# Hardware database format reference v1.2

# 1. Hardware database concepts

MSI Afterburner uses hardware databases concepts to implement voltage control support on a wide range of graphics cards and at the same time keep extremely fast and efficient application startup performance. Database concepts mean that each voltage control capable graphics card supported by MSI Afterburner is being uniquely identified inside the application by personal entry in hardware database. The database entry specific to each card define exact model of voltage controller, define controller's unique hardware location address and provide full set of additional calibration information specific to the voltage controller model. So MSI Afterburner doesn't need to perform full hardware scanning, it doesn't need to perform heuristic detection of all possible controllers to select proper voltage control codepath. Such approach gives software voltage control layer the maximum safety, reduces the risk of incorrect detection of voltage controller model to zero and gives MSI Afterburner huge advantage in application startup performance comparing to tools performing full hardware scan at each startup. However, database approach doesn't allow the application to control voltage on unknown cards with non-reference voltage control circuit.

We ship MSI Afterburner with hardware databases providing voltage control support on 100% reference design AMD and NVIDIA graphics cards and on custom design MSI cards labeled with ***"Voltage control"*** logo. However, rich set of voltage controllers supported by MSI Afterburner core allows implementing voltage control support on many custom design cards from third party hardware vendors. For example, custom design ASUS GTX 980 STRIX or EVGA GTX 780 Classified series cards use the same voltage controllers as MSI Lightning series graphics cards, so it is possible to add such custom design ASUS and EVGA to the database and unlock voltage control for it. We don't include such cards into official databases distributed with application, however we introduce original concepts of third party hardware database, which can be downloaded separately and attached to our application to extend voltage control functionality on third party custom design graphics cards. Third party hardware database format is open, so enthusiasts can add new cards to it and share the result with the community. Please follow this guide if you are going to edit third party hardware database.

# 2. Hardware database format

Third party hardware database is ASCII text file defining independent voltage control scenarios for different models of graphics cards. The database must have "***;OEM***" signature in the very beginning, otherwise the application won't start and display error telling you that some components are missing or corrupted. The database format is derived from standard INI file format, so it is composed of multiple sections containing different fields. Each section inside the database identifies unique graphics card model or the set of similar models, if wildcards are used in the section name. Graphics card section names have the following format:

***[VEN_XXXX&DEV_XXXX&SUBSYS_XXXXXXXX&REV_XX]***

where ***VEN_XXXX*** part is graphics card vendor ID, ***DEV_XXXX*** part is graphics card device ID, ***SUBSYS_XXXXXXXX*** part is graphics card subsystem ID and ***REV_XX*** part is graphics card revision number.

Vendor ID includes 4 hexadecimal digits, e.g. ***VEN_10DE*** for ***NVIDIA*** graphics card. Device ID includes 4 hexadecimal digits, e.g. ***DEV_1004*** for ***NVIDIA GeForce GTX 780***. Subsystem ID includes 8 hexadecimal digits and defines vendor specific sub-model name, e.g. ***SUBSYS_17883842*** for ***EVGA GTX 780 Classified***. Revision number includes 2 hexadecimal digits, however different revisions of the same model normally use the same voltage controller, so it is recommended to avoid specifying revision directly in the section name and use wildcard symbol ***"?"*** instead to define one section for all possible revisions, e.g. ***REV_??.*** Wildcard symbol can be also used to define one section for a set of graphics cards models if necessary, for example ***[VEN_10DE&DEV_1004&SUBSYS_????3842&REV_??]*** section define settings for all possible sub-models of ***EVGA GTX 780*** series graphics cards. Use wildcards with caution and only when it is really necessary, for example ***[VEN_10DE&DEV_1004&SUBSYS_????????&REV_??]*** section define settings for all ***NVIDIA GeForce GTX 780*** series cards, so specifying such section inside third party database will effectively redefine MSI Afterburner's internal database settings for reference design ***NVIDIA GeForce GTX 780*** series graphics cards.

You may easily extract section name for your graphics card by clicking ***<i>*** button in MSI Afterburner, each GPU GUID displayed in MSI Afterburner information window is exactly what you are searching for.

## 2.1. [VEN_XXXX&DEV_XXXX&SUBSYS_XXXXXXXX&REV_XX] section format

This section defines settings for the graphics card model uniquely identified by section name or the set of graphics card models if the section name includes wildcard symbols. The settings include optional graphics card model name, voltage controller model, controller hardware location address and other helper settings specific to voltage controller model. For example:

```
[VEN_1002&DEV_6798&SUBSYS_99991043&REV_??]

Desc                                  = ASUS ARES II
VDDC_CHL8228_Detection                = 6:30h
VDDC_CHL8228_Defaults                 = C6 9F
VDDC_CHL8228_VIDReadback              = 1
MVDDC_CHL8228_Detection               = 6:30h
MVDDC_CHL8228_Defaults                = CD BF
MVDDC_CHL8228_VIDReadback             = 1
```

***Desc*** field is optional and it can be specified to display custom name for a graphics card in MSI Afterburner GUI, e.g. ***ASUS ARES II*** in this example. If this field is not specified, MSI Afterburner will display graphics card name as it is reported by Windows. Please keep in mind that graphics card name is displayed in limited space in some MSI Afterburner skins, so try to keep it as compact as it is possible and always test the result.

***VDDC_CHL8228_Detection*** field defines the first voltage controller detection information. Controller detection information includes voltage controller target, model and hardware location address. Voltage controller target in this example is ***VDDC***, which means that we're defining settings for ***core*** voltage controller. Voltage controller model is ***CHL8228***, and voltage controller location is I2C bus **6**, device address **30h**.

***VDDC_CHL8228_Defaults*** field defines ***CHL8228*** controller specific settings for core voltage controller, in this example we're telling MSI Afterburner that core voltage is adjusted by ***CHL8228*** voltage control loop 1 VID LUT register ***C6*** and default VID for this register is equal to ***9F***.

***VDDC_CHL8228_VIDReadback*** field defines additional ***CHL8228*** controller specific settings for core voltage controller, we're telling MSI Afterburner that we want to see VID readback (i.e. programmed target voltage) on core voltage monitoring graph instead of real voltage readback.

***MVDDC_CHL8228_Detection*** field defines the second voltage controller detection information. Controller detection information includes voltage controller target, model and hardware location address. Voltage controller target in this example is ***MVDDC***, which means that we're defining settings for ***memory*** voltage controller. Voltage controller model is ***CHL8228***, and voltage controller location is I2C bus **6**, device address **30h**.

***MVDDC_CHL8228_Defaults*** field defines ***CHL8228*** controller specific settings for memory voltage controller, in this example we're telling MSI Afterburner that memory voltage is adjusted by ***CHL8228*** voltage control loop 2 VID LUT register ***CD*** and default VID for this register is equal to ***BF***.

***MVDDC_CHL8228_VIDReadback*** field defines additional ***CHL8228*** controller specific settings for core voltage controller, we're telling MSI Afterburner that we want to see VID readback (i.e. programmed target voltage) on memory voltage monitoring graph instead of real voltage readback.

Voltage controller detection fields are the only unified fields for all models of controllers. Detection fields have the following format:

***<target>_<model>_Detection = [<i2c_bus_filter>:]<i2c_device_filter>***

where ***<target>*** is voltage controller target, which can be set to ***VDDC*** (for core voltage controller detection), ***MVDDC*** (for memory voltage controller detection), ***VDDCI*** or ***PEXVDD*** (for auxiliary voltage controller detection). ***VDDCI*** and ***PEXVDD*** target definitions are functionally equal and both map the voltage controller to ***Aux voltage*** slider in MSI Afterburner user interface, different target names are used just to improve internal database readability (***VDDCI*** on AMD cards, ***PEXVDD*** on NVIDIA graphics cards). <***model>*** is a model of voltage controller, which can be set to ***CHL8214***, ***CHL8228***, ***CHL8266***, ***CHL8318***, ***IR3567B***, ***IR3595A***, ***L6788A***, ***NCP4206***, ***NCP81022***, ***UP1637***, ***UP6204***, ***UP6208***, ***UP6218***, ***UP6262***, ***UP6266***, ***VT1165***, ***VT1556*** or ***Generic***. ***<i2c_bus_filter>*** specify the range of I2C buses where the controller can reside, for example it can be set to ***"3"*** if the controller can be located on I2C bus 3 only, ***"3-5"*** if it can be in bus 3 to 5 range or ***"3,5"*** if it can be on bus 3 or 5. ***<i2c_device_filter>*** specify the range of 7-bit I2C device addresses where the controller can reside. Similar to bus filter, you can specify exact device address, range of addresses or a few comma separated addresses. If you don't specify bus filter and use device filter only, MSI Afterburner will search for the controller on all I2C buses available on this graphics card until matching I2C device is found. In most cases it is safe to specify device filter only, however, some controllers (e.g. ***UP6262***) have no internal identification register and reside in the same address space as monitor DDC. So in this case it is absolutely necessary to specify both bus and device filters in order to detect the controller properly.

It is allowed to specify a few voltage controller detection fields for the same voltage controller target (e.g. for ***VDDC***) with different voltage controller models inside one section if some graphics card model can use different models of voltage controllers. For example reference design ***ATI RADEON 4870*** series cards can use either ***VT1165*** or ***L6788A*** to control core voltage. So database section for this graphics card contains both ***VDDC_VT1165_Detection*** and ***VDDC_L6788A_Defaults*** fields.

The rest voltage controller related fields, such as ***VDDC_CHL8228_Defaults*** or ***VDDC_CHL8228_VIDReadback*** mentioned in this example are specific to model of voltage controller and will be discussed further in controller model specific chapters.

### 2.1.1. CHL8214 voltage controller

CHL8214 is a dual-loop controller, which can control up to two voltage outputs independently. CHL8214 supports internal lookup table (LUT) of voltages for all possible GPU power states, which means that it allows controlling 3D performance state voltage independently of the rest performance states. CHL8214 supports output voltage monitoring in VID readback form, which means that it can report target programmed voltage only.

This controller can reside in ***full 7-bit I2C device address range***. The controller is identified by register ***9Ch*** (must be equal to ***43h***) and register ***8Ch*** (must be equal to ***16h***). Model specific settings for CHL8214 include the following:

***<target>_CHL8214_Defaults = <vid_lut_register_address> <default_vid>***

where ***<vid_lut_register_address>*** is the address of VID register controlling voltage for 3D performance state, the address can be in ***43h - 4Ah*** range for the first voltage loop and in ***4Bh - 4Eh*** range for the second voltage loop.

***<default_vid>*** specify default value for previously defined VID register.

*Implementation note: This controller can be invisible in standard I2C dump, if it is located in upper I2C address space. In this case upper I2C address space need to be scanned additionally. Refer to chapter 4 to get more detailed info.*

### 2.1.2. CHL8228 voltage controller

CHL8228 is dual-loop controller, which can control up to two voltage outputs independently. CHL8228 supports internal lookup table (LUT) of voltages for all possible GPU power states, which means that it allows controlling 3D performance state voltage independently of the rest performance states. CHL8228 supports output voltage monitoring in VID readback and real monitoring forms, which means that it can report both target programmed and real monitored voltages. ***CHL8228*** codepath can be also use for compatible ***CHL8225A*** and ***CHL8225B*** controllers.
This controller can reside in ***full 7-bit I2C device address range***. The controller is identified by register ***8Ch*** (can be equal to ***03h*** for ***CHL8225A***, ***04h*** for ***CHL8225B*** or ***0Eh*** for CHL8228). Model specific settings for CHL8228 include the following:

***<target>_CHL8228_Defaults = <vid_lut_register_address> <default_vid>***

where ***<vid_lut_register_address>*** is the address of VID register controlling voltage for 3D performance state, the address can be in ***C6h - C9h*** range for the first voltage loop and in ***CAh - CDh*** range for the second voltage loop.
***<default_vid>*** specify default value for previously defined VID register.

***<target>_CHL8228_VIDReadback = <readback_mode>***

where ***<readback_mode>*** is controlling VID readback mode. When it is set to 1 VID readback mode is enabled and the voltage monitored by this controller is a target voltage programmed by VID. When it is set to 0 VID readback mode is disabled and real voltage is being monitored by this controller.

*Implementation note: This controller can be invisible in standard I2C dump, if it is located in upper I2C address space. In this case upper I2C address space need to be scanned additionally. Refer to chapter 4 to get more detailed info.*

### 2.1.3. CHL8266 voltage controller

CHL8266 is single-loop controller, which can control only one voltage output. CHL8266 supports voltage control in output voltage override form only. CHL8266 supports output voltage monitoring in real form, which means that it can report real monitored voltage only.
This controller can reside in fixed ***46h*** device address only. The controller has no identification registers and no model specific settings.

### 2.1.4. CHL8318 voltage controller

CHL8318 is single-loop controller, which can control only one voltage output. CHL8318 supports voltage control in output voltage override and offset forms. CHL8318 supports output voltage monitoring in VID readback form, which means that it can report target programmed voltage only.
This controller can reside in fixed ***42h***, ***44h***, ***46h***, ***70h***, ***72h***, ***74h*** or ***76h*** device addresses only. The controller has no identification registers. Model specific settings for CHL8318 include the following:

***<target>_CHL8318_Type = <control_type>***

where ***<control_type>*** can be set to 0 for fixed output voltage override or 1 for applying offset to output voltage.

*Implementation note: This controller can be invisible in standard I2C dump, if it is located in upper I2C address space. In this case upper I2C address space need to be scanned additionally. Refer to chapter 4 to get more detailed info.*

### 2.1.5. IR3567B voltage controller

IR3567B is dual-loop controller, which can control up to two voltage outputs independently. IR3567B supports voltage control in output voltage override and offset forms. IR3567B supports output voltage monitoring in VID readback form, which means that it can report target programmed voltage only. ***IR3567B*** codepath can be also used for compatible ***IR356x*** and ***IR3570*** family controllers.
This controller can reside in ***28h - 46h*** device address range only. The controller is identified by register ***92h*** (must be equal to ***43h***). Model specific settings for IR3567B include the following:

***<target>_IR3567B_Type = <control_type>***

where ***<control_type>*** can be set to 0 for fixed output voltage override or 1 for applying offset to output voltage.

***<target>_IR3567B_Output = <output_index>***

where ***<output_index>*** can be set to 0 to control the first voltage loop or 1 to control the second voltage loop. This setting is optional, MSI Afterburner defaults to the first loop if output index is not specified.

### 2.1.6. IR3595A voltage controller

IR3595A is single-loop controller, which can control only one voltage output. IR3595A supports voltage control in output voltage override form only. IR3595A supports output voltage monitoring in real form, which means that it can report real monitored voltage only. This controller can reside in ***full 7-bit I2C device address range***. The controller is identified by registers ***FCh*** (must be equal to ***49h***), ***FDh*** (must be equal to ***52h***) and ***FBh*** (must be equal to ***27h, 28h, 29h*** or ***2Ah***). Model specific settings for IR3595A include the following:

***<target>_IR3595A_Output = <output_index>***

where ***<output_index>*** can be set to 0 to control the first voltage loop or 1 to control the second voltage loop. This setting is optional, MSI Afterburner defaults to the first loop if output index is not specified.

### 2.1.7. L6788A voltage controller

L6788A is single-loop controller, which can control only one voltage output. L6788A supports internal lookup table (LUT) of voltages for all possible GPU power states, which means that it allows controlling 3D performance state voltage independently of the rest performance states. This controller can reside in fixed ***40h*** device address only. The controller is identified by register ***D0h*** (must be equal to ***53h***). Model specific settings for L6788A include the following:

***<target>_L6788A_Defaults = <vid_lut_register_address> <default_vid>***

where ***<vid_lut_register_address>*** is the address of VID register controlling voltage for 3D performance state, the address can be in ***D4h – D7h*** range.
***<default_vid>*** specify default value for previously defined VID register.

### 2.1.8. NCP4206 voltage controller

NCP4206 is single-loop controller, which can control only one voltage output. NCP4206 supports voltage control in output voltage override form only. NCP4206 supports output voltage monitoring in real form, which means that it can report real monitored voltage only.
This controller can reside in fixed ***20h*** device address only. The controller is identified by register ***99h*** (must be equal to ***41h***). The controller has no model specific settings.

*Implementation note: This controller is invisible in standard I2C dump tool due to its architecture specific and it needs to be dumped using special parameters of polled directly. Refer to chapter 4 to get more detailed info.*

### 2.1.9. NCP81022 voltage controller

NCP81022 is single-loop controller, which can control only one voltage output. NCP81022 supports voltage control in offset form only. NCP81022 supports output voltage monitoring in VID readback form, which means that it can report target programmed voltage only.
This controller can reside in ***20h - 27h*** device address range only. The controller is identified by 2-byte register ***99h*** (must be equal to ***001Ah***) and 2-byte register ***9Ah*** (must be equal to ***1022h***). The controller has no model specific settings.

### 2.1.10. UP1637 voltage controller

UP1637 is single-loop controller, which can control only one voltage output. UP1637 supports internal lookup table (LUT) of voltages for all possible GPU power states, which means that it allows controlling 3D performance state voltage independently of the rest performance states. UP1637 supports output voltage monitoring in VID readback form, which means that it can report target programmed voltage only.
This controller can reside in ***46h - 47h*** device address range only. The controller is identified by register ***D0h*** (must be equal to ***1Eh***). Model specific settings for UP1637 include the following:

***<target>_UP1637_Defaults = <vid_lut_register_address> <default_vid>***

where ***<vid_lut_register_address>*** is the address of VID register controlling voltage for 3D performance state, the address can be in ***D4h – D7h*** range.
***<default_vid>*** specify default value for previously defined VID register.

### 2.1.11. UP6204 voltage controller

UP6204 is single-loop controller, which can control only one voltage output. UP6204 supports internal lookup table (LUT) of voltages for all possible GPU power states, which means that it allows controlling 3D performance state voltage independently of the rest performance states. UP6204 supports output voltage monitoring in VID readback form, which means that it can report target programmed voltage only.
This controller can reside in fixed ***40h*** device address only. The controller is identified by register ***D0h*** (must be equal to ***11h***). Model specific settings for UP6204 include the following:

***<target>_UP6204_Defaults = <vid_lut_register_address> <default_vid>***

where ***<vid_lut_register_address>*** is the address of VID register controlling voltage for 3D performance state, the address can be in ***D4h – D7h*** range.
***<default_vid>*** specify default value for previously defined VID register.

### 2.1.12. UP6208 voltage controller

UP6208 is single-loop controller, which can control only one voltage output. UP6208 supports voltage control in offset form only. UP6208 supports output voltage monitoring in VID readback form, which means that it can report target programmed voltage only.
This controller can reside in ***45h - 47h*** device address range only. The controller is identified by register ***B2h*** (must be equal to ***01h***) and register ***0Fh*** (must be equal to ***11h***). The controller has no model specific settings.

### 2.1.13. UP6218 voltage controller

UP6218 is single-loop controller, which can control only one voltage. UP6218 supports voltage control in offset form only. UP6218 supports output voltage monitoring in VID readback form, which means that it can report target programmed voltage only.
This controller can reside in ***45h - 47h*** device address range only. The controller is identified by register ***B2h*** (must be equal to ***0Eh***) and register ***1Dh*** (lower 6 bits must be equal to ***02h*** or ***03h*** depending on the controller revision). The controller has no model specific settings.

### 2.1.14. UP6262 voltage controller

UP6262 is triple-loop controller, which can control up to three voltage outputs independently. UP6262 supports voltage control in offset form only. UP6262 doesn’t support output voltage monitoring in any form.
This controller can reside in fixed ***30h*** device address only. The controller has no identification registers. Model specific settings for UP6262 include the following:

***<target>_UP6262_Output = <output_index>***

where ***<output_index>*** can be set to 0 to control the first voltage loop, 1 to control the second voltage loop or 2 to control the third voltage loop. This setting is optional, MSI Afterburner defaults to the first loop if output index is not specified.

***<target>_UP6262_R1 = <r1_resistance>***

where ***<r1_resistance>*** specify R1 resistor resistance (in Ohm) for voltage control loop. This value is set to zero by default and must be set to non-zero value in the database.

### 2.1.15. UP6266 voltage controller

UP6266 is single loop controller, which can control only one voltage output. UP6266 supports internal lookup table (LUT) of voltages for all possible GPU power states, which means that it allows controlling 3D performance state voltage independently of the rest performance states. UP6266 supports output voltage monitoring in VID readback form, which means that it can report target programmed voltage only.
This controller can reside in ***51h - 53h*** device address range only. The controller is identified by register ***D0h*** (must be equal to ***12h***). Model specific settings for UP6266 include the following:

***<target>_UP6266_Defaults = <vid_lut_register_address> <default_vid>***

where ***<vid_lut_register_address>*** is the address of VID register controlling voltage for 3D performance state, the address can be in ***D4h – D7h*** range.
***<default_vid>*** specify default value for previously defined VID register.

### 2.1.16. VT1165 voltage controller

VT1165 is single-loop voltage controller, which can control only one voltage output. VT1165 supports internal lookup table (LUT) of voltages for all possible GPU power states, which means that it allows controlling 3D performance state voltage independently of the rest performance states. VT1165 supports output voltage monitoring in VID readback form, which means that it can report target programmed voltage only.

This controller can reside in ***70h - 71h*** device address range only. The controller is identified by register ***1Ah*** (must be equal to ***0Ah***). Model specific settings for VT1165 include the following:

***<target>_VT1165_Defaults = <vid_lut_register_address> <default_vid>***

where ***<vid_lut_register_address>*** is the address of VID register controlling voltage for 3D performance state, the address can be in ***15h – 18h*** range.
***<default_vid>*** specify default value for previously defined VID register.

*Implementation note: This controller is invisible in standard I2C dump, because it is located in upper I2C address space. Its' I2C address space needs to be scanned directly. Refer to chapter 4 to get more detailed info.*

### 2.1.17. VT1556 voltage controller

VT1556 is single-loop controller, which can control only one voltage output. VT1556 supports internal lookup table (LUT) of voltages for all possible GPU power states, which means that it allows controlling 3D performance state voltage independently of the rest performance states. VT1556 supports output voltage monitoring in VID readback form, which means that it can report target programmed voltage only.
This controller can reside in ***70h - 73h*** device address range only. The controller is identified by register ***1Ah*** (must be equal to ***02h***). Model specific settings for VT1556 include the following:

***<target>_VT1556_Defaults = <vid_lut_register_address> <default_vid>***

where ***<vid_lut_register_address>*** is the address of VID register controlling voltage for 3D performance state, the address can be in ***94h – 9Bh*** range.
***<default_vid>*** specify default value for previously defined VID register.

*Implementation note: This controller is invisible in standard I2C dump, because it is located in upper I2C address space. Its' I2C address space needs to be scanned directly. Refer to chapter 4 to get more detailed info.*

### 2.1.18. Generic voltage controller

***Generic*** is a special fake controller name allowing MSI Afterburner to use display driver's VID based voltage control API instead of direct access to specific external I2C voltage controller. In this case voltage control range can be seriously limited by display driver, but this mode is the only case for many reference design cards with cost down voltage controllers without I2C programming support. Additionally, some modern graphics processors are equipped with integrated on-die programmable voltage controllers (e.g. ***SMC*** microcontroller on ***AMD Fiji***) and direct access to such integrated controllers is mapped to ***Generic*** controller as well. Voltage controller detection field for this controller supports only ***VDDC*** as the target and uses simple boolean variable to enable/disable it instead of I2C bus and device location info, e.g.

***VDDC_Generic_Detection = 1***

enables voltage control via display driver and

***VDDC_Generic_Detection = 2***

enables voltage control via integrated on-die voltage controller. Generic voltage control has the lowest priority, so when you specify generic and some other controller models for the same card MSI Afterburner first try to use direct access to other external controllers and fall back to generic controller only if no other external controllers found onboard.

# 3. Helper tools

MSI Afterburner contains built-in helper tools aimed to simplify the process of hardware database creation and allowing you to detect and diagnose the voltage controller before adding it to hardware database. The next chapters discuss helper tools in details.

## 3.1. I2C dump tool

I2C dump tool is intended for automated scanning of all devices residing on all I2C buses of each GPU installed in the system. The first register of each scanned I2C device is being polled, if the device reply to polling then dump of 256 byte registers is saved into the dump. I2C dump tool is activated via the command line with the following switch:

***MSIAfterburner.exe /i2cd[[<i2c_bus>],[<i2c_device>],[<register>]]***

where ***<i2c_bus>*** is I2C bus number, ***<i2c_device>*** is 7-bit I2C device address and ***<register>*** is address of I2C device register to be polled when checking I2C device availability. Optional ***<i2c_bus>*** and ***<i2c_device>*** parameters allow scanning required device address on all I2C buses or desired I2C bus only. When I2C bus and device address are not specified I2C dump tool is scanning all available I2C buses and 7-bit I2C address range starting from ***00h*** up to ***4Fh*** inclusive. Optional ***<register>*** parameter allows dumping state of some sophisticated I2C devices with unreadable register 0 (e.g. ***NCP4206***).

## 3.2. I2C read/write console

I2C read / write console is intended for reading registers from or writing registers to a specific I2C device. You can read data from a register of specific I2C device using the following command line switch:

```
MSIAfterburner.exe /ri<i2c_bus>,<i2c_device>,<register>
```

where ***<i2c_bus>*** is I2C bus number, ***<i2c_device>*** is 7-bit I2C device address and ***<register>*** is register address

In addition to read operations you may also write data to a register of specific I2C device using the following command line switch:

```
MSIAfterburner.exe /wi<i2c_bus>,<i2c_device>,<register>,<data>
```

where ***<i2c_bus>*** is I2C bus number, ***<i2c_device>*** is 7-bit I2C device address, ***<register>*** is register address and ***<data>*** is a data to be written to the register

Besides direct data write there a few additional operations, allowing you to read data from I2C device register, apply simple logical operations (***AND***, ***OR*** or ***XOR***) to it and write it back to the register. The following command line switch read register form specific I2C device, apply bitwise ***AND*** operation to it using data you specify then write it back to the register:

```
MSIAfterburner.exe /ai<i2c_bus>,<i2c_device>,<register>,<data>
```

Similar to the previous command, the following commands do almost the same but use bitwise ***OR*** and ***XOR*** operations instead or ***AND*** operation:

```
MSIAfterburner.exe /oi<i2c_bus>,<i2c_device>,<register>,<data>
MSIAfterburner.exe /xi<i2c_bus>,<i2c_device>,<register>,<data>
```

The commands in I2C read / write console can be queued, which mean that you can perform series of read / write operations in one command. Please take a note that all commands apply to GPU selected as a master GPU in MSI Afterburner properties, however you can use GPU selection command ***/sg<gpu_index>*** before I2C read and write commands to redirect them to desired GPU. ***<gpu_index>*** is zero based and it affects all read and write commands in the line until the next GPU selection command is met.

Also take a note that all I2C read and write commands work with 1-byte I2C device registers by default. However, you may use access mode forcing command ***/fm2*** to force 2-byte register access mode and ***/fm1*** command to force 1-byte register access mode back if necessary. Similar to GPU selection commands, access mode forcing command also affects all read and write commands in the line until the next access mode forcing command is met.

# 4. Hardware database creation steps, hints and tricks

- The very first thing you have to do before trying to add new card to third party hardware database is to determine exact model of voltage controller chip. Some graphics card reviewers pay attention to it, for example you may find info about voltage controller model in detailed TechPowerUp graphics card reviews, e.g. this review is telling you that custom design EVGA GTX 780 Ti Classified card uses CHL8318 controller.

- If you cannot find info about voltage controller model in any online reviews, try to examine PCB and visually identify model of controller chip yourself.

- If you identified voltage controller model and it is listed as supported by MSI afterburner in chapter 2.1, proceed with the next steps. If you identified voltage controller model but it is not listed as supported by MSI afterburner in chapter 2.1, then sadly you cannot proceed further and unlock support for it in MSI Afterburner. Some voltage controller models (e.g. ***RT8802A*** or ***NCP81174***) are cost down controller models with no I2C support, so there are absolutely no chances to see support for such controllers in future versions of MSI Afterburner. Support for unsupported programmable controllers can be added to MSI Afterburner in future versions, but only if MSI start using such controller on some custom design MSI graphics cards.

- Use I2C dump tool to scan I2C devices installed on your graphics card. If you already know voltage controller model, you may use the dump to find the controller location address and it is rather easy task. Otherwise you'll need to analyze all devices found inside the dump and compare each device with each possible voltage controller model identification info. For example, let's assume that you know that your controller is CHL8228. Let's peek into the chapter about this controller. You'll see there that CHL8228 controller may reside in any address and that it can be identified by values ***03h***, ***04h*** or ***0Eh*** in register ***8Ch***. Now examine all I2C devices in the dump and select device matching with our address selection (can be any in our case) and identification criteria. In our example there is only one device inside the dump located on bus ***6*** device ***30h*** and it is our target, because register ***8Ch*** is indeed equal to ***0Eh*** (row 9 contain registers ***80h*** – ***8Fh***, column 13 in this row in register ***8Ch***):

  ```
  Scanning GPU VEN_1002&DEV_6798&SUBSYS_99991043&REV_00&BUS_3&DEV _0&FN_0...
  ...
  Scanning I2C bus 6...
  Probing device 00...
  Invalid device
  ...
  Probing device 30...
  25 CA 78 61 DB 5B 01 30 A0 08 80 00 87 7F CF 96
  A0 03 AA 06 A0 00 00 00 00 A4 89 9A 28 00 E8 A1
  28 4B DB DD 30 55 00 00 B9 B9 99 05 05 11 AA 22
  44 33 C5 00 AA 38 39 90 53 2D 00 00 00 3A 01 80
  9F CF 00 00 4B 3E 3F 3F 3F 3E 3E 3F 71 F0 F1 80
  D9 7F D9 81 D9 8B 02 1E D8 00 D0 97 BA 03 B8 20
  F0 30 F0 29 19 00 00 00 00 00 00 F8 7B 00 00 00
  00 03 00 03 51 00 CF FF 00 00 A0 EA FF 00 00 68
  D9 00 D0 4B D9 04 D0 D7 B9 FF BB 56 0E 00 01 FF
  02 07 00 00 C0 00 00 BB 00 00 12 1F 1F AD 50 00
  00 00 00 00 00 00 00 00 00 03 00 00 03 FF 9F 00
  00 00 0F 00 00 00 00 FF FF 88 88 01 0A 0A FF 88
  01 01 01 00 00 C0 9F 9F 9F 9F BF BF BF BF 01 80
  00 00 00 00 80 33 00 00 08 00 01 00 00 00 00 00
  00 00 00 15 15 00 00 00 01 00 00 00 00 01 00 00
  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  ```

  Keep in mind that in some cases default I2C dump may contain no data required for voltage controller identification. It can be the case if voltage controller reside in upper I2C addresses (***5Fh*** and higher), which are not scanned by default I2C dump scanner. For example if we're searching for ***CHL8318*** controller and know that it can reside in fixed ***70h***, ***72h***, ***74h*** and ***76h*** addresses, we can scan those addresses directly with the following commands:

  ```
  MSIAfterburner.exe /i2cd,70
  MSIAfterburner.exe /i2cd,72
  MSIAfterburner.exe /i2cd,74
  MSIAfterburner.exe /i2cd,76
  ```

  In some cases I2C device can be completely invisible to I2C dump tool because the first register of device polled by the scanner of I2C dump tool can be unreadable due to device architecture specifics. For example it applies to ***NCP4206*** voltage controllers. So if we're searching for ***NCP4206*** and it is not found inside the dump, we can specify additional parameters in I2C dump command line to force it to poll different register when checking I2C device availability, e.g. ***NCP4206*** identification register ***99h***:

  ```
  MSIAfterburner.exe /i2cd,,99
  ```

  Alternately, we can poll the controller directly via I2C read / write console. For example the following commands poll identification register ***99h*** of ***NCP4206*** device address ***20h***. We're expecting to see ***41h*** for ***NCP4206*** according to info from the chapter about NCP4206 voltage controller. So, let's assume that we poll this register and address on 2 neighbor I2C buses:

  ```
  MSIAfterburner.exe /ri3,20,99 /ri4,20,99
  ```

  And the output result is:

  ```
  I2C 03 20 99 : 41
  I2C 04 20 99 : invalid
  ```

  which means that our NCP4206 is indeed located on I2C bus ***3***, device address ***20h***.

- Once you determine model and location of your voltage controller (i.e. I2C bus number and I2C device address), you can try to add it to third party hardware database and test changes in MSI Afterburner. To simplify the process, you may create new empty database file and perform all changes inside it instead of working with massive existing database file. So start from creating ASCII text file called ***MSIAfterburner.oem2*** inside the root application folder and paste the following text inside it:

  ```
  ;OEM

  [VEN_XXXX&DEV_XXXX&SUBSYS_XXXXXXXX&REV_??]

  Desc                                    = My card
  ```

Don't forget the ***";OEM"*** signature in the very beginning, otherwise application will not recognize it as valid database file, and use correct section name uniquely identifying your graphics card, i.e. specify the real vendor, device and subsystem IDs of your graphics card instead of XXXX. Restart MSI Afterburner and if you did everything properly, it will unlock new item in the list of voltage control modes available in ***"General"*** tab in application properties: ***"third party"*** mode should become available for selection. Once you select it, MSI Afterburner will offer you to restart application and if you did everything properly, you'll see your card detected as ***"My card"*** in MSI Afterburner after restart. As you probably noticed, ***"My card"*** is a text description, which we specify in ***Desc*** field of newly created database entry, it is the easiest way to ensure that we added new database entry properly.

Now you can change ***Desc*** field to the real name of your card try to add voltage control detection field to new database entry. For example, if we're creating database entry for a card, which control core voltage via ***CHL8228*** on I2C bus ***6*** device address ***30h***, we add the following:

```
VDDC_CHL8228_Detection                          = 6:30h
```

For VDDC voltage control on NVIDIA graphics cards it is also recommended to forcibly disable generic voltage control via display driver. Otherwise MSI Afterburner may fall back to generic voltage control mode if you specify VDDC controller detection field improperly, so it will be harder for you to notice typos in the database if you make any. Generic voltage control mode can be disabled by adding the following line to new database entry:

```
VDDC_Generic_Detection                          = 0
```

Database creation ends at this step for many voltage controllers having no controller specific settings. However, some models of voltage controllers require some more calibration information to be specified inside the database. Additional model specific settings are documented in controller specific chapters of this document, so carefully examine the chapter related to your voltage controller model.

In out example with ***CHL8228***, such model specific setting is ***VDDC_CHL8228_Defaults*** field, which is selecting proper voltage control loop and LUT register for it. According to the chapter 2.1.2, dual-loop ***CHL8228*** supports internal lookup table (LUT) of voltages for all possible GPU power states, which means that it allows controlling 3D performance state voltage independently of the rest performance states. This means that in order to provide independent voltage control for 3D performance state, we must add to the database exact address of LUT register, storing voltage for 3D performance state in ***CHL8228*** on this card. The database should also specify default value of this LUT register to allow MSI Afterburner to restore defaults properly, when necessary.

Finding address of LUT register with voltage for 3D performance state is rather simple task. For example, according to the chapter 2.1.2, LUT registers for the first ***CHL8228*** voltage control loop are ***C6h - C9h***. So we can simply try the database with each of 4 possible LUT registers to find one mapped to 3D performance state. Alternately, you can select LUT register for 3D performance state via VID decoding, but it requires deeper understanding of voltage controller functioning principles.

Finding default value of LUT register is also rather simple task. Once you determine address of LUT register, you may disable all voltage control tools in the system, perform cold system boot then use I2C read / write console to read default value of this register.

- Finally, share all data you've collected (full model name of your custom design graphics card, voltage controller mode, database section name for your graphics card, I2C dumps, results of direct polling of I2C devices, or even complete new database entry for it) with us in Guru3D forums. We'll verify it and add it to public version of third party hardware database, if possible.