# Obsolete BIOS Guide

First time to say thank you to users from The New Rebels Heaven Forum ( Polygon; Nightforce; Pinczakko; TwoBombs ) I will use part of their work for my guide

This Guide is specially made also for the great Win-Raid community which give some advice and impressions

Today the guide is maybe obsolete because of UEFI but let's try to get in. First point my personal intention was to activate or deactivate a feature called „Core Clumping ".

My reference hardware for this: 2x6328 AMD Opteron CPU | Supermicro H8DGi-F | Zotac 1080 Amp Extreme+ | lots of DIMMs | AMIBIOS8 (2MB)

After reaching free available PCIe Slots and no option to plug in an additional NVME Card I started to think about the Intel Bifurcation Feature to share Physical Slot with 2 or More Devices.

Tons of hours of reading Techdocs from AMDs SR5690 Chipset I get the conclusion AMD feature is called UnitID Clump ( counterpart of Intel's Bifurcation ). I get in touch with Supermicro Support and they told me it is not possible to change it with this Board ( Another H8 Series Board got the feature )
Another Supporter from Supermicro told me I need a Special PCIe Card from them - long story short: the special card was a hoax. The UnitID Clumping is working.

INFO: Just be sure you have no hiding options or menus in your AMIBIOS8
Also creating a Excel Sheet to write down everything is required!

Using Qalculate from Hanna which is one the best Calculate for this purpose

You need your BIOS; AmiBCP; MMTool; AmiSCE ( opt. TXTBCP ); an awesome Hex Editor with the ability to mark your selected HEX code with colors + position because you need it later.

Different types for Menus/JMPs/SUB Menus/Options/Texts Strings/User Rights/Structures ... that list may not completed or maybe not 100% clear:

01 __ __ - Within Setup structure Jump (followed by 2-Byte Value) to a Position from Entry Point
41 __ __ - Jump (followed by 2-Byte Value) to an Option from Entry Point
42 __ __ - Label Designation (followed by 2-Byte Value) for Basic Setup Structure
4300 - Separator-Blank Line
4301 - Separator-Single Line
4400 - Indicator for  Listing Options of a Menu
4401 - Indicator for Header/Into Menu Title
4500 ____ ____ - No Refresh > 2-Byte Text String > 2-Byte Function
4501 ____ ____ - Refresh > 2-Byte Text String > 2-Byte Function

01 [up to 16 Option] - Outside Setup Structure are targeting Options

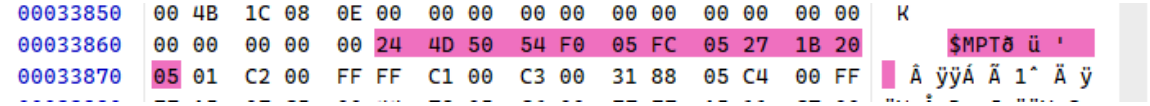
Because you maybe think that it makes no sense because in AMIBCP and in hex editor are differences you need to know that a Setup or Menu String which in Include the Jump Instructions to the Options is also the Title for the Sub Menu

where you find the Options. It is a big puzzle but when you do some calculations it is easy to understand.

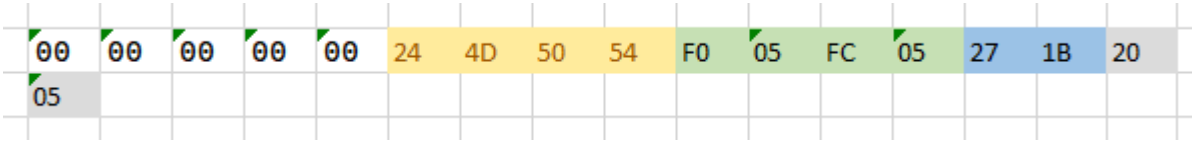
## Starting:

Extract with MMTool the 1B ( Single Link Arch BIOS ) Module uncompressed.

Open it with your HEX Editor and Search fort he String $MPT from the $-Letter mark a length of 0xCh.

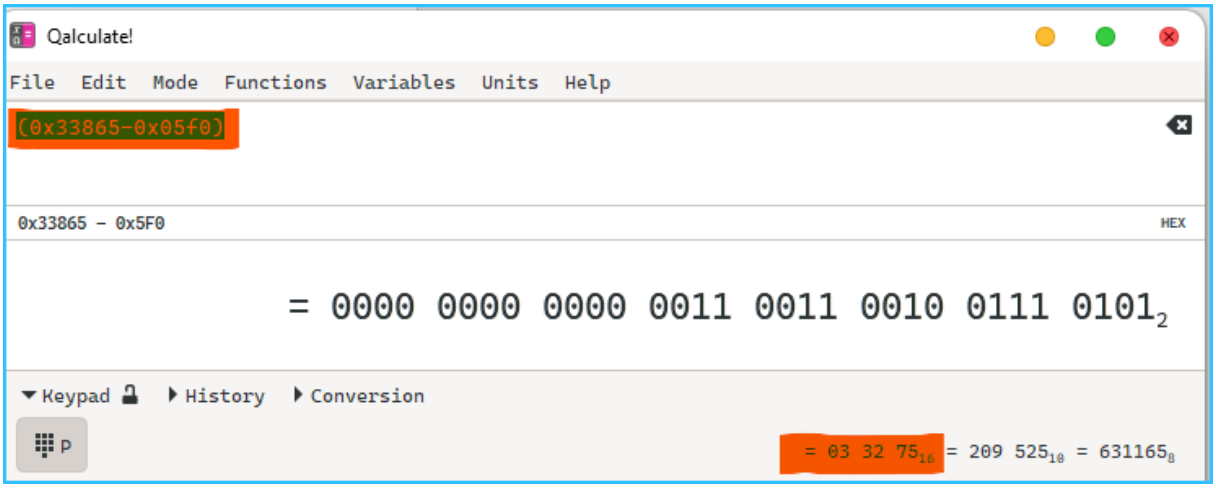



The length of „$MPT“ is for 4 bytes and the Entry Point for everything that follows is 0x33865 which is called offset.



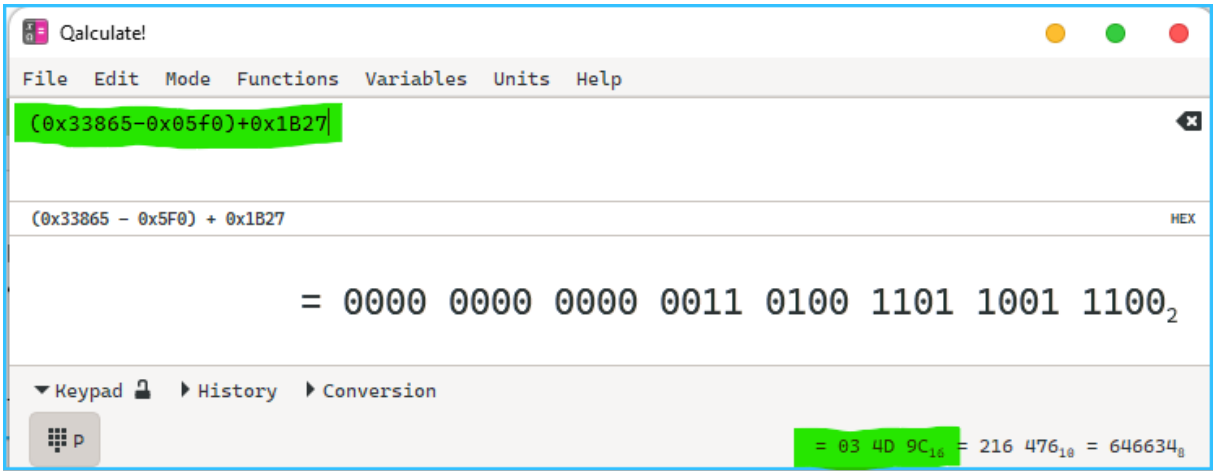

ok above you see in yellow $MPT the next important bytes are the left 2 bytes of the green marked. you need to rotate it and get the value 05F0.

To get the Basic Offset Address for all Options and Menu Strings is:




Resulting Offset: 0x033275

Now focusing the blue bytes rotate it and you will get the value 1B27.
The calculation to get the first BIOS Setup Menu Entry String:




Resulting Offset: 0x034D9C ... move to the Offset:

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00034D80 | 86 | 12 | 86 | 20 | 86 | A1 | 86 | 02 | 2A | 36 | 03 | 85 | 02 | 36 | 03 | 21 |
| 00034D90 | 63 | 98 | 53 | B1 | 86 | 02 | A5 | 6C | 65 | 1A | 80 | 00 | 42 | 87 | 02 | 01 |
| 00034DA0 | 50 | 1B | 02 | 00 | 42 | 88 | 02 | 01 | 97 | 1B | 02 | 00 | 42 | 89 | 02 | 01 |
| 00034DB0 | 2A | 24 | 01 | 00 | 42 | 8A | 02 | 01 | 60 | 24 | 01 | 00 | 42 | 8B | 02 | 01 |
| 00034DC0 | 70 | 25 | 01 | 00 | 00 | 44 | 01 | 5D | 03 | 44 | 00 | 2C | 03 | 43 | 01 | 41 |

Congratulation you've found the Main Setup Entries at this position ( 0x034D9C ) ... in the above screen I've marked all Entries.

Important: The Main Setup Entries have a length of 8 bytes

Analyzing:

42 – Label Designation (Text String) followed by
8702 – rotated 0287
0287 – MAIN
01 – JMP Instruction followed by
501B – rotated 1B50
02 – Misc. Byte
00 – ENDPOINT

next one:

42 – Label Designation (Text String) followed by
8802 – rotated 0288
0288 – ADVANCED
01 – JMP Instruction followed by
971B – rotated 1B97
02 – Misc. Byte
00 – ENDPOINT

.
.
.

last one:

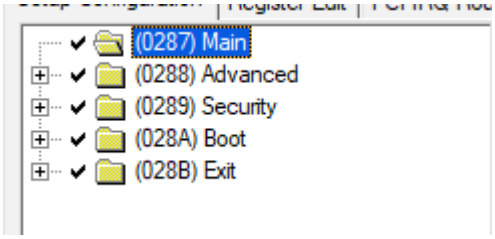
42 – Label Designation (Text String) followed by
8B02 – rotated 028B
028B – EXIT
01 – JMP Instruction followed by
7025 – rotated 2570
01 – Misc. Byte
00 – ENDPOINT

Overview (HEX):

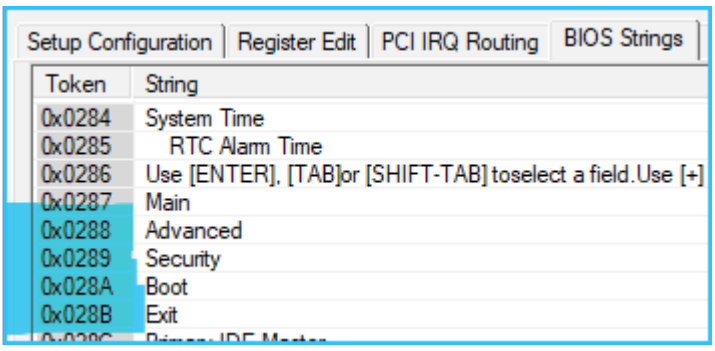
# Main BIOS MENU ENTRY

| HANDLE | OFFSET | 1B | Label Designation | String Handle(r) BCP | | JMP | TOKEN CMOS / BIOS-RAM (Little Endian) | | Show Menu + UAR | END |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0287 | 34D9Ch | Menu – Main | 42 | 87 | 02 | 01 | 50 | 1B | 02 | 00 |
| 0x0288 | 34DA4h | Menu – Advanced | 42 | 88 | 02 | 01 | 97 | 1B | 02 | 00 |
| 0x0289 | 34DACh | Menu – Security | 42 | 89 | 02 | 01 | 2A | 24 | 01 | 00 |
| 0x028A | 34DB4h | Menu – Boot | 42 | 8A | 02 | 01 | 60 | 24 | 01 | 00 |
| 0x028B | 34DBCh | Menu – Exit | 42 | 8B | 02 | 01 | 70 | 25 | 01 | 00 |

Okay Overview un AMIBCP:




You see it is no magic, you see again the String Handlers from Module 20:

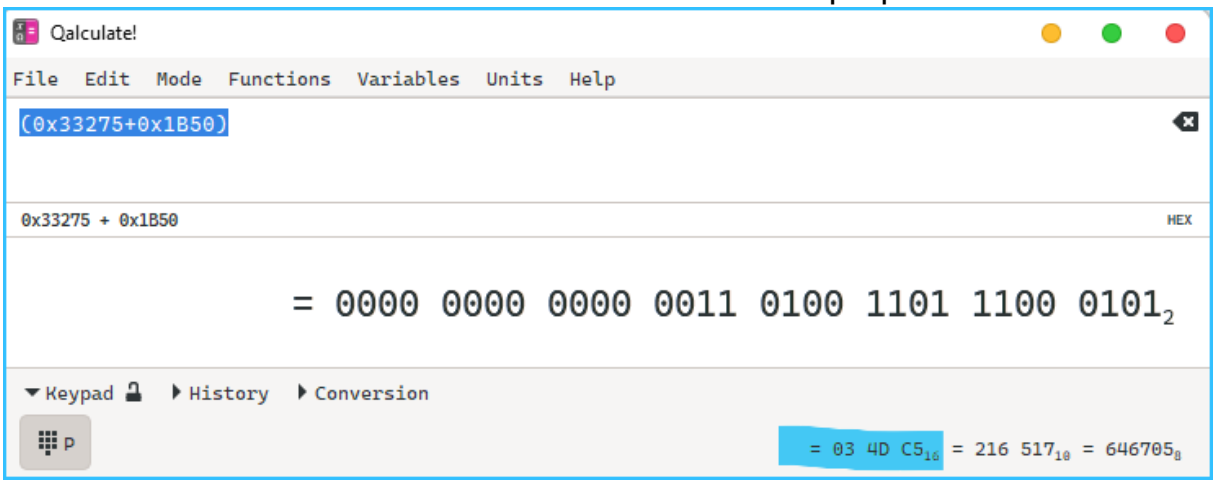



Now we get the Menu Structure which when ENTER is pressed open up the Sub Menu.
But how this works? As the 42 works as a Label Designation you will find the Jump instruction in the same 8-Byte String.
JMP Instructions are 01 and 41 ( explanation of 41 later )
Example for „MAIN“ 01 > JMP to followed by the Address 501B.

```
00034D90  63 98  53 B1  86 02  A5 6C  65 1A  80 00  42 87  02 01
00034DA0  50 1B  02 00  42 88  02 01  97 1B  02 00  42 89  02 01
```

This value tells the BIOS or BIOS Tools where it found the Sub Menu entries for MAIN. You need to rotate it and get 1B50

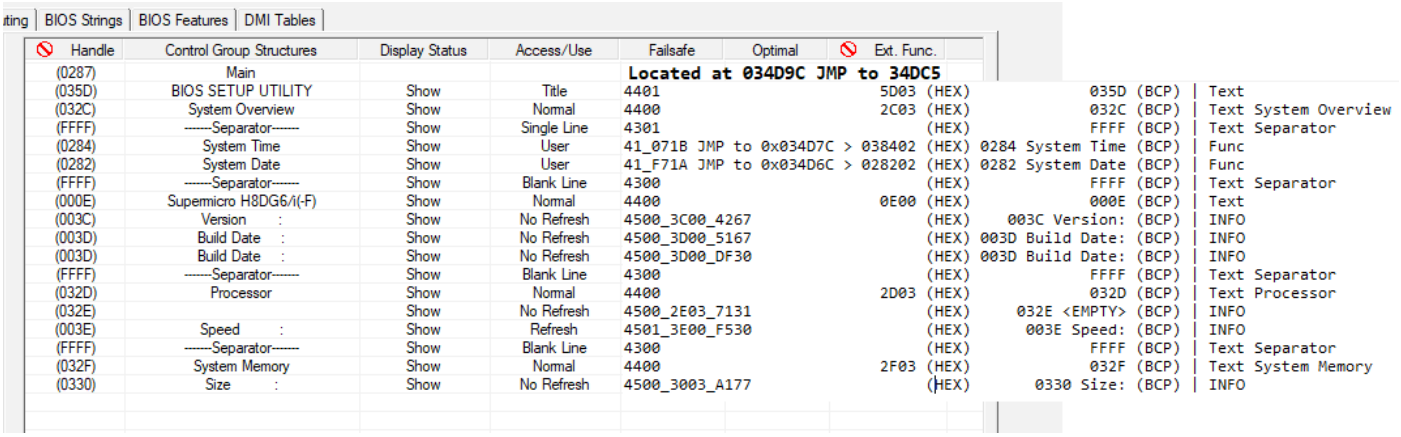
Now we're calculate the Offset for the Main Setup Options:




Resulting Offset: 0x034DC5 ... move to that Offset:

```
00034D90  63 98  53 B1  86 02  A5 6C  65 1A  80 00  42 87  02 01
00034DA0  50 1B  02 00  42 88  02 01  97 1B  02 00  42 89  02 01
00034DB0  2A 24  01 00  42 8A  02 01  60 24  01 00  42 8B  02 01
00034DC0  70 25  01 00  00 44  01 5D  03 44  00 2C  03 43  01 41
00034DD0  07 1B  41 F7  1A 43  00 44  00 0E  00 45  00 3C  00 42
```

I thought about how to demonstrate it remember that you need to rotate the Address that it make sens:




Okay now we have some basics about how the AMIBIOS8 is working and this is no new stuff because the guys from the new rebel heaven forum figured this out. At the beginning I mentioned that I will show a way to get all compiled BIOS options include the ones who are not written into the setup that's the second part of this guide.

## Foreword:

I've bought a CH341 Programmer with SOIC 8 and SOIC 16 Pin cable clamp because if you make a mistake you can reflash a working state of the modded BIOS.
Also I'm not 100% sure if in some cases the BIOS Settings Mod have negative effects with the recovery procedure ( AMI.ROM; BIOS.ROM; SUPER.ROM )
I've watched the Bootcode process with the IPMI Software and saw sometimes that the recovery procedure is hanging ( not waiting for the ROM )

At this working state It is not possible to add menu's to the BIOS because the entire 1B Module is continuous filled with code, there is no left empty space with FF or 00.

What is possible to overwrite used space i.e. from not used IDE detection but it is really difficult and you take into account the recovery part.

You have 1 Setup part with your settings, one with failsafe and one with optimized settings that is important to know because if you change one settings entry you need to search forward to the next one with the same bytecode If you don't do this and you enter a menu with that settings the bios breaks immediately.

False myth: if a setting is not visible in menu it is not possible to use it – that not true

## Let's Go!

### Case #1

If you're on the machine which is you want to Mod

Open a CMD in your folder AMISCEV2.20.and type

amiscew.exe /oc cmos.txt

You need /oc because we need the exact CMOS handlers to keep our work on

### Case #2

There was also a possibility to use a program to get it if the machine is offline but I don't remember maybe someone knew it?

Open up the  cmos.txt and you got every settings which are saved to CMOS/NVRAM also the Settings without Entry. You can, If don't like to mod or dislike this guide, keep your work into this file. For example I will using the not configurable Setting called „MPS Revision" search for it ( remember using H8DGI BIOS File )

```
Setup Question = MPS Revision
// CMOS Reg.    = EF                    // Do NOT change this line
// CMOS Mask    = 20                    // Do NOT change this line
Token           = 177D                  // Do NOT change this line
BIOS Default    = [01]1.4
MFG Default     = [01]1.4
Options         = [00]1.1               // Move "*" to the desired Option
                  *[01]1.4
```

Yellow mark is the Function Text String
Blue mark is the CMOS token – the most important value
Then the orange marked is the actually set option into NVRAM/CMOS as you see after the „//“ when you move the „*“ to the [00]1.1 you will change the MPS Revision to 1.1.
Save the file and update the CMOS/NVRAM with  amiscew.exe /I cmos.txt
You’re system will freeze but after reset the setting is accepted and written into CMOS/NVRAM

Now the professional way:

look for the settings you want to get into your BIOS write them down in your excel sheet

Lets do an example for the Setting Up the GPP Cores of the H8DGI because I want to use UnitID Clumping and Supermicro said it is not possible with that board. Reading the SR5690 Guide we know we need the following Settings:

Setup Question = Core Configuration

Token = 263A

Setup Question = Core Configuration

Token = 263C

Setup Question = Core Configuration

Token = 3583

Setup Question = UnitID Clumping

Token = 35D8

Second part in AMIBCP > BIOS Strings search for the Setup Question’s after the „= “ in H8DGI you will get the following results:

Setup Question = Core Configuration

Token = 263A > Rotated: 3A26
Text = 019A > Rotated: 9A01

Setup Question = Core Configuration

Token = 263C > Rotated: 3C26
Text = 019A > Rotated: 9A01

Setup Question = Core Configuration

Token = 3583 > Rotated: 8335
Text = 019A > Rotated: 9A01

Setup Question = UnitID Clumping

Token = 35D8 > Rotated: D835
Text = 016C > Rotated: 6C01

## Fulfill the puzzle:

At the beginning we've learned that the 01-indicator outside the setup structure is an entry point for every changeable Option.
01 followed by Text and Token Results in:

```
01 9A 01 3A 26 [....] Core Configuration ( GPP1 )
01 9A 01 3C 26 [....] Core Configuration ( GPP2 )
01 9A 01 83 35 [....] Core Configuration ( GPP3a/b )
01 6C 01 D8 35 [....] UnitID Clumping
```

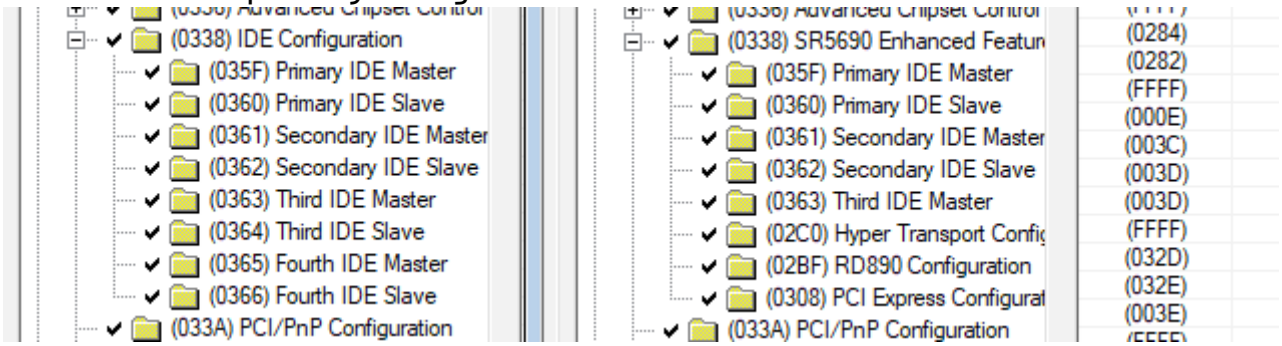
Now in your HEX Editor with opened 1B Module search for Hex Code above you will find exact 1 search result for each line.
I don't describe the entire Setting string because for this guide it is not important.

Well try this method with your target Settings in your BIOS and it will work.

Ok we have a method to get the settings, change it in NVRAM/CMOS – and also find them in your 1B Module now and the last step is to get them into your BIOS Setup

## BIOS Setup:

Because it is hardly to add new Entries, we will edit useless settings. I have completely changed the obsolete IDE Detection Menus:




BEFORE AFTER

I will don't describe the complete way to make it looking nice so let's focus on the 4 settings above.
Look in AMIBCP in the IDE Configuration it will look like this:

| Handle | Control Group Structures | Display Status | Access/Use | Failsafe | Optimal | Ext. Func. |
|---|---|---|---|---|---|---|
| (0338) | IDE Configuration | | | | | |
| (035D) | BIOS SETUP UTILITY | Show | Title | | | |
| (035E) | IDE Configuration | Show | Normal | | | |
| (FFFF) | -------Separator------- | Show | Single Line | | | |
| (011B) | OnBoard PCI IDE Controller | Show | User | Enabled | Enabled | No |
| (022D) | OnChip SATA Channel | Show | User | Enabled | Enabled | No |
| (0170) | OnChip SATA Type | Show | User | Native IDE | Native IDE | No |
| (0260) | Raid CodeBase | Show | User | 01 | 01 | Yes |
| (022E) | SATA IDE Combined Mode | Show | User | Enabled | Enabled | Yes |
| (022F) | PATA Channel Config | Show | User | SATA as pri... | SATA as pri... | No |
| (FFFF) | -------Separator------- | Show | Blank Line | | | |
| (FFFF) | -------Separator------- | Show | Blank Line | | | |
| (011E) | IDE Detect Time Out (Sec) | Show | User | 35 | 35 | No |

Because I decided we remap the Settings OnBoard PCI IDE Controller; OnChip SATA Channel; OnChip SATA Type and SATA IDE Combined Mode. Change this 4 settings how you need them ( careful: Failsafe and Optimal ) save the BIOS and extract the 1B

module uncompressed again. Using the Method to get the Options with Text+Token results in (already rotated):

```
Setup Question = OnChip SATA Type

Token          = DB35
Text           = 7001
Code           = 017001
Offset         = 0x3408E

Setup Question = OnChip SATA Channel

Token          = 4317
Text           = 2D02
Code           = 012D02
Offset         = 0x34A99

Setup Question = SATA IDE Combined Mode

Token          = 4417
Text           = 2E02
Code           = 012E02
Offset         = 0x34AA5

Setup Question = OnBoard PCI IDE Controller

Token          = 3D33
Text           = 1B01
Code           = 011B01
Offset         = 0x33D27
```

Search for each Code and with 01 marked write down the Offset of it. It looks obvious but it wasn't. Remember the magic Offset we figured out at the beginning: 0x33275 now we got the second Offset's marked blue above. The calculation we need now:


Do it for all 4 Offsets! Also calculate the 4 needed green values for Core Configuration and UnitID Clumping.

This results in:

```
Setup Question = OnChip SATA Type
Offset         = 0x3408E
Value (rotated)= 190E

Setup Question = OnChip SATA Channel
Offset         = 0x34A99
Value (rotated)= 2418

Setup Question = SATA IDE Combined Mode
Offset         = 0x34AA5
Value (rotated)= 3018

Setup Question = OnBoard PCI IDE Controller
Offset         = 0x33D27
Value (rotated)= B20A

Setup Question = Core Configuration
Offset         = 0x3429C
Value (rotated)= 2710

Setup Question = Core Configuration
Offset         = 0x342AA
Value (rotated)= 3510

Setup Question = Core Configuration
Offset         = 0x33EC6
Value (rotated)= 510C

Setup Question = UnitID Clumping
Offset         = 0x3407C
Value (rotated)= 070E
```

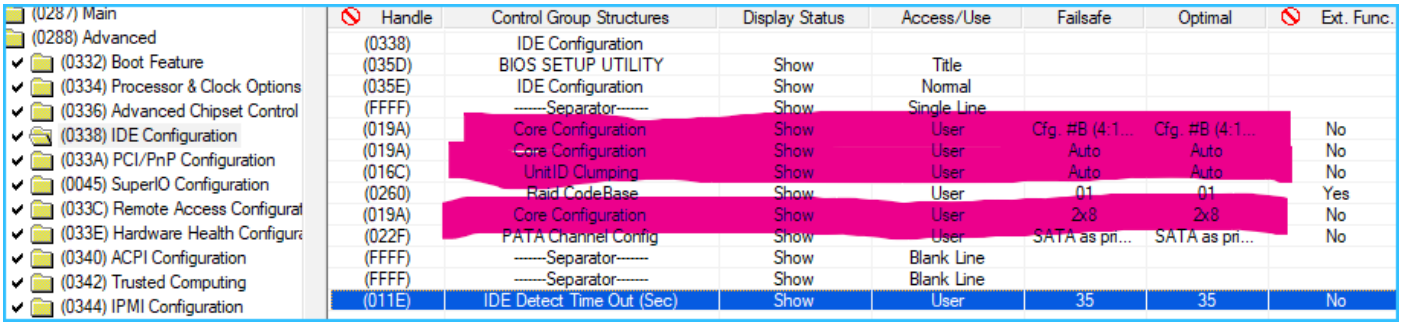
Remember that 41 indicates a function Jump to changeable BIOS option put 41 at the beginning of the calculated and rotated values above, which I now call „Setup Item“:

```
Setup Question = OnChip SATA Type
Setup Item = 41190E

Setup Question = OnChip SATA Channel
Setup Item = 412418

Setup Question = SATA IDE Combined Mode
Setup Item = 413018

Setup Question = OnBoard PCI IDE Controller
Setup Item = 41B20A

Setup Question = Core Configuration
Setup Item = 412710

Setup Question = Core Configuration
Setup Item = 413510

Setup Question = Core Configuration
Setup Item = 41510C

Setup Question = UnitID Clumping
Setup Item = 41070E
```

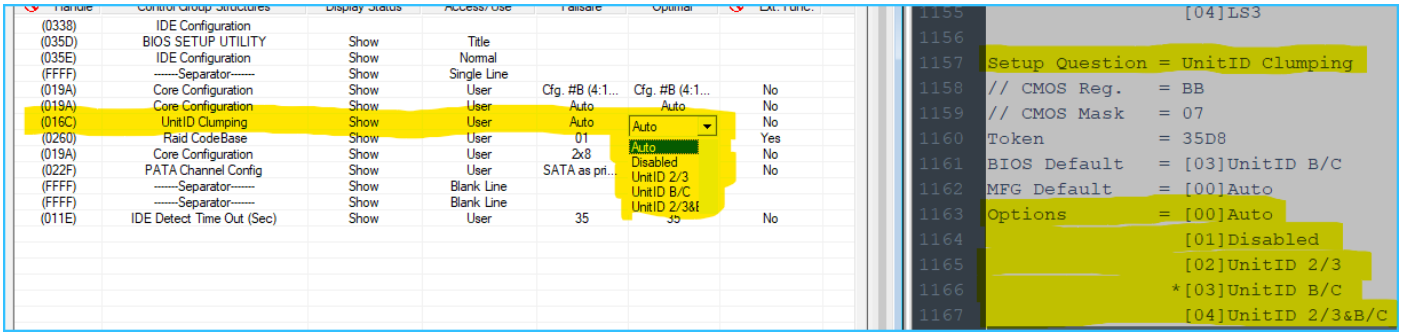
Give it a try and search for the 3-Byte length value in your HEX Editor with opened 1B Module ... you will get results for the first 4 Options which are also

shown in AMIBCP as Visible Settings. When you search for the second 4 Options you will get no results. As I said you have different parts in the Setup this means you will be found for the first 4 options respectively 2 results

Now use Find & Replace Function in your Hex Editor and change OnChip SATA Type (41190E) to UnitID Clumping (41070E), Change all founded entries to it. Do the same Find & Replace method for the other 3 Options.
Save 1B Module file open the BIOS in MMTool and select 1B Module within. Move to the tab „replace“ open the file dialog and select your modified 1B file. replace it uncompressed, save the BIOS.ROM file and reopen it in AMIBCP and checkout the successfully replaced settings which are now visible:




Maybe you already knows it but if not here some explanation. I told you to setup the option first you want to remove because you couldn’t access them from now on in your setup. But they are still in your BIOS at the Offset position you’ve figured out earlier. You have only changed the internal routing from the setup menu structure. Please check and make a comparison if the new 4 options have all included setup possibility like it shown in your cmos.txt:




If you see empty white spaces or lines in AMIBCP you’ve made some mistakes or misconfigured settings. Don’t use it, don’t flash that BIOS!

If you made a harder crap mistake AMIBCP failed to load and hang-up ( end it with ALT+F4 or with the Task Manager.

## Conclusion:

This method is really easier and minimize the factor of misconfigurations. If you understand how the entire menu structure works and start to search for the 41-Jump Executions look before or after it to check other option within the menu, or retrieve the Menu Title and Header; a separator, or a INFO ( 4501/4500 ).

If you are now motivated and maybe impressed you can use the cmos.txt and focusing the CMOS Reg. and CMOS Mask values. I get a calculation from the Rebels Heaven Forum to retrieve them directly in your running CMOS/NVRAM. But that is another Part and not needed for now.