DEVICE\_ERROR while a device error occurred. | 1. Call Reset() with an ExtendedVerification value of TRUE.  2. Call GetState() .  3. Call GetState() again, the return code should be EFI\_NOT\_READY or EFI\_DEVICE\_ERROR. |