[HowTo] Get full NVMe Support for all Systems with an AMI UEFI BIOS - Special Topics / NVMe Support for old Systems - Win-Raid Forum

@ all:
Please read this before posting into this thread:

This thread has been designed by me for users, who want to do the required BIOS modification *themselves* by following this guide.
Users, who want to get an already modded BIOS, may search for it within >this< 2.4k Sub-Forum or post their request into >this< 492 one.

The main topic of this thread is the procedure of the NVMe specific BIOS modification and the support in case of appearing problems.
Users, who don't know how to flash a modded BIOS or how to avoid a clean OS installation, should search for a guide/solution by using the Forum's "Search" box and post their request into the related specific thread.

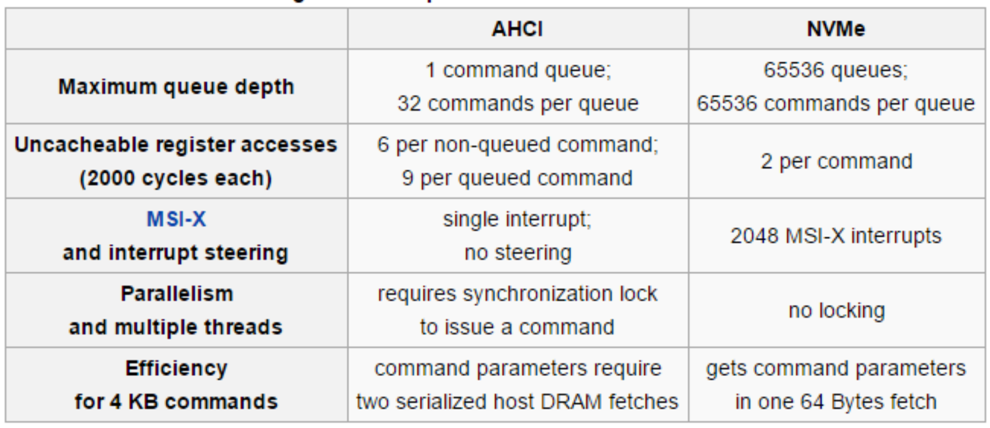
What is NVMe?

▾ **Wikipedia Info:** (open/hide by a click)
NVMe is a specification for accessing Solid-State Drives (SSDs), which are attached through the PCI Express (PCIe) bus. "NVM" stands as an acronym for "Non-Volatile Memory", which is used in SSDs.
While Advanced Host Controller Interface (AHCI) interface has the benefit of legacy software compatibility, it does not deliver optimal performance when an SSD is connected via PCI Express bus. This is because AHCI was developed back at the time when the purpose of a host bus adapter (HBA) in a system was to connect the CPU/memory subsystem with a much slower storage subsystem based on rotating magnetic media. Such an interface has some inherent inefficiencies when applied to SSD devices, which behave much more like DRAM than like spinning media.

NVMe has been designed from the ground up, capitalizing on the low latency and parallelism of PCI Express SSDs, and fulfilling the parallelism of contemporary CPUs, platforms and applications. At a high level, the basic advantages of NVMe over AHCI relate to its ability to exploit parallelism in host hardware and software, manifested by differences in depth of command queues, interrupts processing, the number of uncacheable register accesses etc., resulting in various performance improvements.

The table below summarizes high-level differences between the basic NVMe and AHCI device interfaces:

High-level comparison of AHCI and NVMe[3]



	AHCI	NVMe
Maximum queue depth	1 command queue; 32 commands per queue	65536 queues; 65536 commands per queue
Uncacheable register accesses (2000 cycles each)	6 per non-queued command; 9 per queued command	2 per command
MSI-X and interrupt steering	single interrupt; no steering	2048 MSI-X interrupts
Parallelism and multiple threads	requires synchronization lock to issue a command	no locking
Efficiency for 4 KB commands	command parameters require two serialized host DRAM fetches	gets command parameters in one 64 Bytes fetch

Who can benefit from an NVMe SSD?

▾ **Answer:** (open/hide by a click)
Everything seems to be prepared for a drastical SSD performance boost by the new data transfer standard named NVMe:
The latest Windows Operating Systems from Win8.1 up natively do support NVMe. For Win7 and Server 2008 R2 users I recommend to read the start post of >this< 240 thread.

Nearly all mainboard manufacturers are meanwhile offering new BIOS versions with full NVMe support for their mainboards with an Intel 9-Series or X99 Chipset.

Many consumer SSDs with NVMe support are already available or will be available soon.

There is no doubt, that users with an up-to-date mainboard, whose BIOS contains the required NVMe module(s), will benefit from the performance boost initiated by an NVMe supporting SSD.

What about MBs with an UEFI BIOS, but w/o NVMe Support?

▾ **Answer:** (open/hide by a click)

When I started this thread in April 2015, these were the most interesting questions regarding NVMe support:
Is it possible to give natively not supported systems the option to boot off an NVMe SSD by modding the BIOS?

If yes, which modules have to be inserted and how has it to be done?

After having done the related test with my old Intel Z68 system I found in June 2015 the first time answers to the questions. My success report can be found here:
https://winraid.level1techs.com/t/howto-get-full-nvme-support-for-all-systems-with-an-ami-uefi-bios/30901/117. 562
After this first success the method has been optimized and successfully tested with nearly all existing mainboard models.
Since February 2017 we know for sure, that the insertion of the natively missing NVMe module into the UEFI BIOS enables the ability to boot off an NVMe SSD for all natively not supported Intel and AMD Chipset systems.
You can find the well approved latest version of my Guide within the next chapter.

Guide about how to get full NVMe support

(valid for all Intel/AMD chipset systems with an AMI UEFI BIOS)

(last updated: 12/04/2022)
Changelog:
Guide structure optimized (as good as possible)

This is what you should know:

▾ **Details:** (open/hide by a click)
It is no problem to get a PCIe or M.2 connected NVMe SSD working with an old system without modifying the mainboard BIOS, if
a) the NVMe SSD will be used just for the **storage** of data (as drive D:, E: etc.)
and
b) an appropriate NVMe driver is present within the Operating System (either natively or loaded/integrated),

but the usage of such SSD as bootable system drive C: (incl. the boot sector) usually requires a special mainboard BIOS EFI module, which has to be loaded while booting.
Note: Some NVMe SSDs like Samsung's 950 Pro SSD are natively bootable in LEGACY mode (CSM and loading of Option ROMs has to be enabled within the BIOS), because their Controller chip contains its own NVMe supporting Option ROM module.
These are the best pre-conditions for a successful implementation and configuration of an NVMe SSD as bootable system drive:

The desired OS should be Win10 or Win11 (due to their native NVMe support).

The mainboard BIOS should offer the required UEFI boot settings.

It should be possible to get a modded BIOS successfully flashed into the system's BIOS chip.

The on-board SATA Controller should not been set to "RAID" mode within the BIOS.
Tip: Before you start with the preparations, I recommend to read >this< 1.4k report written by our Forum member Paulos7.

Alternative methods to use an NVMe SSD as bootable drive with older systems (no BIOS modding required):

"Clover-EFI Bootloader Method" (the related guide written by our Forum member Nyctophilia can be found >here< 1.5k)

"DUET-USB Boot Method" (the related guides written by our Forum members noInk resp. mireque can be found >here< 480, >here< 112 and >here< 95.

This is what you will need:

▾ **Details:** (open/hide by a click)
1. A modern unzip tool like WinRAR v6.xx or 7-Zip v2xxx
(has to support the RAR5 compression)
2. A mainboard with an AMI Aptio UEFI BIOS

3. The desired BIOS version for the related mainboard model
as extracted "pure" BIOS file (not an *.exe one), original or already pre-modified

4. An appropriate UEFI BIOS modding tool

These are the alternatives:
a) An Aptio IV compatible AMI UEFI MMTool

(easy to use and usually best choice for AMI Aptio IV BIOSes: v4.50.0.23)
Important: These tools are not free available.
Tip: Do a Google search for "MMTool Aptio 4.50.0023", join the listed TweakTown site, scroll down to the "AMI (UEFI) BIOS Tools" section and download the desired tool.

b) CodeRush's UEFITool
(very good tool for users with advanced BIOS modding knowledge)
The currently latest version can be downloaded from here (as RAR archive):
>UEFITool v0.28.0< 1.4k (>MIRROR< 327)
Advantage: This tool is free available, works very reliable and can be used with **all** AMI UEFI BIOSes without any restrictions.
Attention: The usage of the UEFITool is not recommended, if the opened BIOS contains one or more listed "Pad-Files" within or just beneath the "DXE Driver Volume". A possible BIOS modding issue (removal of a natively present or addition of a natively not present "Pad File") is caused by a wrong BIOS configuration and not by the UEFITool.

5. An appropriate EFI NVME BIOS module
(has to be inserted into the "DXE Driver Volume" of the mainboard's UEFI BIOS)
You have the choice between 2 different NVMe files:
a) NvmExpressDxe_5.ffs dated 09/20/2021
(uncompressed size: 18 KB, best choice for systems without native NVMe support)
>Download link (MEGA)< 2.3k or >Mirror link (OneDrive)< 622 and

b) NvmExpressDxe_Small.ffs dated 04/01/2018
(uncompressed size: only 6 KB, recommended for BIOSes with limited DXE Driver Volume space)
>Download Link (MEGA)< 888 or >Mirror Link (OneDrive)< 282

Both sorts of NVMe modules have been compiled and optimized by our Forum member Ethaniel on the basis of the currently latest source code UDK2018, which had been built by the EDK2/Clover Team for rather different purposes regarding the NVMe usage.
The file named "NvmExpress_Small.ffs" contains just the required code to support booting off an NVMe SSD and should be taken, if there is not enough space available within the BIOS (resp. its DXE Driver Volume) for the insertion of the bigger sized "NvmExpressDxe_5" module.
Both NVMe modules have been successfully tested. For details look >here< 214 and >here< 62.

Notes:

The "Small" variant should be taken, if the BIOS tool gives you the message "File size exceeds the BIOS volume size" while trying to insert one of the above mentioned uncompressed modules.

After having successfully inserted any of the above offered NVMe modules the related name "NvmExpressDxe_5" resp. "NvmExpressDxe_Small" will be shown by the related BIOS tool.

Credits go to
a) the **EDK2 Team** resp. the Clover Team at InsanelyMac
(>LINK< 41) for the source file UDK2018,

b) our Forum member **davidm71** for his efforts to make the source file usable as additionally insertable EFI NVMe BIOS module, which is loaded while booting off an NVMe SSD in UEFI mode and

c) to our Forum member **Ethaniel** for having completely re-structured, optimized, shrinked and re-compiled the NVMe modules.

Since these special NVMe modules worked until now with **all** tested mainboards, whose BIOS is an AMI UEFI one, there is no doubt for me, that it will properly work with your special mainboard UEFI BIOS as well.

This is what you should do:

Step 1 - Preparation:

▼ **Open Guide:** (open/hide by a click)
Create a separate folder within the root of any PC partition, name it anyhow (here: D:\Source BIOS) and copy the "pure" (completely extracted) source BIOS file into that folder.

Additionally you should create another, but currently empty folder for the later created modded BIOS (example: D:\Mod BIOS).

Only for ASUS BIOSes with the suffix *.CAP:
To avoid problems while trying to flash later on the modded *.CAP file via the ASUS USB Flashback method, it is recommended to extract the "Body" of the original *.CAP BIOS. This can easily be done with the UEFITool by opening the *.CAP file, doing a right-click onto the "AMI Aptio Capsule", choosing the "Extract body..." option and saving it as *.ROM file.
>Here< you can find the complete instructions layed down by Lost_N_BIOS.

Unzip the downloaded AMI Aptio MMTool resp. the UEFITool and the chosen NvmExpressDxE module into the "Source BIOS" folder.

Step 2 - BIOS modification (2 alternative Methods):

A. Guide for the usage of AMI's MMTool 4.50.0023:

(easiest and usually successful method for AMI Aptio IV BIOSes)

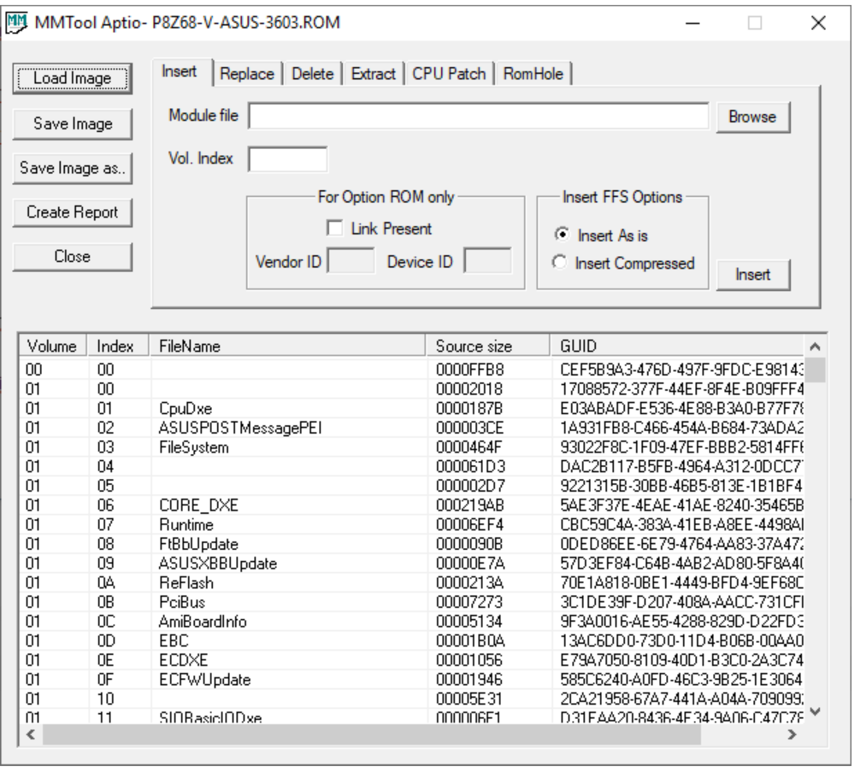
▾ **Open Guide:** (open/hide by a click)

a) Opening the BIOS file and finding the target DXE Volume:

Double-click onto the file named MMTool.exe.

Click onto the "Load Image" button and navigate to the folder, where the source BIOS is located. If it should not yet been shown within the pop-up window, choose the file type option "All files (*)", which is located at the bottom of the "MM Open" window.

After having double-clicked onto the BIOS file, which you want to get modified, you will see a picture similar to this one:

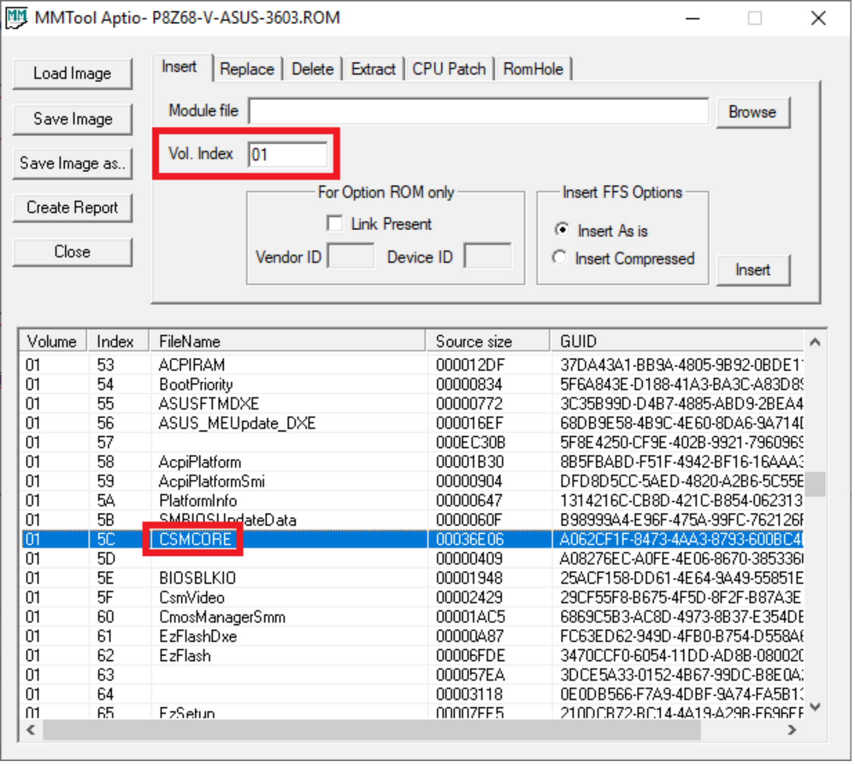


Volume	Index	FileName	Source size	GUID
00	00		0000FFB8	CEF5B9A3-476D-497F-9FDC-E9814
01	00		00002018	17088572-377F-44EF-8F4E-B09FFF4
01	01	CpuDxe	0000187B	E03ABADF-E536-4E88-B3A0-B77F7
01	02	ASUSPOSTMessagePEI	000003CE	1A931FB8-C466-454A-B684-73ADA
01	03	FileSystem	0000464F	93022F8C-1F09-47EF-BBB2-5814FF
01	04		000061D3	DAC2B117-B5FB-4964-A312-0DCC7
01	05		000002D7	9221315B-30BB-46B5-813E-1B1BF4
01	06	CORE_DXE	000219AB	5AE3F37E-4EAE-41AE-8240-35465B
01	07	Runtime	00006EF4	CBC59C4A-383A-41EB-A8EE-4498A
01	08	FtBbUpdate	0000090B	0DED86EE-6E79-4764-AA83-37A47
01	09	ASUSXBBUpdate	00000E7A	57D3EF84-C64B-4AB2-AD80-5F8A4
01	0A	ReFlash	0000213A	70E1A818-0BE1-4449-BFD4-9EF68C
01	0B	PciBus	00007273	3C1DE39F-D207-408A-AACC-731CF
01	0C	AmiBoardInfo	00005134	9F3A0016-AE55-4288-829D-D22FD
01	0D	EBC	00001B0A	13AC6DD0-73D0-11D4-B06B-00AA0
01	0E	ECDXE	00001056	E79A7050-8109-40D1-B3C0-2A3C74
01	0F	ECFWUpdate	00001946	585C6240-A0FD-46C3-9B25-1E3064
01	10		00005E31	2CA21958-67A7-441A-A04A-709099
01	11	SIOBasicIODxe	000006F1	D31FAA20-8436-4F34-9A06-C47C7

Scroll down within the great MMTool window until you find the module with the FileName **"CSMCORE"**.

Click onto the line with the FileName "CSMCORE". After having done that, the number of the Volume will be shown within the box named "Vol. Index" in the upper part of the MMTool GUI.

Here is the related picture of my test BIOS:


MMTool Aptio- P8Z68-V-ASUS-3603.ROM

Insert | Replace | Delete | Extract | CPU Patch | RomHole

Module file:

Vol. Index: 01

For Option ROM only: Link Present [ ]; Vendor ID; Device ID

Insert FFS Options: [✓ Insert As is] Insert Compressed

Volume	Index	FileName	Source size	GUID
01	53	ACPIRAM	000012DF	37DA43A1-BB9A-4805-9B92-0BDE1
01	54	BootPriority	00000834	5F6A843E-D188-41A3-BA3C-A83D8
01	55	ASUSFTMDXE	00000772	3C35B99D-D4B7-4885-ABD9-2BEA4
01	56	ASUS_MEUpdate_DXE	000016EF	68DB9E58-4B9C-4E60-8DA6-9A714
01	57		000EC30B	5F8E4250-CF9E-402B-9921-796096
01	58	AcpiPlatform	00001B30	8B5FBABD-F51F-4942-BF16-16AAA
01	59	AcpiPlatformSmi	00000904	DFD8D5CC-5AED-4820-A2B6-5C55E
01	5A	PlatformInfo	00000647	1314216C-CB8D-421C-B854-062313
01	5B	SMBIOSUpdateData	0000060F	B98999A4-E96F-475A-99FC-762126
01	5C	CSMCORE	00036E06	A062CF1F-8473-4AA3-8793-600BC4
01	5D		00000409	A08276EC-A0FE-4E06-8670-385336
01	5E	BIOSBLKIO	00001948	25ACF158-DD61-4E64-9A49-55851E
01	5F	CsmVideo	00002429	29CF55F8-B675-4F5D-8F2F-B87A3E
01	60	CmosManagerSmm	00001AC5	6869C5B3-AC8D-4973-8B37-E354DE
01	61	EzFlashDxe	00000A87	FC63ED62-949D-4FB0-B754-D558A
01	62	EzFlash	00006FDE	3470CCF0-6054-11DD-AD8B-08002
01	63		000057EA	3DCE5A33-0152-4B67-99DC-B8E0A
01	64		00003118	0E0DB566-F7A9-4DBF-9A74-FA5B1
01	65	EzSetup	00007FF5	210DCB72-BC14-4A19-A29B-F696FF

In my test BIOS it was the Volume “01”, but this is just an example. In other BIOSes the CSMCORE module may be within a Volume with another number (e.g. “04” or “02:01.00”).

Since the CSMCORE module is present within nearly all AMI UEFI BIOSes and always located within the DXE Volume, where the NVMe module has to be inserted, **you are now within the target Volume**.

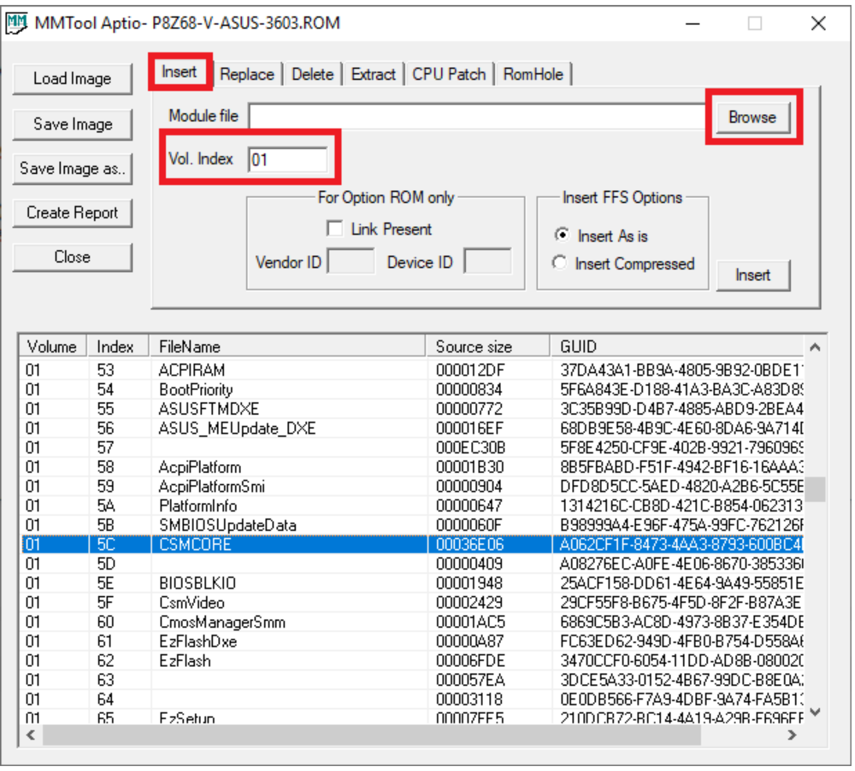
Note: In the very rare case, that the MMTool doesn’t show any module named “CSMCORE”, you should scroll down the MMTool window with the listed modules until you find the first ones with the letters “DXE” within its name. This way you can be sure, that you are within the correct “DXe Driver Volume”, where the NVMe module has to be inserted.

b) Insertion of the NVMe module

Make sure, that the correct DXE Driver **Volume number** (usually with the CSMCORE module within it) has been written into the Vol. Index box.

Click onto the "Insert" tab on the on-top MMTool menu bar.

Here is the related picture:


MMTool Aptio- P8Z68-V-ASUS-3603.ROM

Insert | Replace | Delete | Extract | CPU Patch | RomHole

Module file:

Vol. Index: 01

For Option ROM only: Link Present [ ]; Vendor ID; Device ID

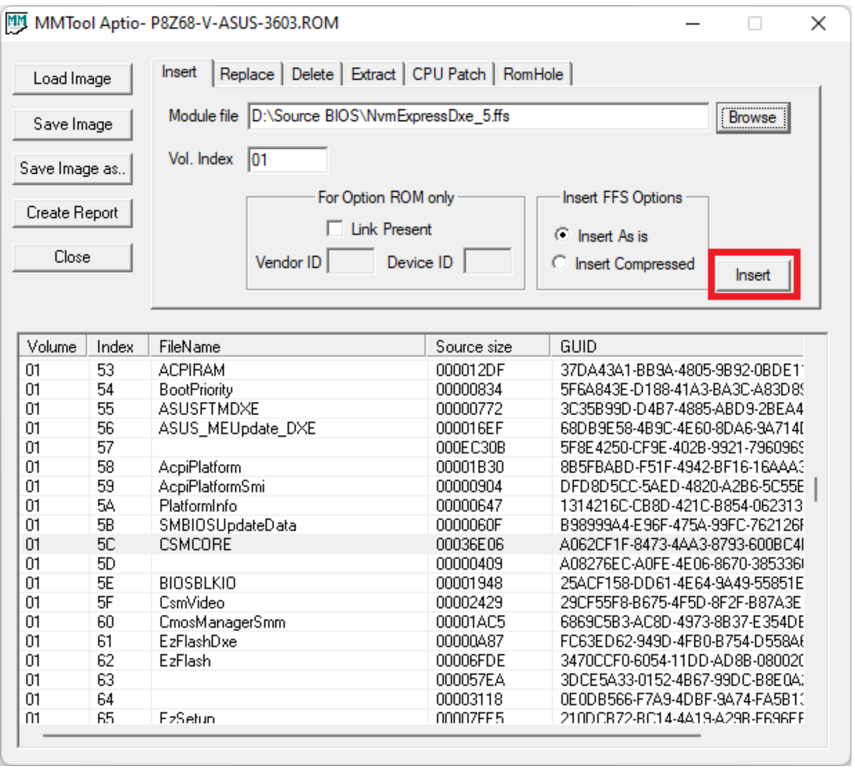
Insert FFS Options: [✓ Insert As is] Insert Compressed

Volume	Index	FileName	Source size	GUID
01	53	ACPIRAM	000012DF	37DA43A1-BB9A-4805-9B92-0BDE1
01	54	BootPriority	00000834	5F6A843E-D188-41A3-BA3C-A83D8
01	55	ASUSFTMDXE	00000772	3C35B99D-D4B7-4885-ABD9-2BEA4
01	56	ASUS_MEUpdate_DXE	000016EF	68DB9E58-4B9C-4E60-8DA6-9A714
01	57		000EC30B	5F8E4250-CF9E-402B-9921-796096
01	58	AcpiPlatform	00001B30	8B5FBABD-F51F-4942-BF16-16AAA
01	59	AcpiPlatformSmi	00000904	DFD8D5CC-5AED-4820-A2B6-5C55E
01	5A	PlatformInfo	00000647	1314216C-CB8D-421C-B854-062313
01	5B	SMBIOSUpdateData	0000060F	B98999A4-E96F-475A-99FC-762126
01	5C	CSMCORE	00036E06	A062CF1F-8473-4AA3-8793-600BC4
01	5D		00000409	A08276EC-A0FE-4E06-8670-385336
01	5E	BIOSBLKIO	00001948	25ACF158-DD61-4E64-9A49-55851E
01	5F	CsmVideo	00002429	29CF55F8-B675-4F5D-8F2F-B87A3E
01	60	CmosManagerSmm	00001AC5	6869C5B3-AC8D-4973-8B37-E354DE
01	61	EzFlashDxe	00000A87	FC63ED62-949D-4FB0-B754-D558A
01	62	EzFlash	00006FDE	3470CCF0-6054-11DD-AD8B-08002
01	63		000057EA	3DCE5A33-0152-4B67-99DC-B8E0A
01	64		00003118	0E0DB566-F7A9-4DBF-9A74-FA5B1
01	65	EzSetup	00007FF5	210DCB72-BC14-4A19-A29B-F696FF

Click onto the “Browse” button and navigate the folder, where your desired NVMe module as “pure” *.ffs file is located.

Double-click onto the *.ffs file you want to get inserted.

Now you will see a picture like this:



Module file D:\Source BIOS\NvmExpressDxe_5.ffs

Vol. Index 01

Link Present

Vendor ID Device ID

[✓ Insert As is] Insert Compressed

Volume	Index	FileName	Source size	GUID
01	53	ACPIRAM	000012DF	37DA43A1-BB9A-4805-9B92-0BDE1
01	54	BootPriority	00000834	5F6A843E-D188-41A3-BA3C-A83D8
01	55	ASUSFTMDXE	00000772	3C35B99D-D4B7-4885-ABD9-2BEA4
01	56	ASUS_MEUpdate_DXE	000016EF	68DB9E58-4B9C-4E60-8DA6-9A714
01	57		000EC30B	5F8E4250-CF9E-402B-9921-796096
01	58	AcpiPlatform	00001B30	8B5FBABD-F51F-4942-BF16-16AAA
01	59	AcpiPlatformSmi	00000904	DFD8D5CC-5AED-4820-A2B6-5C55E
01	5A	PlatformInfo	00000647	1314216C-CB8D-421C-B854-062313
01	5B	SMBIOSUpdateData	0000060F	B98999A4-E96F-475A-99FC-762126
01	5C	CSMCORE	00036E06	A062CF1F-8473-4AA3-8793-600BC4
01	5D		00000409	A08276EC-A0FE-4E06-8670-385336
01	5E	BIOSBLKIO	00001948	25ACF158-DD61-4E64-9A49-55851E
01	5F	CsmVideo	00002429	29CF55F8-B675-4F5D-8F2F-B87A3E
01	60	CmosManagerSmm	00001AC5	6869C5B3-AC8D-4973-8B37-E354DE
01	61	EzFlashDxe	00000A87	FC63ED62-949D-4FB0-B754-D558A
01	62	EzFlash	00006FDE	3470CCF0-6054-11DD-AD8B-08002
01	63		000057EA	3DCE5A33-0152-4B67-99DC-B8E0A
01	64		00003118	0E0DB566-F7A9-4DBF-9A74-FA5B1
01	65	EzSetup	00007FF5	210DCB72-BC14-4A19-A29B-F696FF

You can choose within the “Insert FFS Options” area of the MMTool GUI, whether you want to get the previously chosen module inserted “as it is” (normal option) or in “compressed” form (option in case of limited space within the DXE Volume).
Note: Don’t touch the “For Option ROM only” area of the MMTool GUI!

Now you can click onto the “Insert” button, which is on the upper right hand side below the “Browse” button (see above picture).

After having done that, the insertion of the desired NVMe module should be done automaticly and correctly by the MMTool.
Note: If the MMTool should not be able to insert the desired module properly, the MMTool will give you a meaningful error message (e.g. **“Not enough space within the Volume”**). In this case you should try to get the “small” variant of the NVMe module in “compressed” form inserted. If there should not even be enough space for this small sized module, I recommend to follow the advices given by MeatWar within the first post of >this< 19 thread.

As final step you have to store the modded BIOS:
Click onto the button "Save Image as...", navigate to the previously created folder for the modded BIOS (e.g. D:\Mod BIOS) and give the modded BIOS a meaningful name (with the same extension as the original BIOS).

Now you have stored your modded BIOS within the related folder.

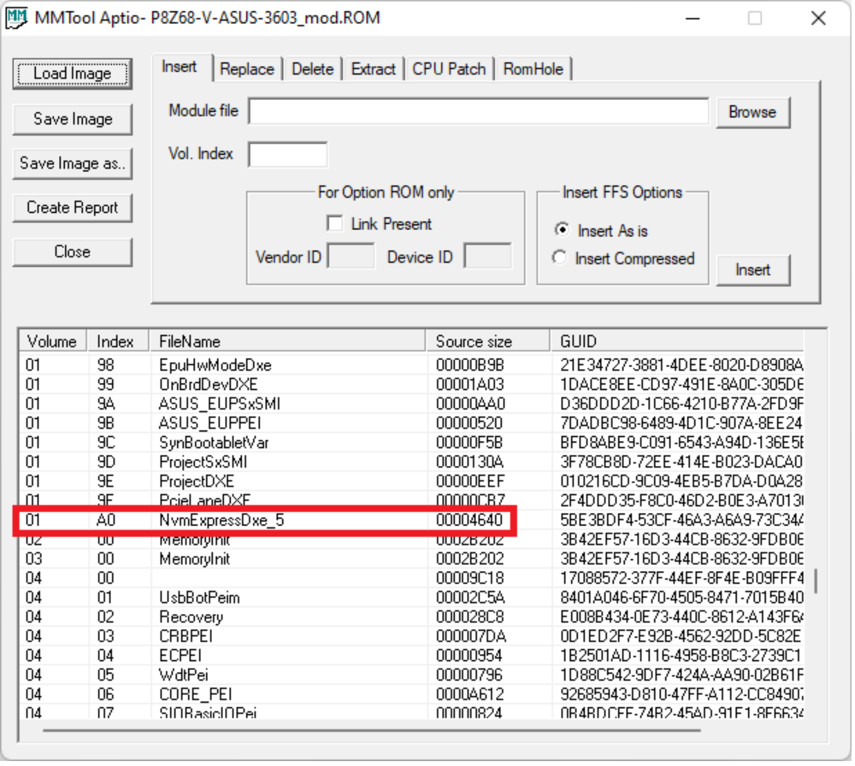
c) Verification of the successful insertion

Open the modded BIOS as written above by running the MMTool.

Scroll down the content of the MMTool window showing the various modules of the DXE Volume (where the CSMCORE is located) until you come to bottom of that Volume.

The last (undermost) "DXE Driver" module of the related Volume Number should now be new and named either "NvmExpressDxe_5" or "NvmExpressDxe_small" (depending on the module variant you had previously inserted.

A picture like this is the proof, that the NVMe module has been successfully inserted:



Volume	Index	FileName	Source size	GUID
01	98	EpuHwModeDxe	00000B9B	21E34727-3881-4DEE-8020-D8908A
01	99	OnBrdDevDXE	00001A03	1DACE8EE-CD97-491E-8A0C-305DE
01	9A	ASUS_EUPSxSMI	00000AA0	D36DDD2D-1C66-4210-B77A-2FD9F
01	9B	ASUS_EUPPEI	00000520	7DADBC98-6489-4D1C-907A-8EE24
01	9C	SynBootableVar	00000F5B	BFD8ABE9-C091-6543-A94D-136E5
01	9D	ProjectSxSMI	0000130A	3F78CB8D-72EE-414E-B023-DACA0
01	9E	ProjectDXE	00000EEF	010216CD-9C09-4EB5-B7DA-D0A28
01	9F	PcieLaneDXE	00000CB7	2F4DDD35-F8C0-46D2-B0E3-A7013
01	A0	NvmExpressDxe_5	00004640	5BE3BDF4-53CF-46A3-A6A9-73C34
02	00	MemoryInit	0002B202	3B42EF57-16D3-44CB-8632-9FDB0E
03	00	MemoryInit	0002B202	3B42EF57-16D3-44CB-8632-9FDB0E
04	00		00009C18	17088572-377F-44EF-8F4E-B09FFF4
04	01	UsbBotPeim	00002C5A	8401A046-6F70-4505-8471-7015B40
04	02	Recovery	000028C8	E008B434-0E73-440C-8612-A143F6
04	03	CRBPEI	000007DA	0D1ED2F7-E92B-4562-92DD-5C82E
04	04	ECPEI	00000954	1B2501AD-1116-4958-B8C3-2739C1
04	05	WdtPei	00000796	1D88C542-9DF7-424A-AA90-02B61F
04	06	CORE_PEI	0000A612	92685943-D810-47FF-A112-CC8490
04	07	SIOBasicIOPei	00000824	0B4BDCEF-74B2-45AD-91F1-8F663

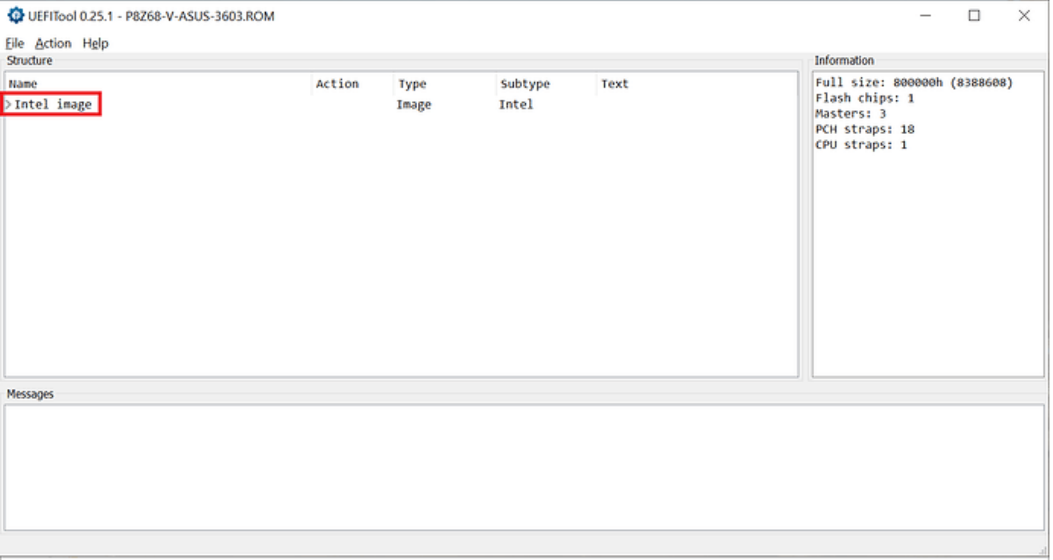
Warning:
Very rarely even AMI’s AptioIV MMTool may touch a **“Pad-file”** of the BIOS while executing its NVMe module insertion task. To avoid later problems after having flashed the modded BIOS, I recommend insistently to compare the entire “DXE Driver Volume” of the original and the just modified BIOS regarding the presence and location of Pad-files by using the UEFITool (the MMTool doesn’t show the “Pad-files”). The best visual way for such comparison is to open the original and modded BIOS side by side. For details look >here<. The only difference between the original and the modded BIOS should be, that a new DXE driver named “NvmExpressDxe_5” resp. “NvmExpressDxe_small” has been added. All other listed modules and Pad-files should stay untouched by the BIOS modification.
If you should realize, that a natively present Pad-file module is missing within the modded BIOS or a natively not present Pad-file has been added within the modded BIOS, post a report into this thread, but don’t flash this modded BIOS!
If everything looks fine, you can be sure, that the related module has been properly inserted into the correct GUID section, and you can save the modded BIOS and give it an appropriate name and suffix (usually similar to the original BIOS).
Only for ASUS BIOSes, whose capsule had been removed by extracting the “Body”, but shall be flashed by using the ASUS USB Flashback feature:
Don’t forget to re-insert the original capsule and to save the modded BIOS as *.CAP file according to >this< 187 guide written by MeatWar.

B. Guide for the usage of the UEFITool:
(only recommended for users with advanced BIOS modding knowledge)
▾ **Open Guide:** (open/hide by a click)
a) Opening the BIOS file and finding the target DXE Volume:
Double-click onto the file named UEFITool.exe.

Use the full size option to see the complete content of the UEFITool GUI.

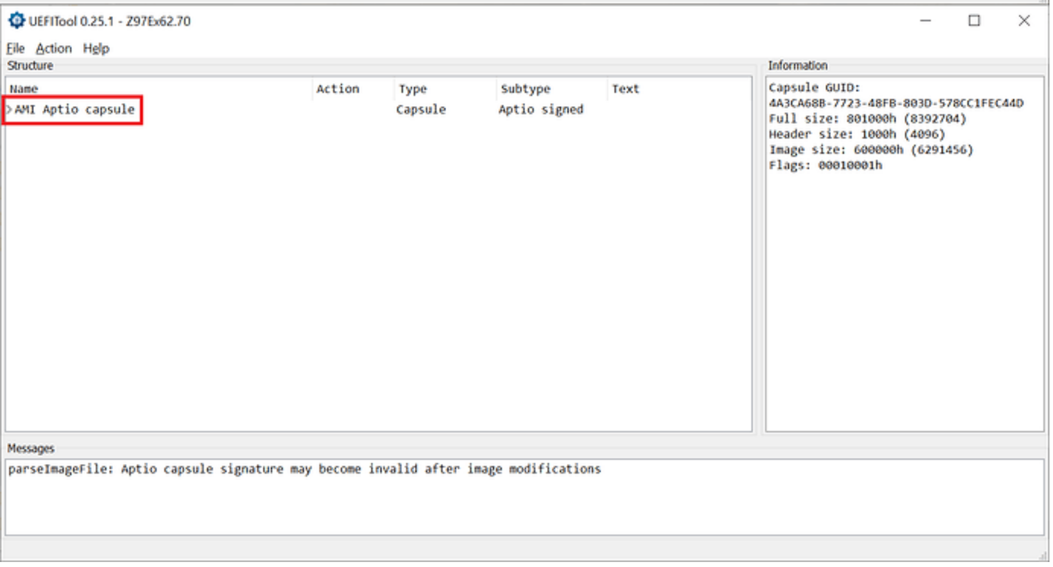
Click onto "File" from the UEFITool menu bar, choose the option "Open image file..." and navigate to the folder, where the source BIOS is located. If it should not yet been shown within the pop-up window, choose the file type option "All files (*)", which is located just above the "Open" button.

After having double-clicked onto the BIOS file, which you want to get modified, you will see a picture similar to one of these:



```
Name                  Action   Type     Subtype   Text
> Intel image                  Image    Intel
```

```
Full size: 800000h (8388608)
Flash chips: 1
Masters: 3
PCH straps: 18
CPU straps: 1
```



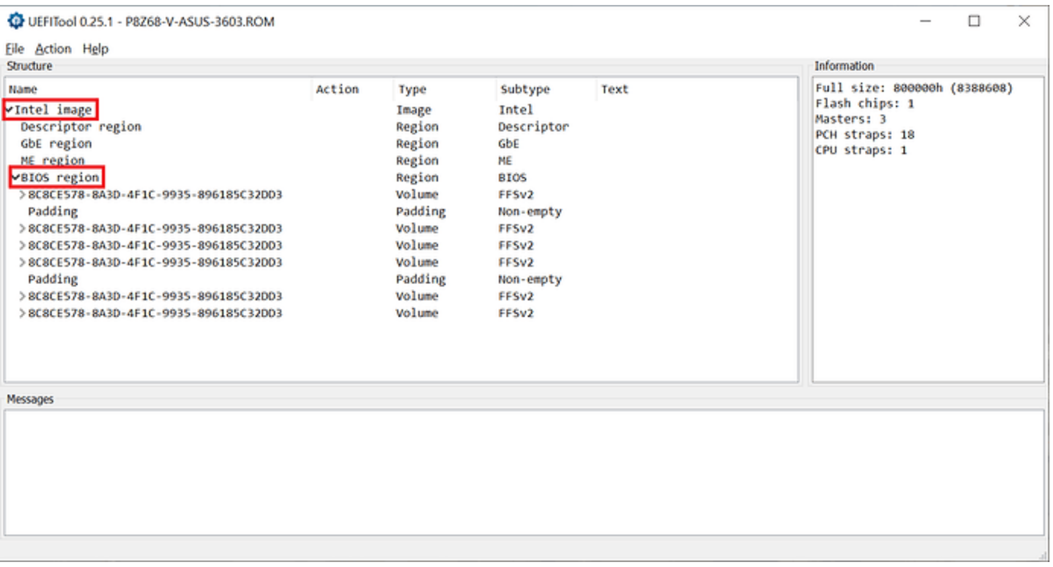
```
Name                  Action   Type     Subtype        Text
> AMI Aptio capsule            Capsule  Aptio signed
```

```
Capsule GUID:
4A3CA68B-7723-48FB-803D-578CC1FEC44D
Full size: 801000h (8392704)
Header size: 1000h (4096)
Image size: 600000h (6291456)
Flags: 00010001h
```

```
parseImageFile: Aptio capsule signature may become invalid after image modifications
```

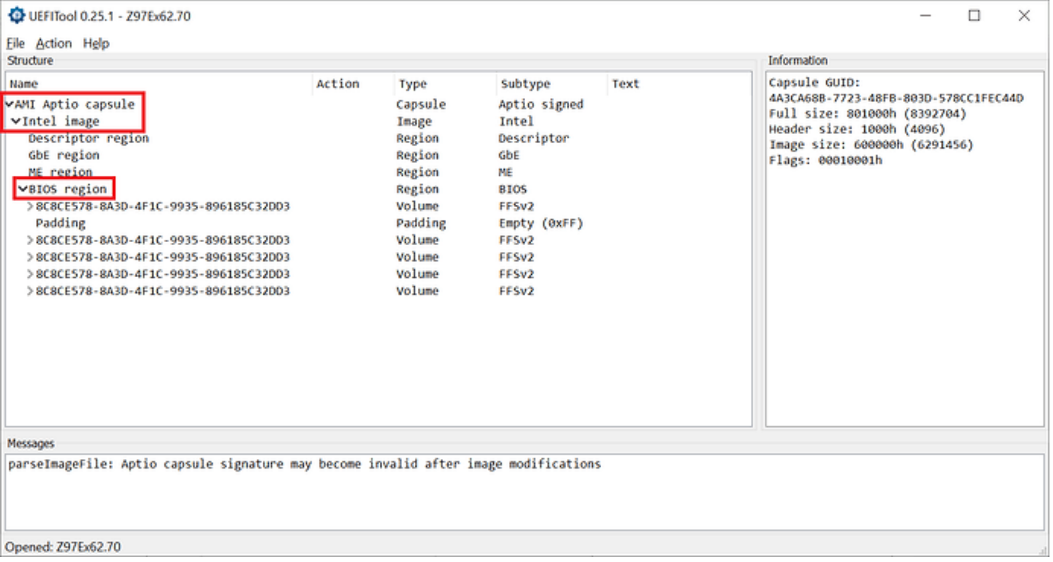
Expand the shown image resp. capsule by clicking onto the right-direction arrow in front of them. Now you should see the main contents of the BIOS inclusive the most important "BIOS region".

After having expanded the "BIOS region" by hitting onto the right-direction arrow in front of it, you will see a picture like one of these:



```
Name                                      Action   Type     Subtype      Text
v Intel image                                      Image    Intel
   Descriptor region                               Region   Descriptor
   GbE region                                      Region   GbE
   ME region                                       Region   ME
  v BIOS region                                    Region   BIOS
   > 8C8CE578-8A3D-4F1C-9935-896185C32DD3          Volume   FFSv2
     Padding                                       Padding  Non-empty
   > 8C8CE578-8A3D-4F1C-9935-896185C32DD3          Volume   FFSv2
   > 8C8CE578-8A3D-4F1C-9935-896185C32DD3          Volume   FFSv2
   > 8C8CE578-8A3D-4F1C-9935-896185C32DD3          Volume   FFSv2
     Padding                                       Padding  Non-empty
   > 8C8CE578-8A3D-4F1C-9935-896185C32DD3          Volume   FFSv2
   > 8C8CE578-8A3D-4F1C-9935-896185C32DD3          Volume   FFSv2
```

```
Full size: 800000h (8388608)
Flash chips: 1
Masters: 3
PCH straps: 18
CPU straps: 1
```


UEFITool 0.25.1 - Z97Ex62.70

File Action Help

Structure

```
Name                                    Action   Type       Subtype         Text
⌄AMI Aptio capsule                               Capsule    Aptio signed
 ⌄Intel image                                    Image      Intel
   Descriptor region                             Region     Descriptor
   GbE region                                    Region     GbE
   ME region                                     Region     ME
  ⌄BIOS region                                   Region     BIOS
   >8C8CE578-8A3D-4F1C-9935-896185C32DD3         Volume     FFSv2
    Padding                                      Padding    Empty (0xFF)
   >8C8CE578-8A3D-4F1C-9935-896185C32DD3         Volume     FFSv2
   >8C8CE578-8A3D-4F1C-9935-896185C32DD3         Volume     FFSv2
   >8C8CE578-8A3D-4F1C-9935-896185C32DD3         Volume     FFSv2
   >8C8CE578-8A3D-4F1C-9935-896185C32DD3         Volume     FFSv2
```

Information

```
Capsule GUID:
4A3CA68B-7723-48FB-803D-578CC1FEC44D
Full size: 801000h (8392704)
Header size: 1000h (4096)
Image size: 600000h (6291456)
Flags: 00010001h
```

Messages

```
parseImageFile: Aptio capsule signature may become invalid after image modifications
```

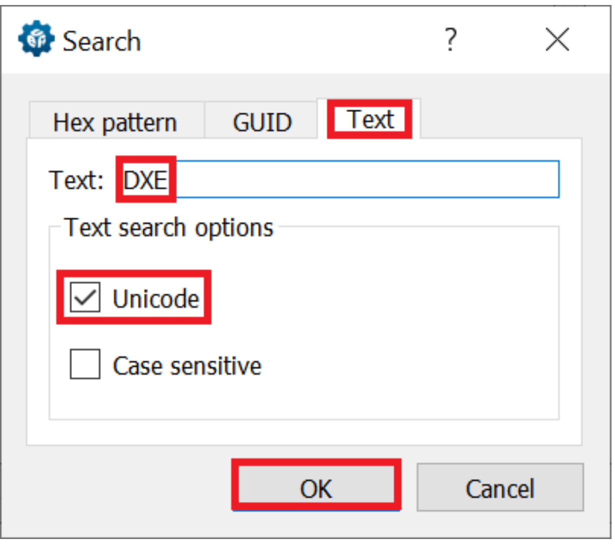
Opened: Z97Ex62.70

Due to the complicated and mainboard specific internal structure of the "BIOS region", it may be rather time consuming to find the DXE Volume, where all the EFI modules are located and where an additional EFI module like the NVMe one has to be inserted. The easiest way to find it, is to use the "Search" option of the UEFITool.

This is what I recommend to do after having opened the source BIOS with the UEFITool:

Click onto the menu tab "File" and choose the option "Search...". Now you will get access to the UEFITool "Search" pop-up window.

Click onto the tab "Text" and enter the word "DXE" into the Text box (the "Unicode" option will be checked by default). This is the related picture:

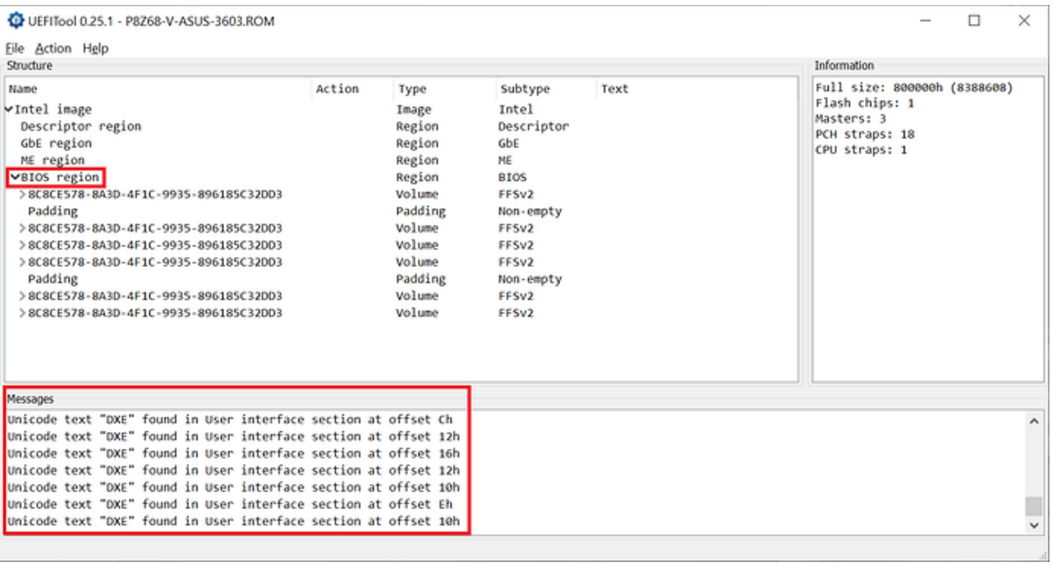


Text: DXE

[✓] Unicode

[ ] Case sensitive

Once you have hit the "OK" button, you will get a picture like this:


UEFITool 0.25.1 - P8Z68-V-ASUS-3603.ROM

File Action Help

Structure

```
Name                                    Action   Type       Subtype         Text
⌄Intel image                                     Image      Intel
  Descriptor region                              Region     Descriptor
  GbE region                                     Region     GbE
  ME region                                      Region     ME
 ⌄BIOS region                                    Region     BIOS
  >8C8CE578-8A3D-4F1C-9935-896185C32DD3          Volume     FFSv2
   Padding                                       Padding    Non-empty
  >8C8CE578-8A3D-4F1C-9935-896185C32DD3          Volume     FFSv2
  >8C8CE578-8A3D-4F1C-9935-896185C32DD3          Volume     FFSv2
  >8C8CE578-8A3D-4F1C-9935-896185C32DD3          Volume     FFSv2
   Padding                                       Padding    Non-empty
  >8C8CE578-8A3D-4F1C-9935-896185C32DD3          Volume     FFSv2
  >8C8CE578-8A3D-4F1C-9935-896185C32DD3          Volume     FFSv2
```

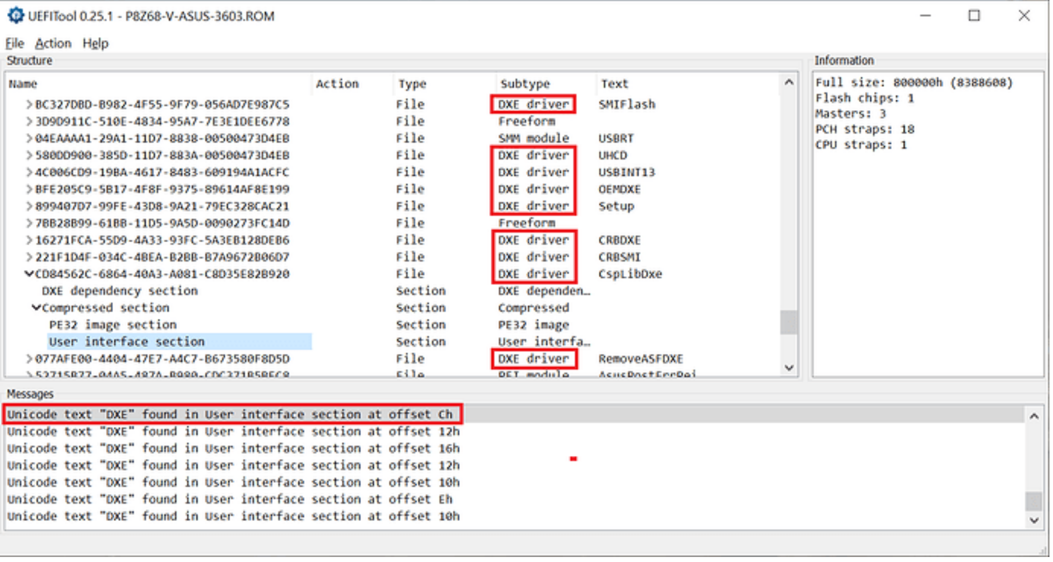
Information

```
Full size: 800000h (8388608)
Flash chips: 1
Masters: 3
PCH straps: 18
CPU straps: 1
```

Messages

```
Unicode text "DXE" found in User interface section at offset Ch
Unicode text "DXE" found in User interface section at offset 12h
Unicode text "DXE" found in User interface section at offset 16h
Unicode text "DXE" found in User interface section at offset 12h
Unicode text "DXE" found in User interface section at offset 10h
Unicode text "DXE" found in User interface section at offset Eh
Unicode text "DXE" found in User interface section at offset 10h
```

Double click onto any line of the "Messages" you can see at the bottom of the UEFITool GUI. The result is a picture like this one:

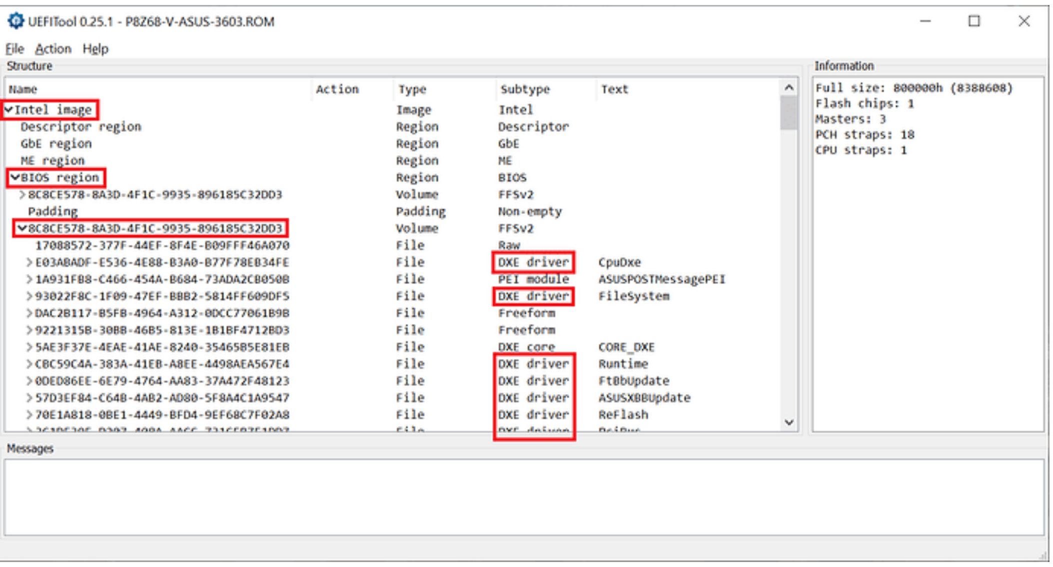


```
UEFITool 0.25.1 - P8Z68-V-ASUS-3603.ROM
File  Action  Help
Structure
Name                                      Action   Type      Subtype        Text
 > BC327DBD-B982-4F55-9F79-056AD7E987C5            File      DXE driver     SMIFlash
 > 3D9D911C-510E-4834-95A7-7E3E1DEE6778            File      Freeform
 > 04EAAAA1-29A1-11D7-8838-00500473D4EB            File      SMM module     USBRT
 > 580DD900-385D-11D7-883A-00500473D4EB            File      DXE driver     UHCD
 > 4C006CD9-19BA-4617-8483-609194A1ACFC            File      DXE driver     USBINT13
 > BFE205C9-5B17-4F8F-9375-89614AF8E199            File      DXE driver     OEMDXE
 > 899407D7-99FE-43D8-9A21-79EC328CAC21            File      DXE driver     Setup
 > 7BB28B99-61BB-11D5-9A5D-0090273FC14D            File      Freeform
 > 16271FCA-55D9-4A33-93FC-5A3EB128DEB6            File      DXE driver     CRBDXE
 > 221F1D4F-034C-4BEA-B2BB-B7A9672B06D7            File      DXE driver     CRBSMI
 v CD84562C-6864-40A3-A081-C8D35E82B920            File      DXE driver     CspLibDxe
     DXE dependency section                        Section   DXE dependen…
   v Compressed section                            Section   Compressed
       PE32 image section                          Section   PE32 image
       User interface section                      Section   User interfa…
 > 077AFE00-4404-47E7-A4C7-B673580F8D5D            File      DXE driver     RemoveASFDXE

Information
Full size: 800000h (8388608)
Flash chips: 1
Masters: 3
PCH straps: 18
CPU straps: 1

Messages
Unicode text "DXE" found in User interface section at offset Ch
Unicode text "DXE" found in User interface section at offset 12h
Unicode text "DXE" found in User interface section at offset 16h
Unicode text "DXE" found in User interface section at offset 12h
Unicode text "DXE" found in User interface section at offset 10h
Unicode text "DXE" found in User interface section at offset Eh
Unicode text "DXE" found in User interface section at offset 10h
```

As you can see, you are now within the "Compressed section" of a "DXE driver" (which one doesn't matter).

This verifies, that you are now within the DXE Volume of the BIOS (= location, where the EFI modules are stored and can be inserted).

If you close the sub-folders of the related DXE driver GUID by clicking onto the downside arrow in front of it, you will get a picture like this, where all the GUIDs are listed, which are inside the DXE Volume:



```
UEFITool 0.25.1 - P8Z68-V-ASUS-3603.ROM
File  Action  Help
Structure
Name                                      Action   Type      Subtype        Text
v Intel image                                      Image     Intel
   Descriptor region                               Region    Descriptor
   GbE region                                      Region    GbE
   ME region                                       Region    ME
 v BIOS region                                     Region    BIOS
   > 8C8CE578-8A3D-4F1C-9935-896185C32DD3          Volume    FFSv2
     Padding                                       Padding   Non-empty
   v 8C8CE578-8A3D-4F1C-9935-896185C32DD3          Volume    FFSv2
       17088572-377F-44EF-8F4E-B09FFF46A070        File      Raw
     > E03ABADF-E536-4E88-B3A0-B77F78EB34FE        File      DXE driver     CpuDxe
     > 1A931FB8-C466-454A-B684-73ADA2CB050B        File      PEI module     ASUSPOSTMessagePEI
     > 93022F8C-1F09-47EF-BBB2-5814FF609DF5        File      DXE driver     FileSystem
     > DAC2B117-B5FB-4964-A312-0DCC77061B9B        File      Freeform
     > 9221315B-30BB-46B5-813E-1B1BF4712BD3        File      Freeform
     > 5AE3F37E-4EAE-41AE-8240-35465B5E81EB        File      DXE core       CORE_DXE
     > CBC59C4A-383A-41EB-A8EE-4498AEA567E4        File      DXE driver     Runtime
     > 0DED86EE-6E79-4764-AA83-37A472F48123        File      DXE driver     FtBbUpdate
     > 57D3EF84-C64B-4AB2-AD80-5F8A4C1A9547        File      DXE driver     ASUSXBBUpdate
     > 70E1A818-0BE1-4449-BFD4-9EF68C7F02A8        File      DXE driver     ReFlash

Information
Full size: 800000h (8388608)
Flash chips: 1
Masters: 3
PCH straps: 18
CPU straps: 1

Messages
```

Important: Make sure, that the target DXE Volume contains a module named “CSMCORE”.

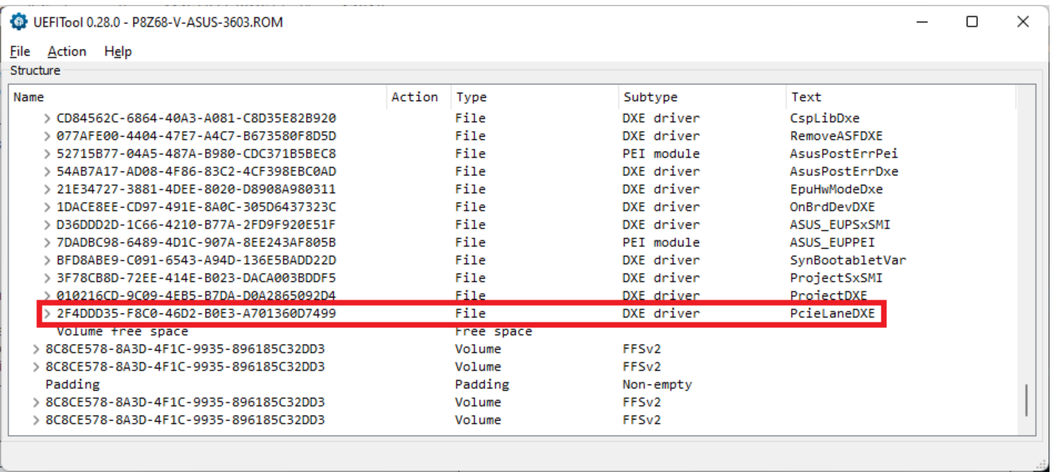
Reason: Some BIOSes contain more than 1 DXE Volume, but only the one with the CSMCORE module is suitable for the insertion of a natively not present EFI module.

Note: In the very rare case, that no module named CSMCORE is listed, you have to search for the upmost Volume, which contains modules with the letters “DXE” within their name.

b) Insertion of the NVMe module

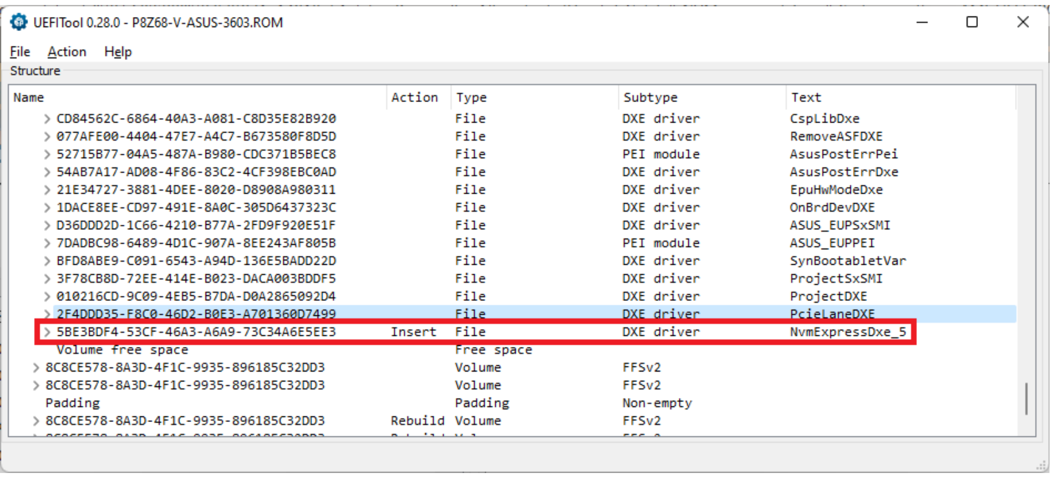
Once you have expanded the DXE Volume, scroll down to the **undermost module, which is listed within the DXE Volume and has a GUID** (no matter, whether its Subtype is "DXE driver" or "Freeform" and whether it is listed at the rightmost column with a "text").

Here is the related picture (**in this example** it is the DXE File named "PcieLaneDXE", but - depending on the specific BIOS - it may have any other or even no name):



```
UEFITool 0.28.0 - P8Z68-V-ASUS-3603.ROM
File  Action  Help
Structure
Name                                          Action  Type         Subtype      Text
    > CD84562C-6864-40A3-A081-C8D35E82B920            File         DXE driver   CspLibDxe
    > 077AFE00-4404-47E7-A4C7-B673580F8D5D            File         DXE driver   RemoveASFDXE
    > 52715B77-04A5-487A-B980-CDC371B5BEC8            File         PEI module   AsusPostErrPei
    > 54AB7A17-AD08-4F86-83C2-4CF398EBC0AD            File         DXE driver   AsusPostErrDxe
    > 21E34727-3881-4DEE-8020-D8908A980311            File         DXE driver   EpuHwModeDxe
    > 1DACE8EE-CD97-491E-8A0C-305D6437323C            File         DXE driver   OnBrdDevDXE
    > D36DDD2D-1C66-4210-B77A-2FD9F920E51F            File         DXE driver   ASUS_EUPSxSMI
    > 7DADBC98-6489-4D1C-907A-8EE243AF805B            File         PEI module   ASUS_EUPPEI
    > BFD8ABE9-C091-6543-A94D-136E5BADD22D            File         DXE driver   SynBootableVar
    > 3F78CB8D-72EE-414E-B023-DACA003BDDF5            File         DXE driver   ProjectSxSMI
    > 010216CD-9C09-4EB5-B7DA-D0A2865092D4            File         DXE driver   ProjectDXE
    > 2F4DDD35-F8C0-46D2-B0E3-A701360D7499            File         DXE driver   PcieLaneDXE
      Volume free space                               Free space
  > 8C8CE578-8A3D-4F1C-9935-896185C32DD3              Volume       FFSv2
  > 8C8CE578-8A3D-4F1C-9935-896185C32DD3              Volume       FFSv2
    Padding                                           Padding      Non-empty
  > 8C8CE578-8A3D-4F1C-9935-896185C32DD3              Volume       FFSv2
  > 8C8CE578-8A3D-4F1C-9935-896185C32DD3              Volume       FFSv2
```

Right-click onto the GUID of the **undermost** listed "DXE driver" or "Freeform" module, choose the option "Insert after...", navigate to the EFI NVMe module file you want to insert (must have the suffix *.ffs!), and double-click onto it.
After having done that, you hopefully will see a picture like this:



```
UEFITool 0.28.0 - P8Z68-V-ASUS-3603.ROM
File  Action  Help
Structure
Name                                         Action  Type        Subtype        Text
  > CD84562C-6864-40A3-A081-C8D35E82B920             File        DXE driver     CspLibDxe
  > 077AFE00-4404-47E7-A4C7-B673580F8D5D             File        DXE driver     RemoveASFDXE
  > 52715B77-04A5-487A-B980-CDC371B5BEC8             File        PEI module     AsusPostErrPei
  > 54AB7A17-AD08-4F86-83C2-4CF398EBC0AD             File        DXE driver     AsusPostErrDxe
  > 21E34727-3881-4DEE-8020-D8908A980311             File        DXE driver     EpuHwModeDxe
  > 1DACE8EE-CD97-491E-8A0C-305D6437323C             File        DXE driver     OnBrdDevDXE
  > D36DDD2D-1C66-4210-B77A-2FD9F920E51F             File        DXE driver     ASUS_EUPSxSMI
  > 7DADBC98-6489-4D1C-907A-8EE243AF805B             File        PEI module     ASUS_EUPPEI
  > BFD8ABE9-C091-6543-A94D-136E5BADD22D             File        DXE driver     SynBootableVar
  > 3F78CB8D-72EE-414E-B023-DACA003BDDF5             File        DXE driver     ProjectSxSMI
  > 010216CD-9C09-4EB5-B7DA-D0A2865092D4             File        DXE driver     ProjectDXE
  > 2F4DDD35-F8C0-46D2-B0E3-A701360D7499             File        DXE driver     PcieLaneDXE
  > 5BE3BDF4-53CF-46A3-A6A9-73C34A6E5EE3     Insert  File        DXE driver     NvmExpressDxe_5
    Volume free space                                Free space
> 8C8CE578-8A3D-4F1C-9935-896185C32DD3               Volume      FFSv2
> 8C8CE578-8A3D-4F1C-9935-896185C32DD3               Volume      FFSv2
  Padding                                            Padding     Non-empty
> 8C8CE578-8A3D-4F1C-9935-896185C32DD3       Rebuild Volume      FFSv2
```

Note: If the UEFITool should not be able to insert the desired module properly, it will give you the message **“Not enough space within the Volume”**. In this case you should try to get the “small” variant of the NVMe module inserted. If there should not even be enough space for this small sized module, I recommend to follow the advices given by MeatWar within the first post of >this< 19 thread.

Don't forget to save the modded BIOS as final step of the modding procedure. You can start it either by clicking onto "File" > "Save image file..." or by hitting CTRL+S. Then you can navigate to the folder, which has been designed for the modded BIOS (e.g. D:\Mod BIOS), choose a meaningful BIOS file name with an appropriate suffix (you can use any suffix, if you choose the "All files" option) and click onto the "Save" button.

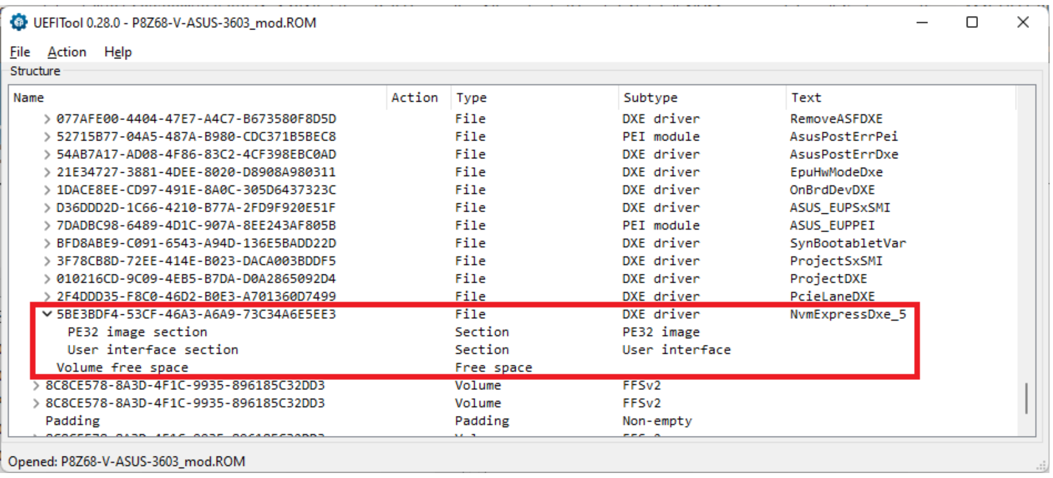
c) Verification of the successful insertion

Immediately after you have saved the modded BIOS the UEFITool will offer you the option to open the reconstructed file. Click onto "Yes".
If you had already closed the UEFITool, you will have to re-run the tool as written above.

Expand the content of the "BIOS region" and its GUIDs until you have found the specific GUID again, which contains the modules of the Subtype "DXE driver".

Scroll down and search for the "DXE driver" with the right hand text "NvmExpressDxE_5" resp. "NvmExpressDxe_small" (depending on what you had inserted).

Expand the sub-sections of the freshly inserted NVMe module by clicking onto the downside arrows.
This is, what you should see:



```
UEFITool 0.28.0 - P8Z68-V-ASUS-3603_mod.ROM
File  Action  Help
Structure
Name                                         Action  Type        Subtype        Text
  > 077AFE00-4404-47E7-A4C7-B673580F8D5D             File        DXE driver     RemoveASFDXE
  > 52715B77-04A5-487A-B980-CDC371B5BEC8             File        PEI module     AsusPostErrPei
  > 54AB7A17-AD08-4F86-83C2-4CF398EBC0AD             File        DXE driver     AsusPostErrDxe
  > 21E34727-3881-4DEE-8020-D8908A980311             File        DXE driver     EpuHwModeDxe
  > 1DACE8EE-CD97-491E-8A0C-305D6437323C             File        DXE driver     OnBrdDevDXE
  > D36DDD2D-1C66-4210-B77A-2FD9F920E51F             File        DXE driver     ASUS_EUPSxSMI
  > 7DADBC98-6489-4D1C-907A-8EE243AF805B             File        PEI module     ASUS_EUPPEI
  > BFD8ABE9-C091-6543-A94D-136E5BADD22D             File        DXE driver     SynBootableVar
  > 3F78CB8D-72EE-414E-B023-DACA003BDDF5             File        DXE driver     ProjectSxSMI
  > 010216CD-9C09-4EB5-B7DA-D0A2865092D4             File        DXE driver     ProjectDXE
  > 2F4DDD35-F8C0-46D2-B0E3-A701360D7499             File        DXE driver     PcieLaneDXE
  v 5BE3BDF4-53CF-46A3-A6A9-73C34A6E5EE3             File        DXE driver     NvmExpressDxe_5
      PE32 image section                             Section     PE32 image
      User interface section                         Section     User interface
    Volume free space                                Free space
> 8C8CE578-8A3D-4F1C-9935-896185C32DD3               Volume      FFSv2
> 8C8CE578-8A3D-4F1C-9935-896185C32DD3               Volume      FFSv2
  Padding                                            Padding     Non-empty
Opened: P8Z68-V-ASUS-3603_mod.ROM
```

Warning:
In some cases the UEFITool may remove an existing **“Pad-file”** of the BIOS or create a natively not present Pad-file while executing its NVMe module insertion task (trying to “repair” a supposedly wrong BIOS structure). To avoid later problems after having flashed the modded BIOS, I recommend insistently to compare the entire “DXE Driver Volume” of the original and the just modified BIOS regarding the presence and location of Pad-files by using the UEFITool (the MMTool doesn’t show the “Pad-files”). The best visual way for such comparison is to open the original and modded BIOS side by side. For details look >here<. The only difference between the original and the modded BIOS should be, that a new DXE driver named “NvmExpressDxe_5” resp. “NvmExpressDxe_small” has been added. All other listed modules should be untouched by the BIOS modification.
If you should realize, that within the modded BIOS a natively present Pad-file module is missing resp. has been moved or a natively not present Pad-file has been added by the UEFITool, post a report into this thread, but don’t flash this modded BIOS!

If everything looks fine, you can be sure, that the related module has been properly inserted into the correct GUID section, and you can save the modded BIOS and give it an appropriate name and suffix (usually similar to the original BIOS).

Only for ASUS BIOSes, whose capsule had been removed by extracting the "Body", but shall be flashed by using the ASUS USB Flashback feature:
Don't forget to re-insert the original capsule and to save the modded BIOS as *.CAP file according to >this< 187 guide written by MeatWar.

Step 3 - Flashing of the modded BIOS:

The related detailed Guide can be found >here<. If you have a question regarding the BIOS flashing procedure or if you should run into a problem while trying to flash your modded BIOS, please post into >this< 189 thread and not here.

Warning:

Please keep in mind, that this is the only really risky part of this Guide.
For BIOS modding beginners it is strictly recommended
a) to double-check the integrity of their modded BIOS and
b) to read carefully >this< Guide
before flashing any modified BIOS.

▾ **Additional Notes:** (open/hide by a click)

The preparation and the exact BIOS Flashing procedure depends on the mainboard's manufacturer and maybe even on the mainboard model (for details look into the manual of your mainboard).

As medium for the modded BIOS during the flashing procedure you should take a 100% working and freshly FAT32 formatted USB 2.0 Flash Drive. It should contain just the modded "pure" BIOS file and nothing else.

It is not recommended to flash a modded BIOS from within the OS.

After having successfully flashed the modded BIOS, don't forget to re-enter the BIOS, to redo your personal BIOS settings and to make sure, that your system will be able to boot in UEFI mode.

Users, who don't know where to find the specific BIOS settings, which are mentioned within this Guide, should set the BIOS language to "English".

Success verification (the NVMe SSD has to be connected!):
Enter the "BOOT" section of the BIOS and make sure, that "CSM" has been set to "Enabled". Then look for the listed bootable Disk Drives.

If you now can see a new disk drive named "PATA" or "PATA_SS", you can be sure,
a) that you have correctly inserted the NVMe module into the BIOS and
b) that the modded BIOS has been successfully flashed.

Don't try to boot off the listed disk drive named "PATA" or "PATA_SS"! You will not succeed, because the required Option ROM is usually missing within the NVMe SSD. Nobody (except the user of a Samsung 950 PRO) is able to boot off an NVMe SSD in LEGACY mode.

Step 4 - Installation of Win10/11 onto the NVMe SSD:

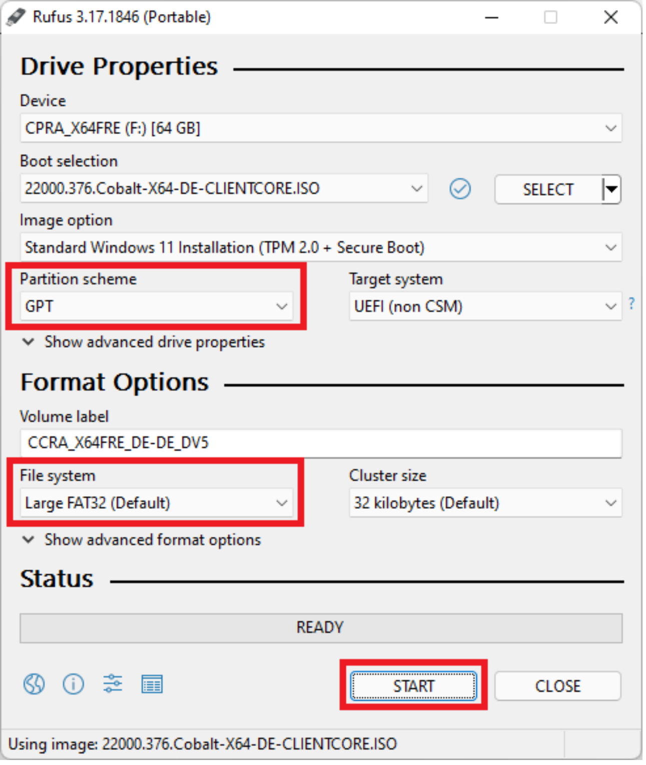
▾ **Open Guide:** (open/hide by a click)

A. Preparation/Notes:
If not already done, you can now insert the NVMe SSD resp. its M.2>PCIe adapter into the M.2 port resp. PCIe slot, which offers the best possible performance and doesn't share the PCI lanes with other devices (please look into your mainboard manual).

Enter the BIOS and look for the changes, which were caused by the BIOS modification (insertion of the NVMe EFI module). Don't be afraid about the fact, that the **NVMe SSD's manufacturer and model will not be shown there** (it will be bootable nevertheless later on). The only difference should be, that within the "BOOT" section now a new **"Legacy Mode"** device named **"PATA"** or **"PATA_SS"** is listed (only visible, when CSM has been temporarily set to "Enabled"). That is your NVMe SSD, but don't try to boot off it - only Samsung 950 Pro SSDs can boot in Legacy mode, because it has the required NVMe Option ROM in-the-box. All other NVMe SSDs can only boot in "UEFI Mode", but this requires the creation of an EFI Boot Sector named **"Windows Boot Manager"**, which will be done during the installation of the OS (see next step).

B. Best/safest procedure to get Win10/11 properly installed onto an NVMe SSD:
Save the important data, which are currently on the NVMe SSD.

Create an UEFI mode bootable USB Flash drive containing the desired Win10 image by using the latest version of the tool Rufus (important: choose the UEFI mode partition table = GPT).
Here is a picture, which shows the most important Rufus settings:


Rufus 3.17.1846 (Portable)

Drive Properties

Device: CPRA_X64FRE (F:) [64 GB]

Boot selection: 22000.376.Cobalt-X64-DE-CLIENTCORE.ISO

Image option: Standard Windows 11 Installation (TPM 2.0 + Secure Boot)

Partition scheme: GPT

Target system: UEFI (non CSM)

Format Options

Volume label: CCRA_X64FRE_DE-DE_DV5

File system: Large FAT32 (Default)

Cluster size: 32 kilobytes (Default)

Status

READY

Using image: 22000.376.Cobalt-X64-DE-CLIENTCORE.ISO

Important note: Usually the USB Flash Drive has to be **FAT32** formatted to be bootable in **UEFI mode**, but this file system cannot be created, if any file within the ISO file (e.g. the install.wim) is bigger sized than 4 GB. In this case the tool Rufus will automaticly format the USB Flash Drive by using the **NTFS** file system, but nevertheless makes it possible to boot the USB Flash Drive in **UEFI mode** (provided, that the **"Secure Boot"** option within the BIOS has been set to **"Disabled"**).

Enter the BIOS and navigate to the "BOOT" section and - if applicable - the "SECURITY" or "Keys" section.
Make sure, that the "Secure Boot" and "Fast Boot" options have been set to **Disabled**. The "Compatibility Support Module" (CSM) can either be set to "Disabled" as well (better option, but requires full UEFI compatibility of the graphics adapter) or to "Enabled" with the ability/preference to load EFI BIOS modules for the Storage Disk Drives. If you see BIOS options for the "OS type", choose "other OS". This will disable the Secure Boot setting.
Tip to avoid SATA/NVMe interferences during the OS installation:
It is recommended to temporarily unplug all SATA connected devices and additionally to disable the on-board SATA Controller(s) within the "Storage Configuration" section of the BIOS before starting the OS installation.
These measures of precaution can/should be reversed once the OS installation onto the NVMe SSD has been successfully completed.

Insert the prepared USB Flash drive and boot off it in UEFI mode (the related bootable USB drive should be shown by the Boot Manager with the prefix "[UEFI]").

When you come to the point, where you have to decide onto which Drive and which partition the OS shall be installed, **delete all existing partitions** from your NVME supporting SSD.
After having done that, let the Win10/11 Setup create a new partition for your future drive C: on the related SSD.
Then point to this just created partition as the desired future OS location.

The rest should be done by the Setup automaticly.
You will get a message, that some additional partitions have to be created. Accept that and follow the advice of the Setup where to install the OS.

Once the OS is up and running, shut down the computer, remove the bootable USB Flash driver and reconnect all your previously used storage drives.

Before you restart cour computer, make sure, that the NVMe SSD resp. its listed "Windows Boot Manager" is on top of the bootable storage drives.

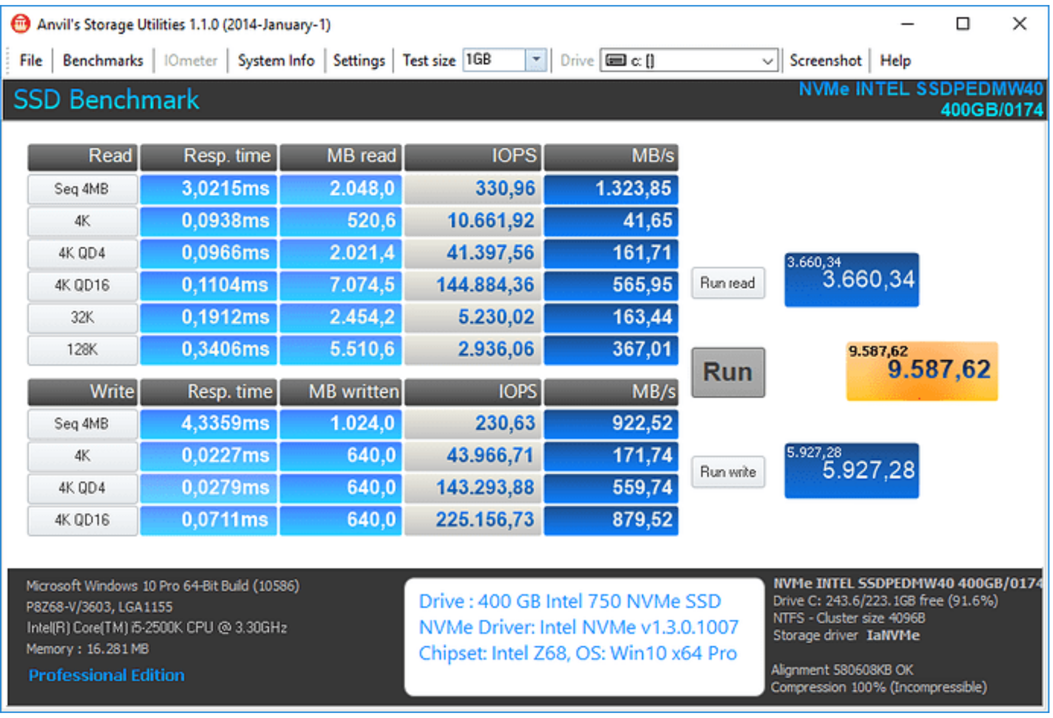
C. Data Recovery from the previously used system drive: (not recommended)
Since the required **boot sector** of an **NVMe** SSD is **not compatible** with the boot sector of a **SATA** SSD (although both boot partitions may be named "Windows Boot Manager"), you will not succeed by simply transfering the data of your previously used SATA SSD to the NVMe SSD.
On the other hand a clean install of the currently latest OS version is a good opportunity to get rid of the complete data garbage, which has been accumulated in the past and decreased the performance of the system drive.
That is why I strongly recommend to do a clean OS installation onto the NVMe SSD and not to try a recovery of the previous used drive C: data.

This is what you hopefully will get:

▾ **Benchmark Results:** (open/hide by a click)

Here are some benchmark results I got with my Z68 system running 2 different NVMe supporting SSDs:

400 GB Intel 750 PCIe SSD:



NVMe INTEL SSDPEDMW40 400GB/0174

Read	Resp. time	MB read	IOPS	MB/s
Seq 4MB	3,0215ms	2.048,0	330,96	1.323,85
4K	0,0938ms	520,6	10.661,92	41,65
4K QD4	0,0966ms	2.021,4	41.397,56	161,71
4K QD16	0,1104ms	7.074,5	144.884,36	565,95
32K	0,1912ms	2.454,2	5.230,02	163,44
128K	0,3406ms	5.510,6	2.936,06	367,01

Write	Resp. time	MB written	IOPS	MB/s
Seq 4MB	4,3359ms	1.024,0	230,63	922,52
4K	0,0227ms	640,0	43.966,71	171,74
4K QD4	0,0279ms	640,0	143.293,88	559,74
4K QD16	0,0711ms	640,0	225.156,73	879,52

Read score: 3.660,34 (3.660,34)
Total score: 9.587,62 (9.587,62)
Write score: 5.927,28 (5.927,28)

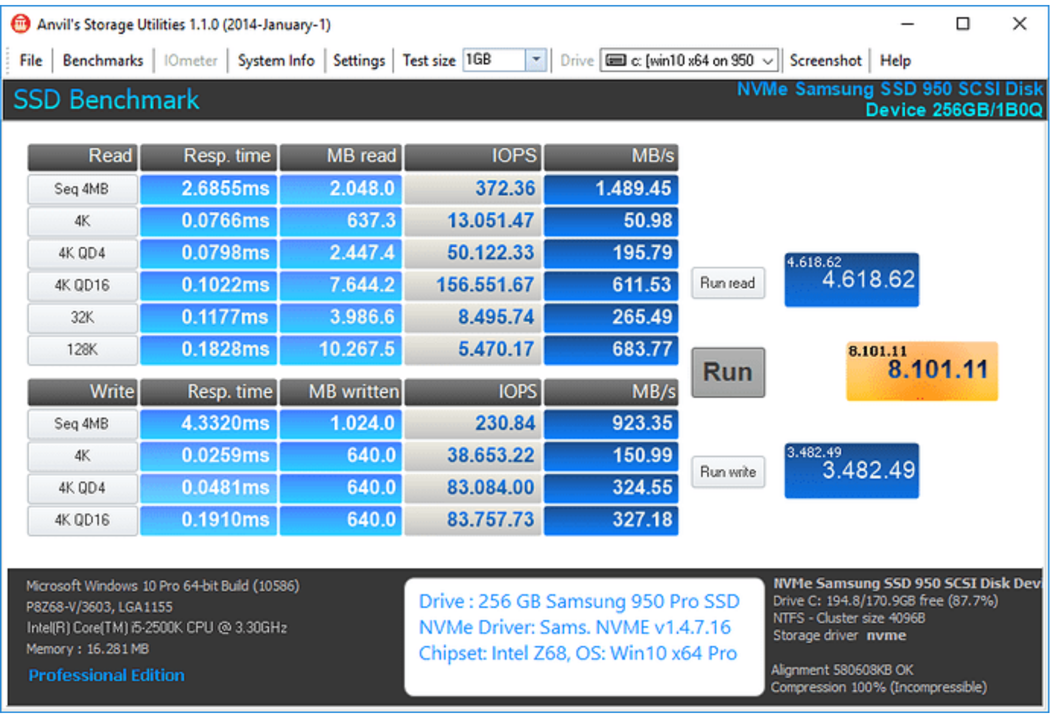
Microsoft Windows 10 Pro 64-Bit Build (10586)
P8Z68-V/3603, LGA1155
Intel(R) Core(TM) i5-2500K CPU @ 3.30GHz
Memory : 16.281 MB
Professional Edition

Drive : 400 GB Intel 750 NVMe SSD
NVMe Driver: Intel NVMe v1.3.0.1007
Chipset: Intel Z68, OS: Win10 x64 Pro

NVMe INTEL SSDPEDMW40 400GB/0174
Drive C: 243.6/223.1GB free (91.6%)
NTFS - Cluster size 4096B
Storage driver IaNVMe

Alignment 580608KB OK
Compression 100% (Incompressible)

256 GB Samsung 950 Pro M.2 SSD (connected via M.2>PCIe adapter card):



NVMe Samsung SSD 950 SCSI Disk Device 256GB/1B0Q

Read	Resp. time	MB read	IOPS	MB/s
Seq 4MB	2.6855ms	2.048.0	372.36	1.489.45
4K	0.0766ms	637.3	13.051.47	50.98
4K QD4	0.0798ms	2.447.4	50.122.33	195.79
4K QD16	0.1022ms	7.644.2	156.551.67	611.53
32K	0.1177ms	3.986.6	8.495.74	265.49
128K	0.1828ms	10.267.5	5.470.17	683.77

Write	Resp. time	MB written	IOPS	MB/s
Seq 4MB	4.3320ms	1.024.0	230.84	923.35
4K	0.0259ms	640.0	38.653.22	150.99
4K QD4	0.0481ms	640.0	83.084.00	324.55
4K QD16	0.1910ms	640.0	83.757.73	327.18

Read score: 4.618.62 (4.618.62)
Total score: 8.101.11 (8.101.11)
Write score: 3.482.49 (3.482.49)

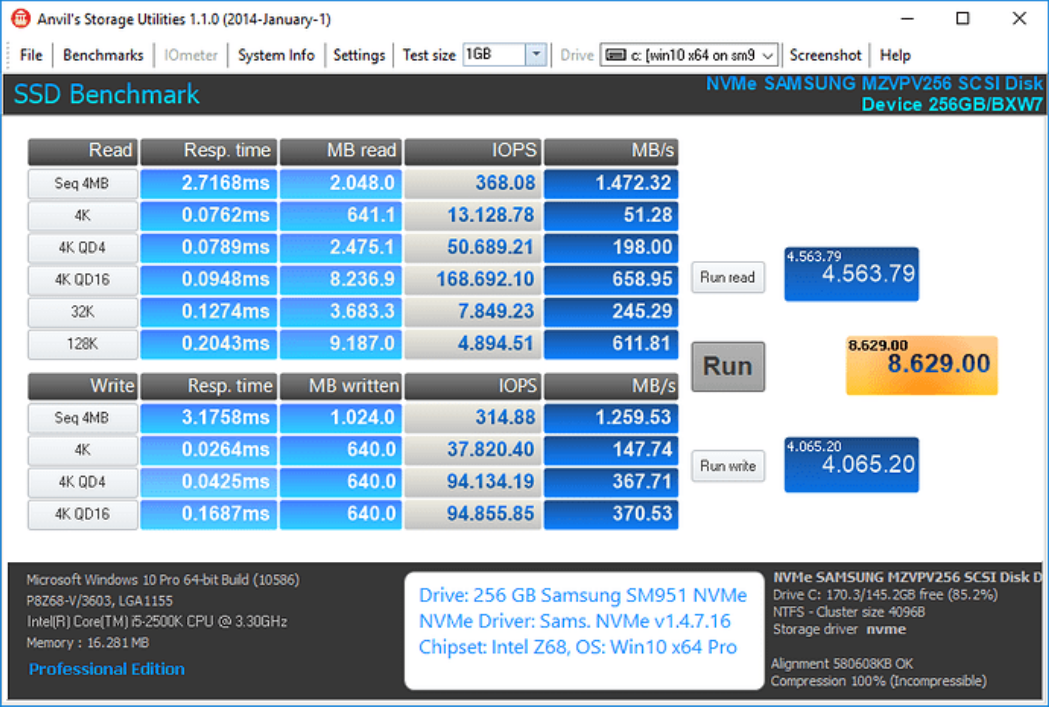
Microsoft Windows 10 Pro 64-bit Build (10586)
P8Z68-V/3603, LGA1155
Intel(R) Core(TM) i5-2500K CPU @ 3.30GHz
Memory : 16.281 MB
Professional Edition

Drive : 256 GB Samsung 950 Pro SSD
NVMe Driver: Sams. NVME v1.4.7.16
Chipset: Intel Z68, OS: Win10 x64 Pro

NVMe Samsung SSD 950 SCSI Disk Dev
Drive C: 194.8/170.9GB free (87.7%)
NTFS - Cluster size 4096B
Storage driver nvme

Alignment 580608KB OK
Compression 100% (Incompressible)

256 GB Samsung SM951 NVMe M.2 SSD (connected via M.2>PCIe adapter card):


Anvil's Storage Utilities 1.1.0 (2014-January-1)
File | Benchmarks | IOmeter | System Info | Settings | Test size 1GB | Drive c: [win10 x64 on sm9 | Screenshot | Help
SSD Benchmark
Run read
Run
Run write

NVMe SAMSUNG MZVPV256 SCSI Disk Device 256GB/BXW7

Read	Resp. time	MB read	IOPS	MB/s
Seq 4MB	2.7168ms	2.048.0	368.08	1.472.32
4K	0.0762ms	641.1	13.128.78	51.28
4K QD4	0.0789ms	2.475.1	50.689.21	198.00
4K QD16	0.0948ms	8.236.9	168.692.10	658.95
32K	0.1274ms	3.683.3	7.849.23	245.29
128K	0.2043ms	9.187.0	4.894.51	611.81

Write	Resp. time	MB written	IOPS	MB/s
Seq 4MB	3.1758ms	1.024.0	314.88	1.259.53
4K	0.0264ms	640.0	37.820.40	147.74
4K QD4	0.0425ms	640.0	94.134.19	367.71
4K QD16	0.1687ms	640.0	94.855.85	370.53

4.563.79 4.563.79

8.629.00 8.629.00

4.065.20 4.065.20

Microsoft Windows 10 Pro 64-bit Build (10586)
P8Z68-V/3603, LGA1155
Intel(R) Core(TM) i5-2500K CPU @ 3.30GHz
Memory : 16.281 MB
Professional Edition

Drive: 256 GB Samsung SM951 NVMe
NVMe Driver: Sams. NVMe v1.4.7.16
Chipset: Intel Z68, OS: Win10 x64 Pro

NVMe SAMSUNG MZVPV256 SCSI Disk D
Drive C: 170.3/145.2GB free (85.2%)
NTFS - Cluster size 4096B
Storage driver nvme

Alignment 580608KB OK
Compression 100% (Incompressible)

Any feedback is much appreciated!
If you are satisfied with my work/support, you can send a ***donation*** to my personal Win-RAID CA PayPal account by clicking onto **>this<** 78 link.

Good luck!
Dieter (alias Fernando)

created

Apr '15

last reply

3d

137k

views

1.3k

users

83

likes

50

links

Frequent Posters

Popular Links

There are **7704** replies with an estimated read time of **1661 minutes**.