

NOTE: Usually on **Insyde & Phoenix BIOS**, EIST locked can be found in these modules:-

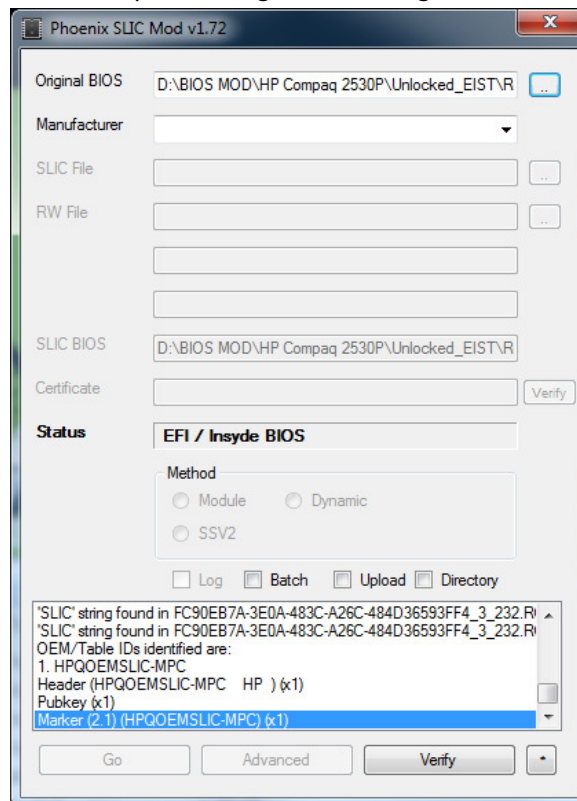
- MOD_5100.ROM (Phoenix)
- F7731B4C-58A2-4DF4-8980-5645D39ECE58.ff (Phoenix)
- F7731B4C-58A2-4DF4-8980-5645D39ECE58_X_XXX.ROM (Insyde)
- BIOSCOD0X.ROM (Phoenix)

On some BIOS, the lock also can be found in other modules too (BIOSCOD0X.ROM - X stands for 0, 1, 2 & so on). So, it is best to check all modules.

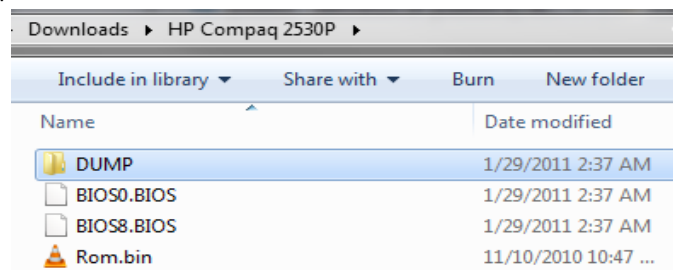
- Based on this information, you can focus on these three modules (see above).
- This example use HP Compaq **2530P**'s BIOS (sp49008_no_whitelist) downloaded from here:- <http://forums.mydigitallife.info/threads/7681-This-is-no-request-thread!-HP-COMPAQ-bioses-how-to-modify-the-bios?p=333358#post333358>

1) Decompress BIOS image using **Phoenix SLIC Mod** tool

- At the **Original BIOS** field, open the original BIOS image, in this example **Rom.bin**:-



- The decompressed BIOS modules are located in **DUMP** folder:-



2) Create a bash script:-

- Requirement: **Cygwin Bash Shell** or linux.
- On Phoenix BIOS, the BIOS modules are wrote in x86 instruction set except the EFI-based modules (e.g. F7731B4C-58A2-4DF4-8980-5645D39ECE58.ff) which wrote in x86_64 instruction set.
- On Insyde BIOS, the BIOS modules are EFI-based modules which wrote in x86_64 instruction set.
- Based on the above information, we'll need to disassemble the BIOS modules in 16-bit & 64-bit modes.
- The bash script below will disassemble all modules in 16-bit & 64-bit mode.
- Save to any name, for example **ndisasm_bat.sh**.
- Copy **ndisasm_bat.sh** to inside **DUMP** folder (where all BIOS modules are located).
- File with extensions **.16.DASM** & **.DASM** are disassembled file in 16-bit & 64-bit mode respectively.

```
#!/bin/bash
echo "--> disassembling *.ROM files"
for f in $( ls *.ROM ); do
    DIS=$(echo $f | sed 's/\(^[a-zA-Z0-9_-]*\)\\.([a-zA-Z]*)\\1\\.DASM/')
    DIS16=$(echo $f | sed 's/\(^[a-zA-Z0-9_-]*\)\\.([a-zA-Z]*)\\1\\.16.DASM/')
    echo "--> Creating disassembled file: $DIS"
    echo "ndisasm -a -b 64 $f > $DIS"
    ndisasm -a -p intel -b 64 $f > $DIS
    echo "--> Creating disassembled file: $DIS16"
    echo "ndisasm -a -b 16 $f > $DIS16"
    ndisasm -a -p intel -b 16 $f > $DIS16
done
for f in $( ls *.ff ); do
    DIS=$(echo $f | sed 's/\(^[a-zA-Z0-9_-]*\)\\.([a-zA-Z]*)\\1\\.DASM/')
    DIS16=$(echo $f | sed 's/\(^[a-zA-Z0-9_-]*\)\\.([a-zA-Z]*)\\1\\.16.DASM/')
    echo "--> Creating disassembled file: $DIS"
    echo "ndisasm -a -b 64 $f > $DIS"
    ndisasm -a -p intel -b 64 $f > $DIS
    echo "--> Creating disassembled file: $DIS16"
    echo "ndisasm -a -b 16 $f > $DIS16"
    ndisasm -a -p intel -b 16 $f > $DIS16
done
for f in $( ls *.PEI ); do
    DIS=$(echo $f | sed 's/\(^[a-zA-Z0-9_-]*\)\\.([a-zA-Z]*)\\1\\.DASM/')
    DIS16=$(echo $f | sed 's/\(^[a-zA-Z0-9_-]*\)\\.([a-zA-Z]*)\\1\\.16.DASM/')
    echo "--> Creating disassembled file: $DIS"
    echo "ndisasm -a -b 64 $f > $DIS"
    ndisasm -a -p intel -b 64 $f > $DIS
    echo "--> Creating disassembled file: $DIS16"
    echo "ndisasm -a -b 16 $f > $DIS16"
    ndisasm -a -p intel -b 16 $f > $DIS16
done
echo "--> Checking MSR register for EIST: mov ecx,0x1A0" > RESULTS.TXT
echo "--> Checking MSR register for EIST: mov ecx,0x1A0"
grep -i "mov ecx,0x1A0" *.DASM >> RESULTS.TXT
grep -i "mov ecx,0x1A0" *.DASM
```

- 3) The above script will filter all modules for “**mov ecx,0x1a0**”. The above script will produced output like this (example from HP Compaq 2530P’s BIOS modules):-

--> Checking MSR register for EIST: **mov ecx,0x1a0**

```

1547B4F3-3E8A-4FEF-81C8-328ED647AB1A_1_744.16.DASM:00001DF6 66B9A0010000    mov ecx,0x1a0
1547B4F3-3E8A-4FEF-81C8-328ED647AB1A_1_744.16.DASM:00001E20 66B9A0010000    mov ecx,0x1a0
1BA0062E-C779-4582-8566-336AE8F78F09_1_1006.DASM:0000047C B9A0010000    mov ecx,0x1a0
62D171CB-78CD-4480-8678-C6A2A797A8DE_3_484.DASM:0000251D B9A0010000    mov ecx,0x1a0
62D171CB-78CD-4480-8678-C6A2A797A8DE_3_484.DASM:00002538 B9A0010000    mov ecx,0x1a0
62D171CB-78CD-4480-8678-C6A2A797A8DE_3_484.DASM:00006030 B9A0010000    mov ecx,0x1a0
62D171CB-78CD-4480-8678-C6A2A797A8DE_3_484.DASM:0000606F B9A0010000    mov ecx,0x1a0
CA9D8617-D652-403B-B6C5-BA47570116AD_1_988.DASM:000009EF B9A0010000    mov ecx,0x1a0
CA9D8617-D652-403B-B6C5-BA47570116AD_1_988.DASM:00000C41 B90E1A0000    mov ecx,0x1a0e
CA9D8617-D652-403B-B6C5-BA47570116AD_1_988.DASM:00000D02 B9A0010000    mov ecx,0x1a0
CA9D8617-D652-403B-B6C5-BA47570116AD_1_988.DASM:00000EA5 B90E1A0000    mov ecx,0x1a0e
CA9D8617-D652-403B-B6C5-BA47570116AD_1_988.DASM:0000111D B9A0010000    mov ecx,0x1a0
F122A15C-C10B-4D54-8F48-60F4F06DD1AD_3_733.DASM:00000E79 B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:00000FDC B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:00000FE6 B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:00001EBE B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:00001EC8 B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:0000201B B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:00002025 B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:00002099 B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:000020A3 B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:00002100 B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:0000210A B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:0000212D B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:00002156 B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:00002631 B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:0000263B B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:000028C5 B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:000028CF B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:000028E1 B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:000028EB B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:00002B65 B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:00002B6F B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:00002C7E B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:00002CDA B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:0000307E B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:00003088 B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:000031B6 B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:000031C0 B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:000031FC B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:00003206 B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:000032A4 B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:000032AE B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:000032D1 B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:000032FA B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:00003CA9 B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:00003CB3 B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:00003CC5 B9A0010000    mov ecx,0x1a0
F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:00003CCF B9A0010000    mov ecx,0x1a0
FF917E22-A228-448D-BDAA-68EFCDDA5D3_2_771.DASM:00000F54 C7C1A0010000    mov ecx,0x1a0
FF917E22-A228-448D-BDAA-68EFCDDA5D3_2_771.DASM:000012C6 C7C1A0010000    mov ecx,0x1a0

```

4) Process/go through all of the modules above for any of these patterns:-

```
0000XXXX B9A0010000    mov ecx,0x1a0
0000XXXX 0F32          rdmsr
0000XXXX 0D00001000    or eax,0x100000
0000XXXX 0F30          wrmsr
```

- 0000XXXX – location/offset
- 0x100000 = 100000h = 0000 0000 0001 0000 0000 0000 0000b = bit 20
- Logical or opcode: or eax,0x100000 – basically set bit 20 with 1 (EIST locked)

OR

```
0000XXXX B9A0010000    mov ecx,0x1a0
0000XXXX 480FBAE814    bts rax,0x14
0000XXXX 488BD0        mov rdx,rax
0000XXXX E8A76D0000    call dword 0x8448
```

- 0000XXXX – location/offset
- 0x14 = 14h = 20 (in decimal) = bit 20
- Opcode **bts** – means **bit test and set**
- Opcode: bts rax,0x14 – basically set bit 20 with 1 (EIST locked)

Based on the patterns above, found two entries in **F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.DASM:-**

(First “**bit 20 lock**”)

```
000028EB B9A0010000    mov ecx,0x1a0
000028F0 480FBAE814    bts rax,0x14
000028F5 488BD0        mov rdx,rax
000028F8 E855360000    call dword 0x5f52
```

(Second “**bit 20 lock**”)

```
00003CCF B9A0010000    mov ecx,0x1a0
00003CD4 480FBAE814    bts rax,0x14
00003CD9 488BD0        mov rdx,rax
00003CDC E871220000    call dword 0x5f52
```

Modified to:-

1) (Please refer to **Page 5** for further explanation)

```
000028EB B9A0010000    mov ecx,0x1a0
000028F0 480FBAF014    btr rax,0x14
000028F5 488BD0        mov rdx,rax
000028F8 E855360000    call dword 0x5f52

00003CCF B9A0010000    mov ecx,0x1a0
00003CD4 480FBAF014    btr rax,0x14
00003CD9 488BD0        mov rdx,rax
00003CDC E871220000    call dword 0x5f52
```

(Explanation on how to modify the bit 20 lock for this pattern is on the next page)

Let me explain how to modify the First “**bit 20 lock**”:-

- Replaced opcode **bts** (bit test & set) with **btr** (bit test & reset)

- Change this:-

```
000028F0 480FBAE814      bts_rax,0x14
```

To this:-

```
000028F0 480FBAF014      btr_rax,0x14
```

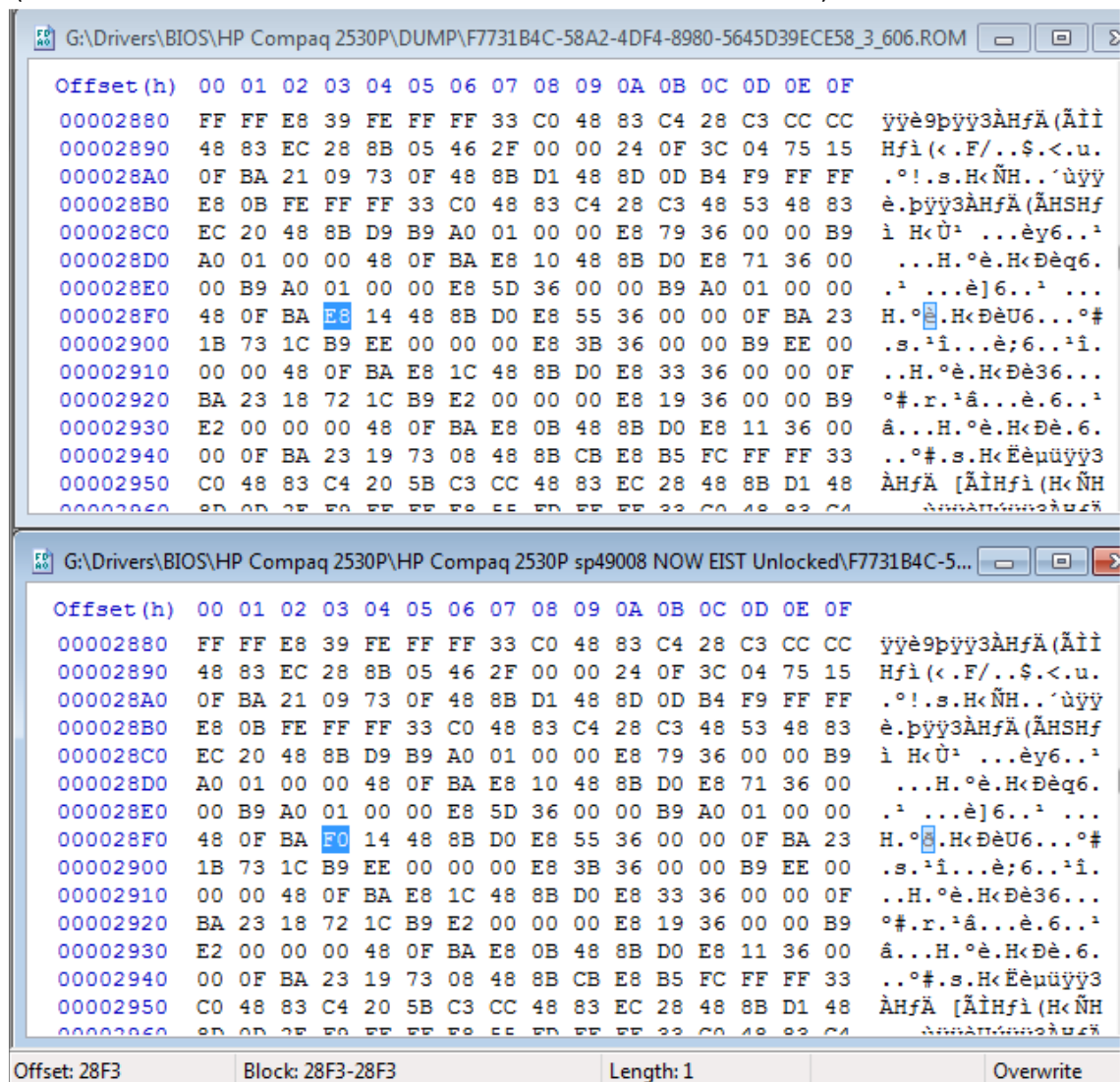
- Remember, the file we will need to edit/modify is the one with the **.ROM** extensions

- For the First “**bit 20 lock**”, we will need to change the hexadecimal value at offset

000028F3 with **0xF0**. Locate the offsets in **F7731B4C-58A2-4DF4-8980-**

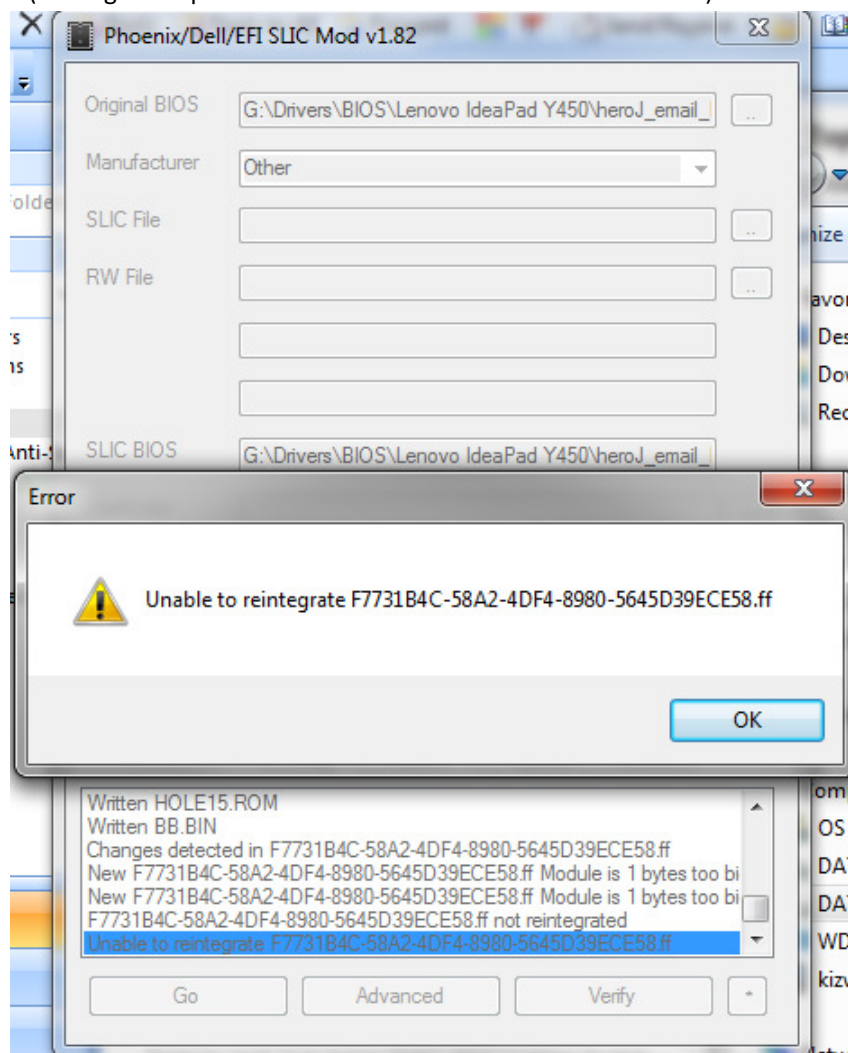
5645D39ECE58_3_606.ROM file, like this:-

(Before & after modification of the hexadecimal value at offset **000028F3**)



- Repeat the same procedure on the Second “**bit 20 lock**”. Change the hexadecimal values at offset **00003CD7** with **0xF0**.

- 2) In case you got “Module is X bytes too big” message during reintegration of the modified module (reintegration process will be covered later in this document) like this:-



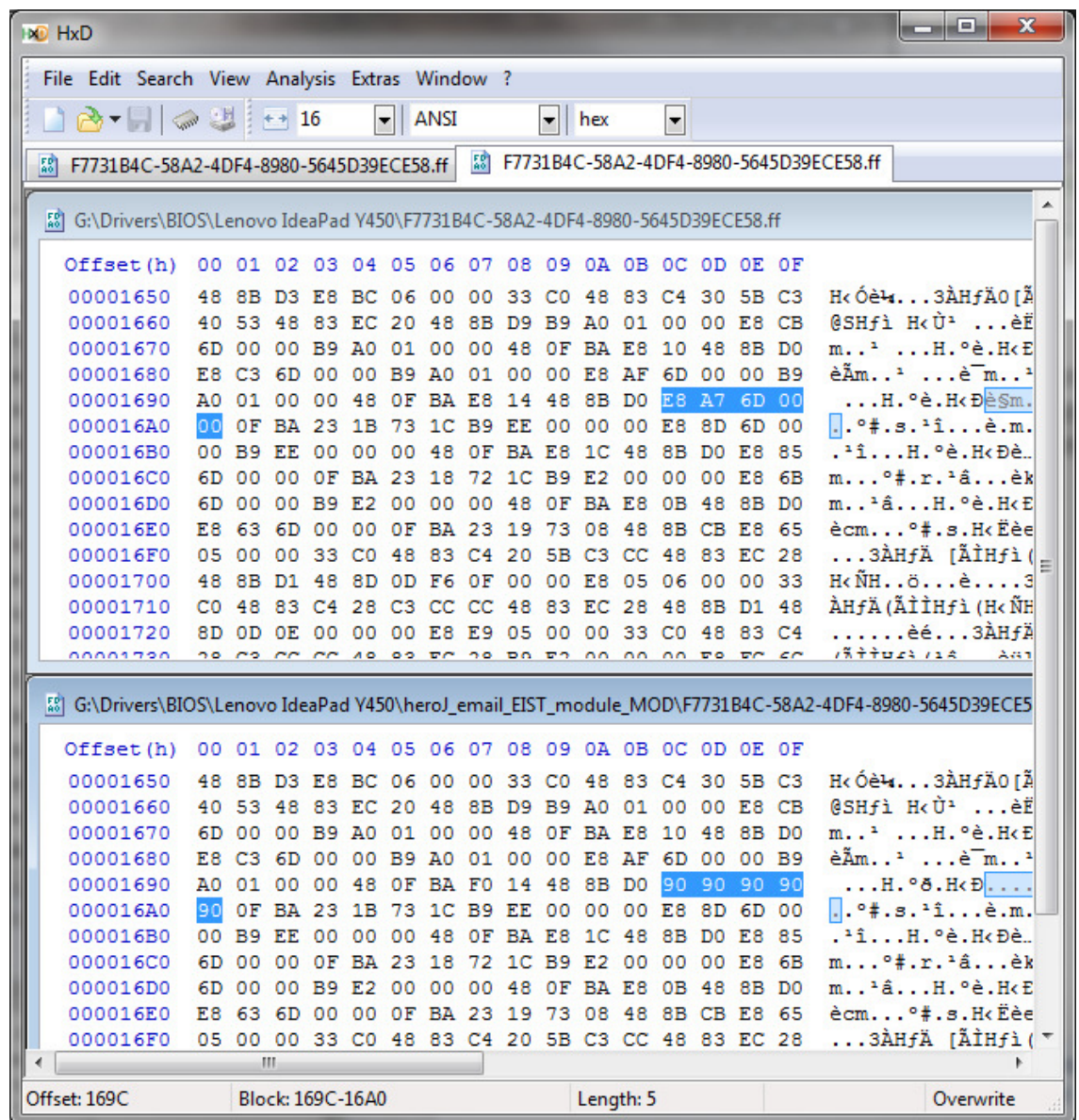
- We'll need to optimize the x86 assembler code a little bit. Let us look at one example from **Lenovo IdeaPad Y450's** BIOS:-

```
0000168F B9A0010000    mov ecx,0x1a0
00001694 480FBAF014    btr rax,0x14
00001699 488BD0        mov rdx,rax
0000169C E8A76D0000    call dword 0x8448
```

- The idea is to **nop**-ed (0x90) some code to allow successful reintegration. Since the idea of this patch is to prevent the BIOS to set the EIST Lock bit, the “call” operation is no longer needed. To explained this in an easy way to understand; the original BIOS has codes to set EIST Lock bit to 1 & to prevent it, we just removed it. In this pattern, the “call” operation is no longer needed. So, we just **nop**-ed it. Like so:-

```
0000168F B9A0010000    mov ecx,0x1a0
00001694 480FBAF014    btr rax,0x14
00001699 488BD0        mov rdx,rax
0000169C 90            nop
0000169D 90            nop
0000169E 90            nop
0000169F 90            nop
000016A0 90            nop
```


- What we need to do is to change the hexadecimal values from offsets **0x169C** to **0x16A0** with **0x90 (nop)**.



- Repeat the same procedure for the remaining lock.
- There is a possibility the pattern will be different than the one showed here but so far the pattern is the same. If you encountered such scenario, you'll need to examine the codes carefully to determine which codes can be **nop**-ed. OR you can post it in the forum & we'll examine it together.

Additional note:-

- In case you found this pattern:-

```
0000XXXX B9A0010000    mov ecx,0x1a0
0000XXXX 0F32          rdmsr
0000XXXX 0D00001000    or eax,0x100000
0000XXXX 0F30          wrmsr
```

Modified to:-

- Change 0x100000 to 0x0

```
0000XXXX B9A0010000    mov ecx,0x1a0
0000XXXX 0F32          rdmsr
0000XXXX 0D00000000    or eax,0x0
0000XXXX 0F30          wrmsr
```

Take this **bit 20 lock** as an example:-

```
000040EB B9A0010000    mov ecx,0x1a0
000040F0 0F32          rdmsr
000040F2 0D00001000    or eax,0x100000
000040F7 0F30          wrmsr
```

Modified to:-

```
000040EB B9A0010000    mov ecx,0x1a0
000040F0 0F32          rdmsr
000040F2 0D00000000    or eax,0x0
000040F7 0F30          wrmsr
```

(Explanation on how to modify the bit 20 lock for this pattern is on the next page)

Let me explain how to modify the **bit 20 lock** for this pattern:-

- The offsets which we will make changes are from offset **000040F2** to offset **000040F6**
- The original hexadecimal values are **0D 00 00 10 00** which the last 4 bytes:-
0x00100000 = 0000 0000 0001 0000 0000 0000 0000 0000 (in binary)
0000 0000 0001 0000 0000 0000 0000 0000 (in binary) = **bit 20**
- To prevent it from setting **bit 20** value to **1** which means locked, we will need to change **0x00100000** to **0x00000000**, like this:-
 (Before & after modification of the hexadecimal values from offset **000040F2** to offset **000040F6**)

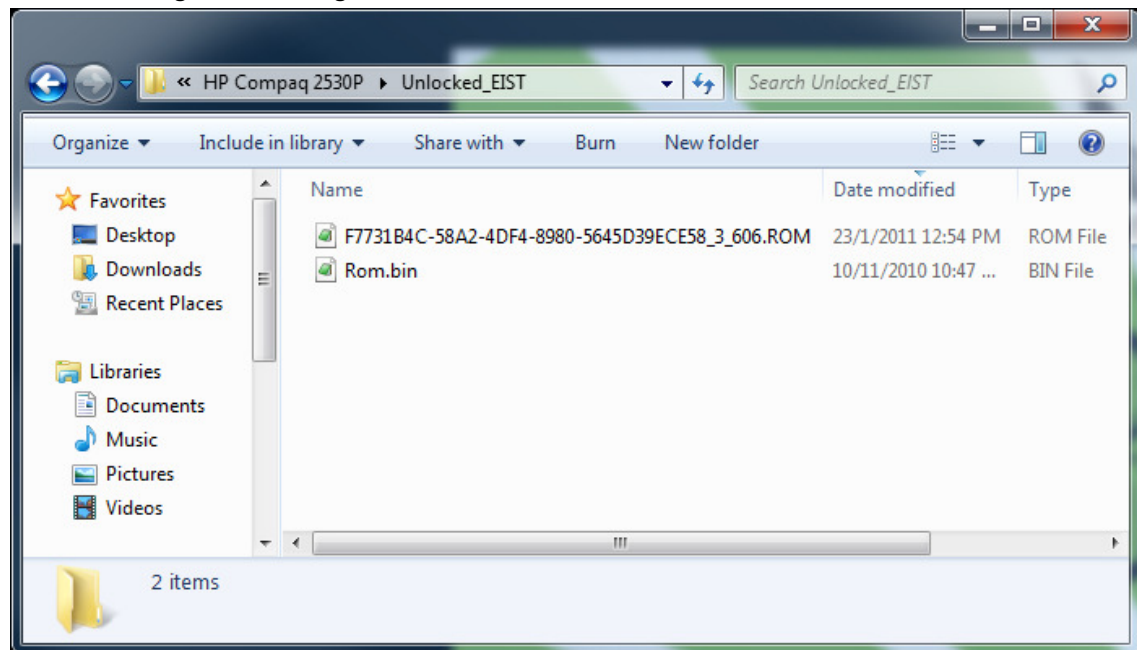
| | | | | | | | | | | | | | | | | | |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|
| Offset(h) | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | |
| 00004090 | 00 | 0F | 30 | 5A | 59 | 58 | C3 | E8 | EE | EE | FF | FF | C3 | 50 | 53 | 51 | ..0ZYXÃëiÿÿÃPSQ |
| 000040A0 | 52 | E8 | 9F | 00 | 00 | 00 | E8 | 63 | EF | FF | FF | 72 | 30 | E8 | C7 | EF | Rèÿ...èciÿÿr0èÇi |
| 000040B0 | FF | FF | 72 | 29 | F7 | 05 | 6B | 28 | 00 | 00 | 00 | 00 | 00 | 01 | 74 | 15 | ÿÿr)÷.k(.....t. |
| 000040C0 | B9 | 4E | 50 | 53 | 53 | E8 | F4 | EE | FF | FF | 83 | 0D | 8E | 14 | 00 | 00 | ³NPSSèôÿÿf.Ž... |
| 000040D0 | 40 | 33 | DB | EB | 03 | 33 | DB | 43 | E8 | 49 | EE | FF | FF | B9 | A0 | 01 | @3Ûë.3ÛCèIÿÿ¹ . |
| 000040E0 | 00 | 00 | 0F | 32 | 0D | 00 | 00 | 01 | 00 | 0F | 30 | B9 | A0 | 01 | 00 | 00 | ...2.....0¹ ... |
| 000040F0 | 0F | 32 | 0D | 00 | 00 | 10 | 00 | 0F | 30 | F7 | 05 | 6B | 28 | 00 | 00 | 00 | .2.....0÷.k(... |
| 00004100 | 00 | 00 | 01 | 75 | 05 | E8 | 26 | 00 | 00 | 00 | 5A | 59 | 5B | 58 | C3 | 50 | ...u.è&...ZY[XÃP |
| 00004110 | 51 | 52 | F7 | 05 | 6B | 28 | 00 | 00 | 00 | 00 | 00 | 01 | 74 | 0E | B9 | E2 | QR÷.k(.....t.²â |
| 00004120 | 00 | 00 | 00 | 0F | 32 | 25 | FF | F7 | FF | FF | 0F | 30 | 5A | 59 | 58 | C3 |2ÿ÷ÿÿ.0ZYXÃ |
| 00004130 | 50 | 51 | 52 | B9 | E2 | 00 | 00 | 00 | 0F | 32 | 0D | 00 | 08 | 00 | 00 | 0F | PQR²â....2..... |
| 00004140 | 30 | 5A | 59 | 58 | C3 | 50 | 51 | 52 | 56 | 83 | 3D | 9A | 14 | 00 | 00 | 00 | 0ZYXÃPQRVf=š.... |
| 00004150 | 0F | 85 | CA | 00 | 00 | 00 | B9 | 9D | 01 | 00 | 00 | 0F | 32 | 25 | 3F | 1F | ...Ê...².....2ÿ?. |
| 00004160 | 00 | 00 | A2 | B2 | 28 | 00 | 00 | 88 | 25 | AC | 28 | 00 | 00 | B9 | 17 | 00 | ..c²(..²ÿ¬(..².. |

| | | | | | | | | | | | | | | | | | |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|
| Offset(h) | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | |
| 00004090 | 00 | 0F | 30 | 5A | 59 | 58 | C3 | E8 | EE | EE | FF | FF | C3 | 50 | 53 | 51 | ..0ZYXÃëiÿÿÃPSQ |
| 000040A0 | 52 | E8 | 9F | 00 | 00 | 00 | E8 | 63 | EF | FF | FF | 72 | 30 | E8 | C7 | EF | Rèÿ...èciÿÿr0èÇi |
| 000040B0 | FF | FF | 72 | 29 | F7 | 05 | 6B | 28 | 00 | 00 | 00 | 00 | 00 | 01 | 74 | 15 | ÿÿr)÷.k(.....t. |
| 000040C0 | B9 | 4E | 50 | 53 | 53 | E8 | F4 | EE | FF | FF | 83 | 0D | 8E | 14 | 00 | 00 | ³NPSSèôÿÿf.Ž... |
| 000040D0 | 40 | 33 | DB | EB | 03 | 33 | DB | 43 | E8 | 49 | EE | FF | FF | B9 | A0 | 01 | @3Ûë.3ÛCèIÿÿ¹ . |
| 000040E0 | 00 | 00 | 0F | 32 | 0D | 00 | 00 | 01 | 00 | 0F | 30 | B9 | A0 | 01 | 00 | 00 | ...2.....0¹ ... |
| 000040F0 | 0F | 32 | 0D | 00 | 00 | 00 | 00 | 0F | 30 | F7 | 05 | 6B | 28 | 00 | 00 | 00 | .2.....0÷.k(... |
| 00004100 | 00 | 00 | 01 | 75 | 05 | E8 | 26 | 00 | 00 | 00 | 5A | 59 | 5B | 58 | C3 | 50 | ...u.è&...ZY[XÃP |
| 00004110 | 51 | 52 | F7 | 05 | 6B | 28 | 00 | 00 | 00 | 00 | 00 | 01 | 74 | 0E | B9 | E2 | QR÷.k(.....t.²â |
| 00004120 | 00 | 00 | 00 | 0F | 32 | 25 | FF | F7 | FF | FF | 0F | 30 | 5A | 59 | 58 | C3 |2ÿ÷ÿÿ.0ZYXÃ |
| 00004130 | 50 | 51 | 52 | B9 | E2 | 00 | 00 | 00 | 0F | 32 | 0D | 00 | 08 | 00 | 00 | 0F | PQR²â....2..... |
| 00004140 | 30 | 5A | 59 | 58 | C3 | 50 | 51 | 52 | 56 | 83 | 3D | 9A | 14 | 00 | 00 | 00 | 0ZYXÃPQRVf=š.... |
| 00004150 | 0F | 85 | CA | 00 | 00 | 00 | B9 | 9D | 01 | 00 | 00 | 0F | 32 | 25 | 3F | 1F | ...Ê...².....2ÿ?. |
| 00004160 | 00 | 00 | A2 | B2 | 28 | 00 | 00 | 88 | 25 | AC | 28 | 00 | 00 | B9 | 17 | 00 | ..c²(..²ÿ¬(..².. |

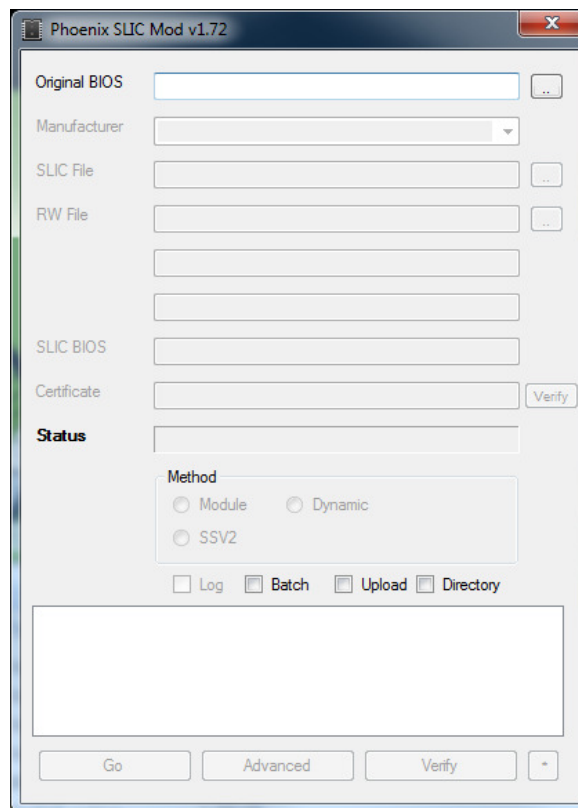
| | | | |
|--------------|------------------|-----------|-----------|
| Offset: 40F2 | Block: 40F2-40F6 | Length: 5 | Overwrite |
|--------------|------------------|-----------|-----------|

How to re-build BIOS image with Phoenix SLIC Mod tool

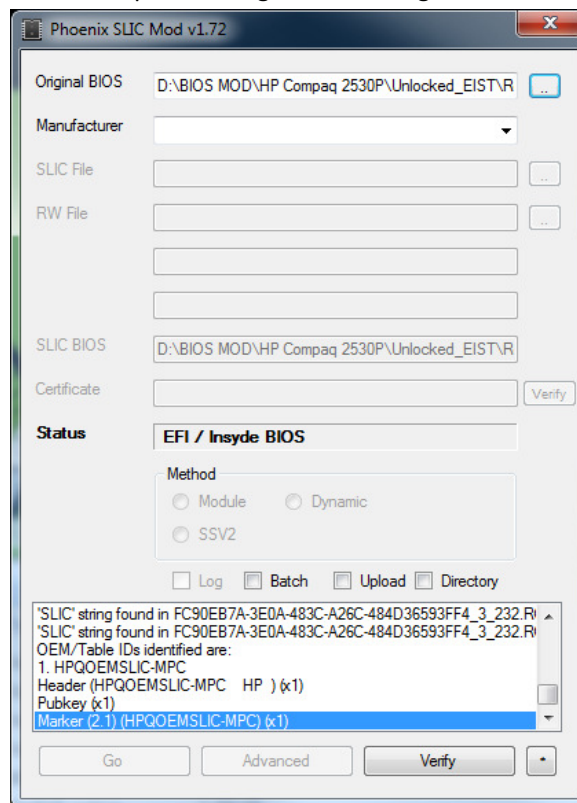
- Put the original BIOS image & the modified module in the same folder:-



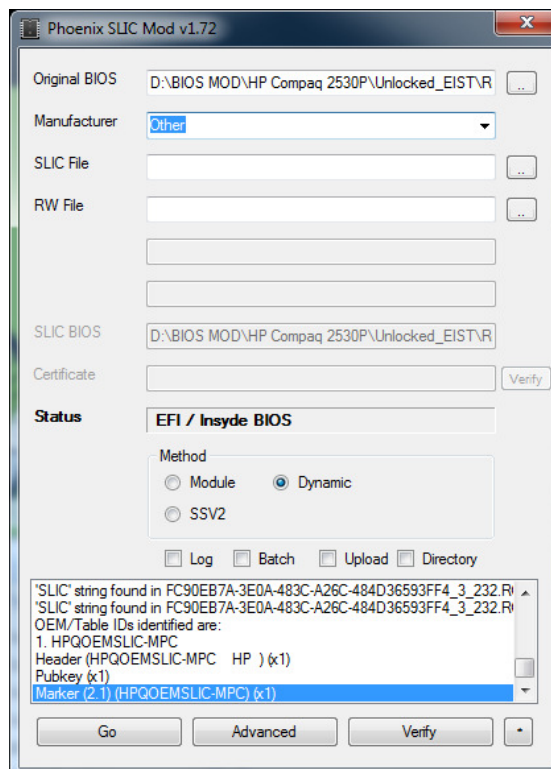
- Launch **Phoenix SLIC Mod** tool:-



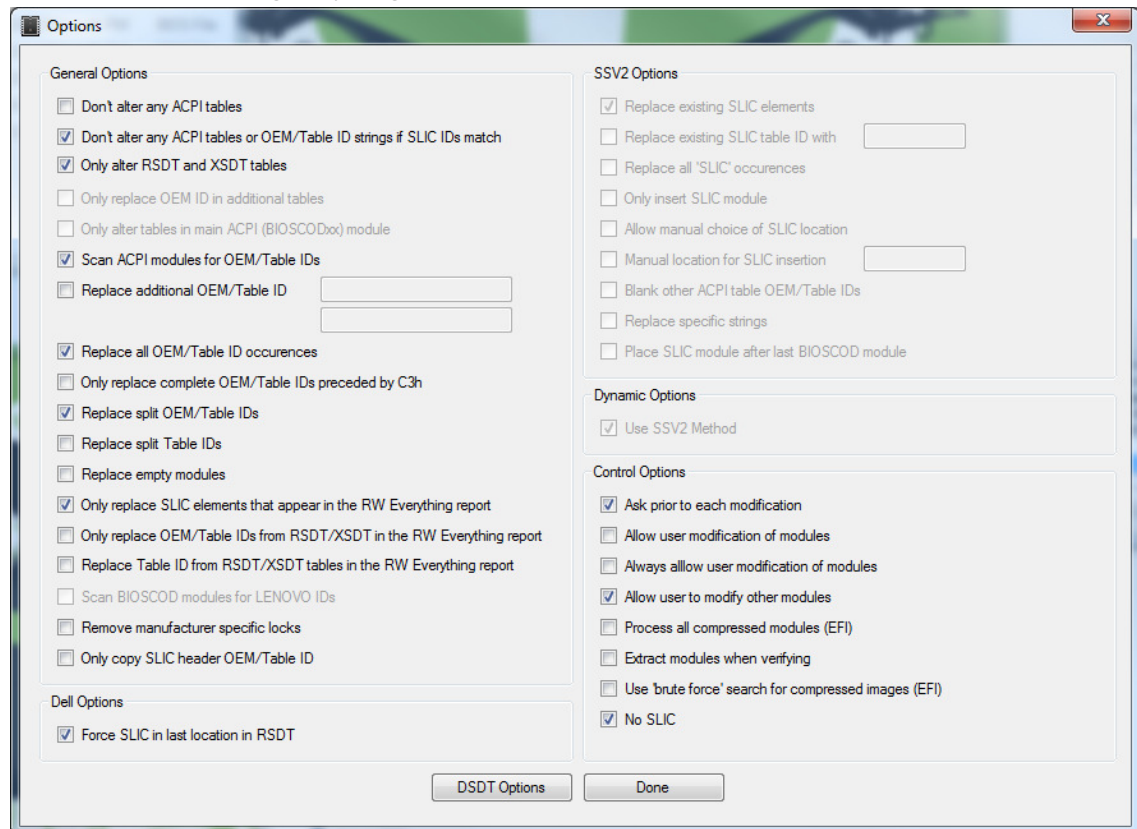
- At the **Original BIOS** field, open the original BIOS image, in this example **Rom.bin**:-



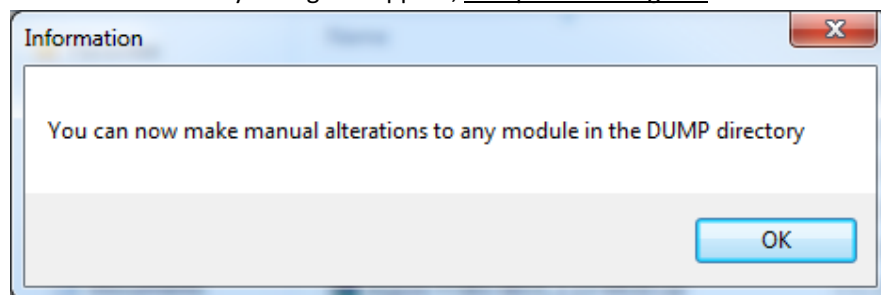
- At the **Manufacturer** field, select any manufacturer – this basically just for enabling the **Advanced** button:-



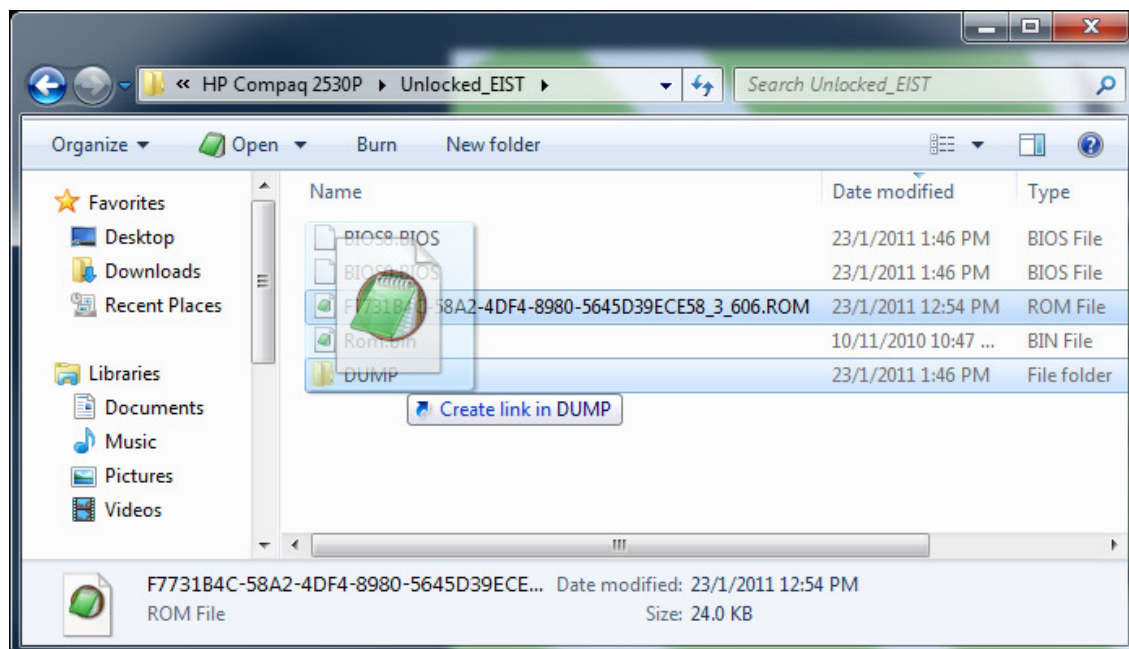
- Open **Advanced** options window & select these three options in the **Control Options** section (don't change anything else):-



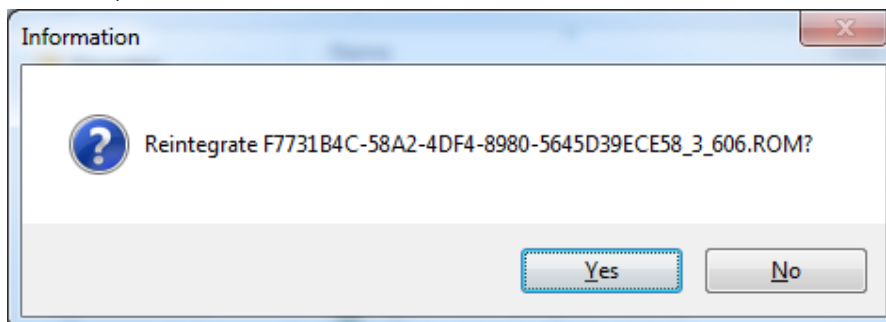
- Click **Done** & click **Go** button:-
Click **Yes** or **OK** if any dialog box appear, except this dialog box:-



- When you see the above dialog box, don't click OK button yet, copy the modified module, in this example is **F7731B4C-58A2-4DF4-8980-5645D39ECE58_3_606.ROM**, to inside **DUMP** folder:-
In the screenshot it says "Create link" which actually copy it to inside **DUMP** folder.
Replace the original module in **DUMP** folder with the modified module.

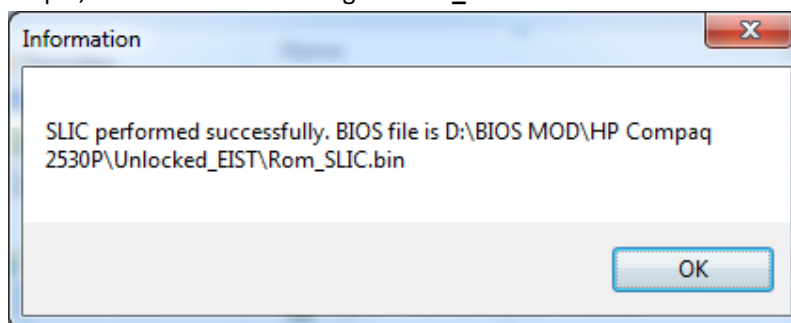


- After you have replaced the module, click **OK**:-
It will notice the changes & ask you whether you want to reintegrate the modified module or not, click **Yes**:-



If it asks to reintegrate other module than the modified module we want to reintegrate, just click **No**.

- When the modified BIOS image is successfully re-build, you'll see this dialog box:-
In this example, the modified BIOS image is **Rom_SLIC.bin**



DISCLAIMER

- **Modifying/patching BIOS code is very dangerous. If done incorrectly, the computer can bricked (nonoperational). Therefore, use the information available in this document at your own risk. I will not held responsible for any damage.**

ACKNOWLEDGEMENT

- [Uncle Joe](#) from **thinkpads.com** for the brilliant information on how to locate EIST Lock bit.
- [nando4](#) from **notebookreview.com** for helping me when I investigate the EIST Lock bit.
- [andyp](#) from **mydigitallife.info** for the great **Phoenix Tool**, also for providing the fix for module reintegration problem.
- [middleton](#) from **notebookreview.com** for the brilliant information regarding the assembly code.