Introduction

A Windows system's DSDT table root bridge definition (ACPI PNP0A08 or PNP0A03) is usually confined to a reserved 32-bit space (under 4GB) budgetted to be large enough to host the notebook's PCIe devices. A watermark TOLUD value is then set and locked in the system firmware. Windows OS honors the root bridge definition and will allocate PCIe devices within it. macOS ignores the root bridge constraints as too does Linux when booted with the 'pci=noCRS' parameter. Neither of those OS require a DSDT override and can allocate freely in the huge 64-bit PCIe address space.

When retrofitting a eGPU, an **error 12 (This device cannot find enough free resources that it can use)** can occur against an eGPU in Windows' device manager making it inoperable. This can indicate there is insufficient 32-bit addressing space available to host the eGPU. An eGPU requires a relatively large PCIe config space to allocate into. Decreasing TOLUD by reducing RAM to 2GB offers a somewhat impractical workaround. Rather, the definitive solution is below.

This three step solution removes Window's 32-bit PCIe allocation constraint in order to resolve the eGPU error 12:

Step 1. Create a dsdt-modified.aml DSDT file with a 36-bit root bridge

Step 2. Load your dsdt-modified.aml as registry override or in-memory substitution

Step 3. Confirm success with a 'Large Memory' area in Device Manager

macOS users: refer instead to Mikeal's post that covers these steps titled *Windows 10 - Clover DSDT memory override* [UEFI Windows on Macbooks only].

Step 1. Create a dsdt-modified.aml DSDT file with a 36-bit root bridge

i. Download and install required tools:

- Windows Binary Tools (WBT - Dec 2016) extracted to **c:\dsdt** directory. [newer WBT has parsing errors]
- Windows Driver Kit (WDK), which contains the Windows ASL Compiler (asl.exe)
- Notepad++ text editor with Search->Goto (line) menu for fast line editting if asl or iasl compilation fails below

ii. Copy WDK's ASL compiler into the c:\dsdt directory. Do this by opening **Command Prompt** (run as administrator) and then **copy-and-paste** the commands below.

```
mkdir c:\dsdt
c: & cd \dsdt
set 64bit_OS_asl="C:\Program Files (x86)\Windows Kits\10\Tools\x64\ACPIVerify\asl.exe"
set 32bit_OS_asl="C:\Program Files (x86)\Windows Kits\10\Tools\x86\ACPIVerify\asl.exe"
copy /y %32bit_OS_asl% c:\dsdt > nul & copy /y %64bit_OS_asl% c:\dsdt > nul

if not exist c:\dsdt\asl.exe echo ERROR: Failed to copy asl.exe to c:\dsdt
```

iii. Dump your ACPI tables to disk files (dsdt.asl and dsdt.dat) with these commands at **Command Prompt** (run as administrator). The created dsdt.asl is copied here as dsdt-modified.asl which is used later on to make our required modifications.

```
c: & cd \dsdt
acpidump -b -z
asl /u dsdt.dat
copy dsdt.asl dsdt-modified.asl
```

iv. You now can **choose either the Intel method** or **Microsoft method** (with blue dsdt-modified.dsl or magenta dsdt-modified.asl work file respectively) to generate a dsdt-modified.aml file, even trying both to maximize success. Consider:

- For systems other than Lenovo, use the Intel method as their DSDT usually has an Intel creation signature.

- Lenovo Thinkpad X220, T420, W530, T540P and likely other 2nd-4th gen i-core Lenovo Thinkpad systems are known to require the Microsoft method . The Intel method causes a "ACPI BIOS ERROR" on Windows bootup there.

OPTION 1: Use the Intel method

i. **Save this >> refs.txt file << to your c:\dsdt folder.** What is it used for? From tonymacx86: *The iasl disassembler will attempt to guess the number of arguments [for unresolved symbols not defined in any file] but often guesses poorly. You can correct it by providing the external declarations in a refs.txt text file. It contains some common (and not so common) missing symbols . . .*

ii. From the Command Prompt (admin), **decompile** dsdt.dat as dsdt.dsl . The refs.txt file is used here. **dsdt.dsl** is then copied as dsdt-modified.dsl on which we'll make the required changes on.

```
c: & cd \dsdt
iasl -da -dl -fe refs.txt dsdt.dat

copy dsdt.dsl dsdt-modified.dsl
```

iii. With Notepad++, open the resultant c:\dsdt\dsdt-modified.dsl file and search for **ResourceProducer**. Beneath it will be a series of "DWordMemory" resource entries. Under the **last DWordMemory entry** in that area, typically **above the _CRS method, add a 'QWordMemory' (64-bit)** entry as shown **below**. The range chosen is in the 36-bit range (< 64GB) to maintain compatibility with PAE-capable 32-bit Windows. A location above 48GB was chosen to alleviate issues with 32GB equipped systems. Here we use between 48.5GB to 56.25GB. Once systems start shipping with 64GB of RAM, this will need to be revised to use a 64-bit address.

```
DWordMemory (ResourceProducer, PosDecode, MinFixed, MaxFixed, Cacheable, ReadWrite,
    0x00000000,          // Granularity
    0x000A0000,          // Range Minimum
    0x000BFFFF,          // Range Maximum
    0x00000000,          // Translation Offset
    0x00020000,          // Length
    ,, , AddressRangeMemory, TypeStatic)
// - ADD THIS SECTION
-------------------------------------------------------------------------
QWordMemory (ResourceProducer, PosDecode, MinFixed, MaxFixed, Cacheable, ReadWrite,
    0x0000000000000000, // Granularity
    0x0000000C20000000, // Range Minimum,  set it to 48.5GB
    0x0000000E0FFFFFFF, // Range Maximum,  set it to 56.25GB
    0x0000000000000000, // Translation Offset
    0x00000001F0000000, // Length calculated by Range Max - Range Min.
    ,, , AddressRangeMemory, TypeStatic)
//
-----------------------------------------------------------------------------------------
})
    Method (_CRS, 0, Serialized)  // _CRS: Current Resource Settings
```

iv. **Create a** dsdt-modified.aml **file** . The '-ve' disables warning messages.

```
c: & cd \dsdt

iasl -ve dsdt-modified.dsl
```

It is unlikely this will succeed first time as the compiler is very strict. Errors reported will need to be looked at with Notepad++ on the line they occur on. Search for a unique error keyword from your dsdt-modified.dsl error line within dsdt-modified.asl (it may look a bit different) and the just swap the lines above/below into your dsdt-modified.dsl file and compile. This was sufficient to get a Dell XPS 9350, Dell E6540, 2016 15" Macbook Pro DSDT override all done perfectly.

If stuck then see eGPU.io's public pre-compiled DSDT repository for various systems. If one exists for your system then can download it along with Winmerge to merge the file changes into your dsdt-modified.dsl work file. There are also other fixes like described at [Guide] Patching LAPTOP DSDT/SSDTs (tonymacx86). Consider also asking for guidance at the DSDT-centric tonymacx86.com DSDT forums . Then try compiling your DSDT again.

v . Proceed to Step 2 to load your dsdt-modified.aml file.

OPTION 2: Use the Microsoft method

i. With Notepad++, find the working area in your c:\dsdt\dsdt-modified.asl file as shown in screenshots below. In that area, **make the following two edits**. These are annonated in those screenshots.

EDIT 1: use Windows' calc in programmers mode to **add hexidecimal 2E** (46 decimal) to your "Buffer(**value**)" line. Examples:

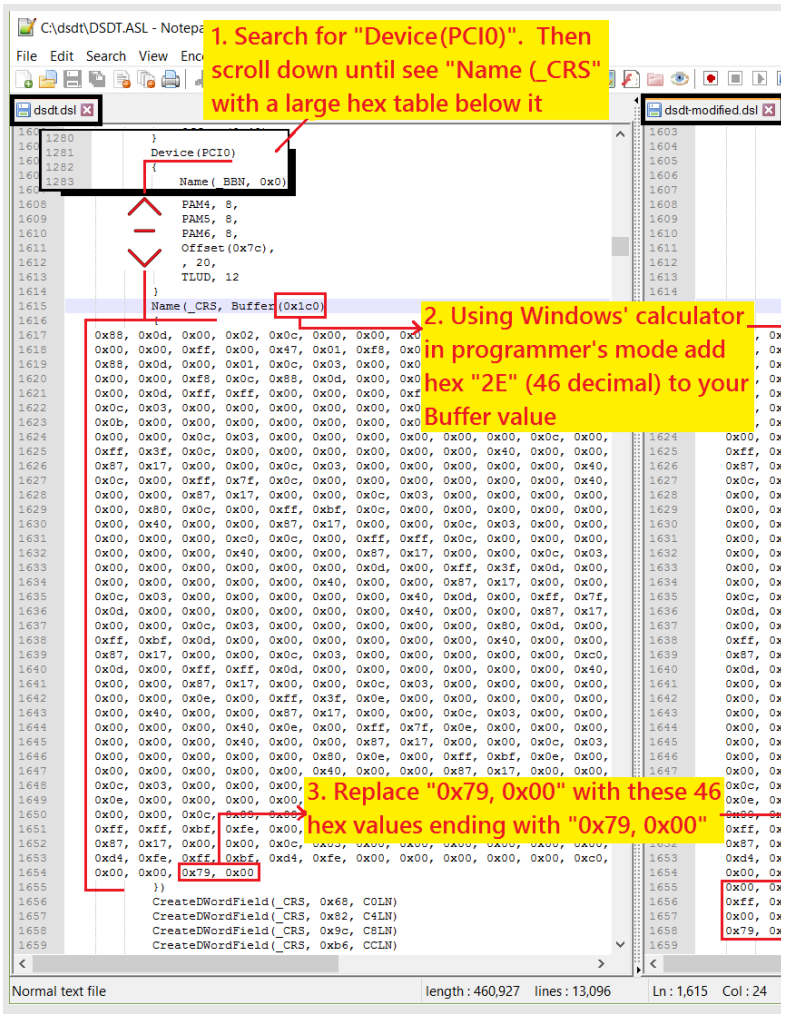
- "Buffer(0x1C0)" becomes "Buffer(0x1EE)"
- "Buffer(0x1D4)" becomes "Buffer(0x202)"

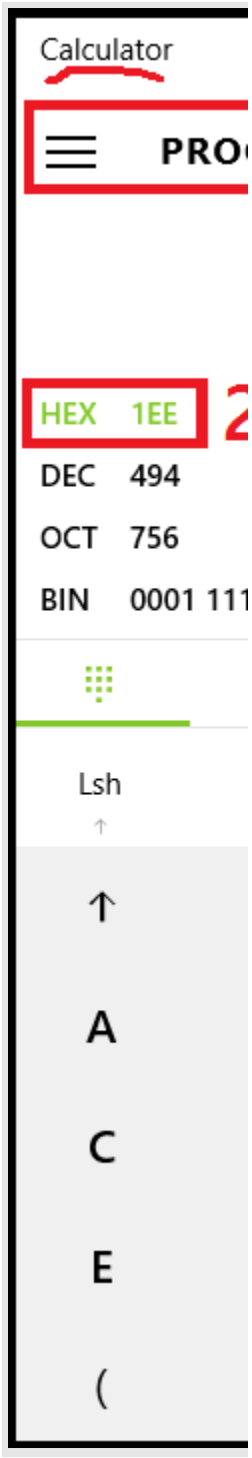
EDIT 2: replace "0x79, 0x00" in the hex table end with the following 46 hex values, also ending with "0x79, 0x00". This adds the same 36-bit **QWordMemory** resource entry shown in the Intel method but as a hex table.

[copy-and-paste this table]

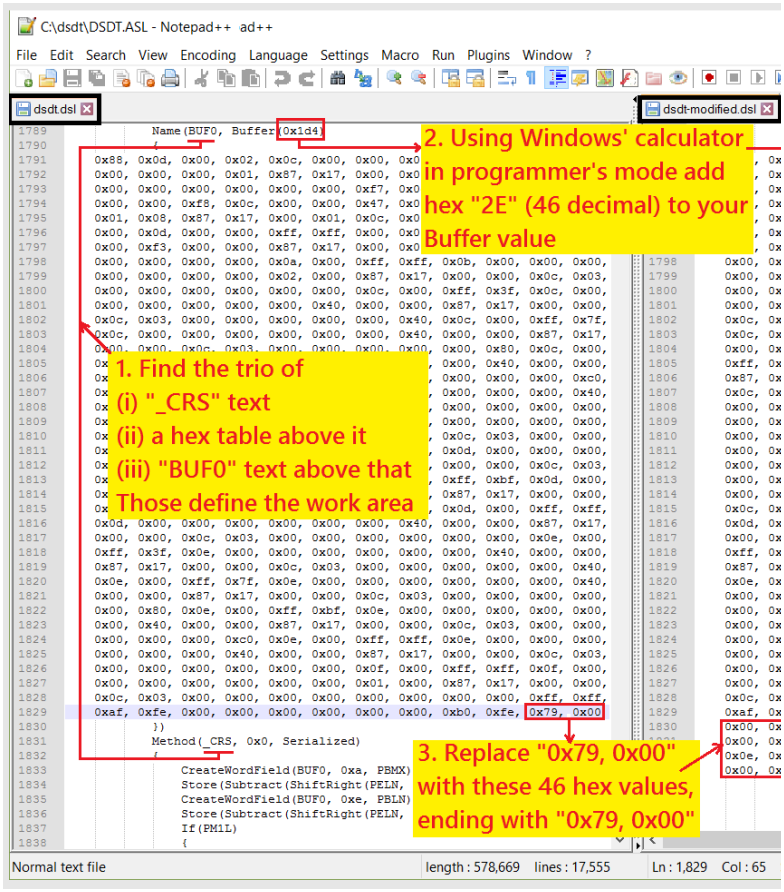
```
0x8a, 0x2b, 0x00, 0x00, 0x0c, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x0c, 0x00, 0x00, 0x00, 0xff, 0xff,
0xff, 0x0f, 0x0e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0xf0, 0x01, 0x00, 0x00, 0x00, 0x79, 0x00
```

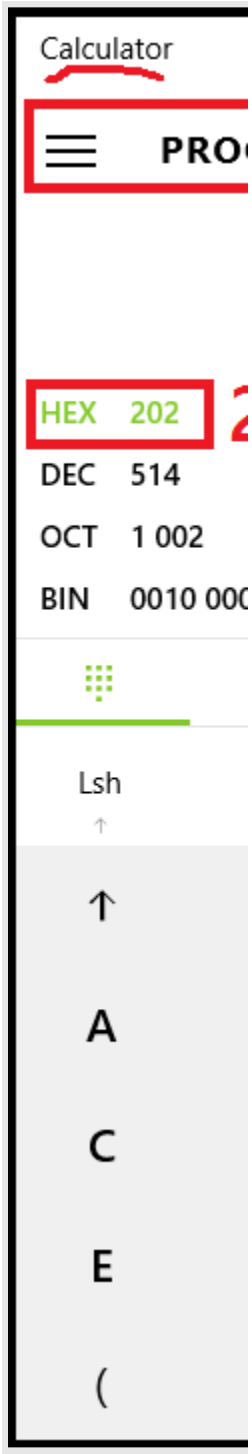
For Lenovo Systems [CLICK on images to zoom]





For other systems [CLICK on images to zoom]





ii. At the Command Prompt (admin), **compile your** dsdt-modified.asl **file to generate a dsdt-modified.aml file:**

```
c: & cd \dsdt
```

```
asl /Fo=dsdt-modified.aml dsdt-modified.asl
```

This will almost certainly fail to generate a dsdt-modified.aml on your first compile attempt due to to error lines requiring further editting of your dsdt-modified.asl file, like the ones below (further example here), followed by another compile attempt. A successful compilation with many warnings is OK.

- ATMC*() - error: ATMC is not a method* -> swap ATMC with _SB_.PCI0.LPC_.EC__.ATMC()

- *Zero - error: unexpected ASL term type* -> add 'Zero' as bracketted argument to line above it, eg: GLIS (Zero)

- *Arg0 - error: unexpected ASL term type* -> add 'Argo0' as bracketted argument to line above it, eg: GDCK (Arg0)

iii. Open c:\dsdt with Windows explorer. **Compare the size of your created dsdt-modified.aml file against the memory dumped dsdt.dat**. They should be within +/- 10% of each other in size. If not, repeat the above process to make sure no mistakes were made.

A HP Elitebook 8440P saw the generated dsdt-modified.aml being only 15% the size of the memory dumped dsdt.dat. There this process simply did not work. For such cases use the Intel method instead.

iv . Proceed to Step 2 to load your dsdt-modified.aml file.

Step 2. Load your dsdt-modified.aml as a registry override or in-memory substitution

OPTION 1: Load your dsdt-modified.aml as a registry override with Windows test signing mode enabled

Here we load your dsdt-modified.aml as Windows registry DSDT override. Do note that an invalid dsdt-modified.aml loaded in this way can cause a BSOD on bootup. Furthermore, Windows **test signing mode** can be problematic for app compatibility. Both these issues can be avoided by using OPTION 2: Avoid test signing mode - load your dsdt-modified.aml as an in memory DSDT substitution.

i. At Command Prompt (**admin**) type the following. WDK containing asl.exe must be installed per Step 1 (i) and (ii) for this to work.

```
c: & cd \dsdt
```

```
asl /loadtable dsdt-modified.aml
```

ii. Enable TESTSIGNING mode for the registry override to apply. At the **Command Prompt** (admin) type:

```
bcdedit -set TESTSIGNING ON
```

If get an error like below when do this, then disable SECURE BOOT in your BIOS.

An error has occurred setting the element data

The value is protected by Secure Boot policy and cannot be modified or deleted

iii. Reboot your system and check for 'large memory' in step 3.

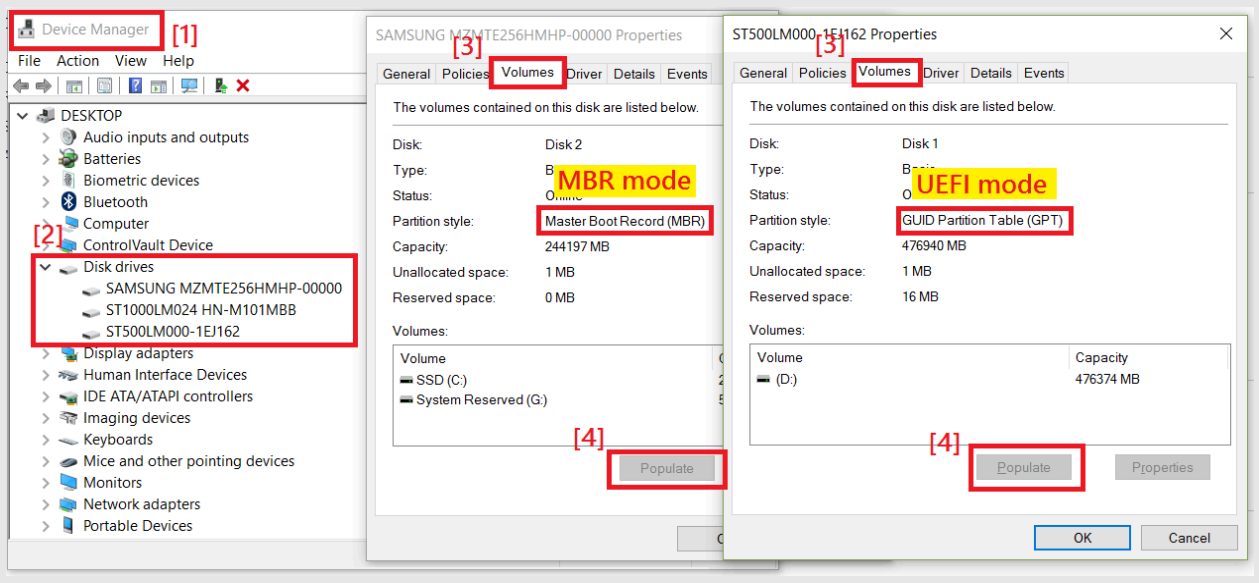
OPTION 2: Avoid test signing mode - load your dsdt-modified.aml as an in memory DSDT substitution

Here we avoid the problematic registry override & test signing mode altogether by loading the DSDT as an in-memory substitution before Windows loads with these steps:

i. Disable the previous registry DSDT override and test signing mode.

```
bcdedit -set TESTSIGNING OFF
```

ii. check whether you are using a **MBR or UEFI Windows installation** by viewing Device Manager->Disk Drives->[double-click boot drive]->Volumes->Populate->Partition Style. If it says "MBR", then it's a MBR install. If it says "GPT" then it's a UEFI install. This will determine which next step to apply.



iii. [**MBR mode**] Load the dsdt-modified.aml as an in-memory substitution via nando's DIY eGPU Setup 1.35 in the next post.

iv. [**UEFI mode**] Load the dsdt-modified.aml as an in-memory substitution via Clover bootloader as follows:

Update Feb-2019 >> Mac users are advised to use @goalque's automate-eGPU EFI instead of Clover to load your resultant DSDT override to avoid issue noted below in the *BIG WARNING*.

BIG WARNING by nando4 >> @Goalque has correctly identified that Clover loads a DSDT table in firmware volume and as such can brick a Macbook as this user found. If you proceed with using Clover to do a DSDT override the you do so at your own risk!! For risk-adverse users it is suggested to simply do a DSDT registry override and persevere with Windows' test signing mode until other solutions are found and presented.

- Mount your EFI volume as s: drive and backup your existing \EFI\BOOT\BOOTX64.efi file. At Command Prompt (admin) type:

```
mountvol s: /s
```

```
copy s:\EFI\BOOT\BOOTX64.efi s:\EFI\BOOT\BOOTX64.win
```

- Download the Clover ISO file and install it by extracting with 7-zip (64-bit only) the clover.tar.lzma->clover.tar->clover.pkg->\EFI folder to s:\EFI. Be sure to use the the 7-zip interface to extract to s: drive as Windows explorer refuses to allow viewing of the s: EFI volume.

- Copy your dsdt-modified.aml file as dsdt.aml to s:\EFI\CLOVER\ACPI\Windows, the directory Clover uses to preload it. If your source dsdt-modified.aml file is not in c:\dsdt then use this workaround to copy-and-paste it. Locate your dsdt-modified.aml file with Explorer, right-click, rename to dsdt.aml, right-click, copy. Then hit CTRL+ALT+DEL, task manager, File->New task. Explore the s:\EFI\CLOVER\ACPI\WINDOWS directory and paste it there.

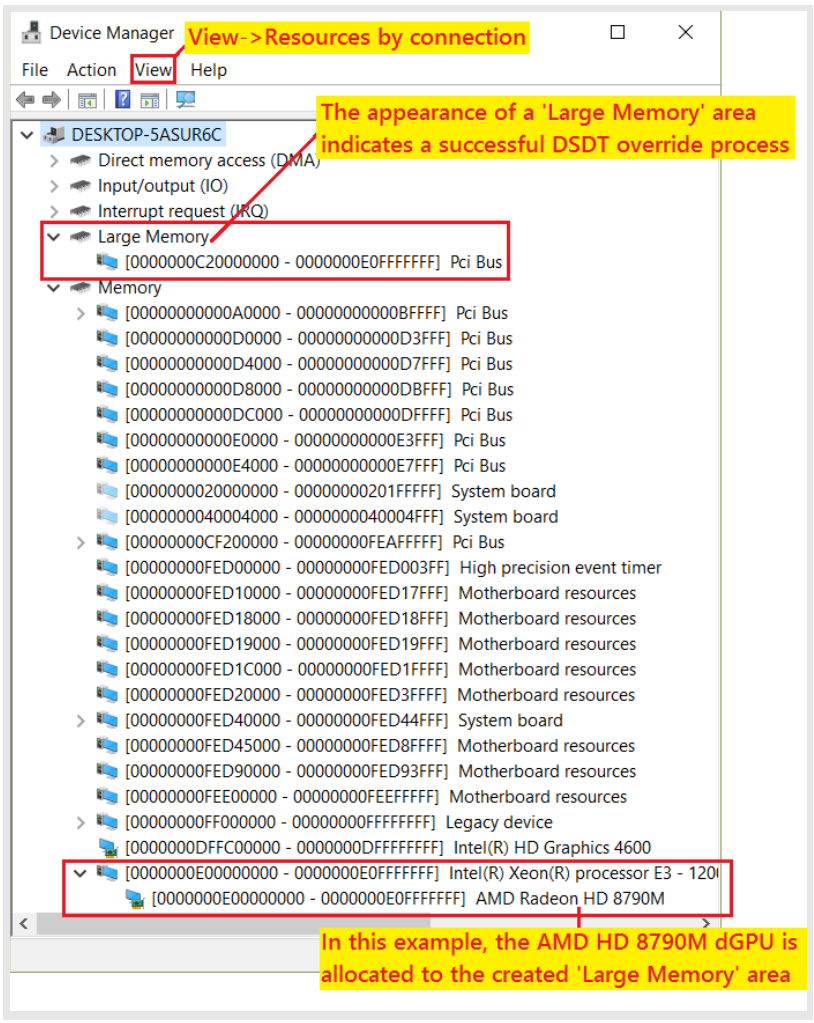
Otherwise at the Command Prompt (admin) type:

```
copy c:\dsdt\dsdt-modified.aml s:\EFI\CLOVER\ACPI\WINDOWS\dsdt.aml
```

- Reboot via Clover -> Windows EFI menu and proceed to step 3 to confirm it worked.

Step 3. Confirm success with a 'large memory' area in Device Manager

Check you now have a new **Large Memory** entry in Device Manager as shown below to confirm your dsdt-modified.aml was a success:



Success stories