

Linux WMI Driver User Guide

Most confidential	
Confidential	
Internal	
Public	V

07/04/2022
Revision 1.6

Revision History	3
1. Introduction	4
2. Features	4
2.1 Basic Access Method	4
2.2 BFPI Version	5
3. Method for AAEON BFPI v0.3	6
3.1 DIO Functions	6
3.2 WDT Functions	8
3.3 HWM Functions	9
3.4 Smart Fan Functions	11
4. Method for AAEON BFPI v0.4 and v0.5	12
4.1 DIO Functions	12
4.2 WDT Functions	13
4.3 HWM Functions	14
4.4 Smart Fan Functions	15
4.5 Backlight controller Functions	17
5. Method for AAEON BFPI v0.6	18
5.1 DIO Functions	18
5.2 WDT Functions	20
5.3 HWM Functions	22
5.4 Smart Fan Functions	24
5.5 Backlight controller Functions	26
5.6 LED	27

Revision History

Version Number	Description	Revision Date
V1.6	<ul style="list-style-type: none"> • Add Chapter 5 • Support AAEON BFPI v0.6 • Supported Ubuntu 20.04.4 	07/04/2022
V1.5	<ul style="list-style-type: none"> • Modify Chapter 3.4 & Chapter 4.4. Remove 'Slope-linear mode', 'SMART FAN IV Mode' in chapter 3.4. Remove 'SMART FAN IV Mode' in chapter 4.4. • Add Chapter 4.5 • Modify Chapter 3.1 & Chapter 3.2. Add 'Report DIO Capability' in Chapter 3.1. Add 'Report capabilities' in Chapter 3.2. 	01/06/2022
V1.4	Support AAEON BFPI v0.4	06/25/2021
V1.3	Rename 'ACPI WMI' to 'BFPI'	05/21/2021
V1.2	Modify Chapter 3.4 & Chapter 4.4	05/06/2021
V1.1	Add AAEON ACPI WMI v0.3 support	04/12/2021
V1.0	Rename document name from 'Hardware Control User Guide' to 'AAEON Linux WMI Driver User Guide'	04/29/2020
V0.3	Modify Chapter 1	04/07/2020
V0.2	Add revision history section and modify Chapter 1 for COM-WHUC6 BIOS version CWHUAM12	03/02/2020
V0.1	Initial Release	02/15/2019

1. Introduction

AAEON Linux WMI Driver provides an interface to communicate with the hardware. Users can control and collect data from hardware devices. There are several powerful functions supported by AAEON Linux WMI Driver, such as Hardware Monitor, Smart Fan, Watchdog, Digital IO, and etc.

2. Features

2.1 Basic Access Method

Users can control hardware by asus-nb-wmi driver

1. Type "sudo su" to get permission
2. Enter the user password
3. Type " cd /sys/kernel/debug/asus-nb-wmi"
4. Type "ls -l" to review the node

```
sdd2@sdd2-desktop:~$ sudo su
[sudo] password for sdd2:
root@sdd2-desktop:/home/sdd2# cd /sys/kernel/debug/asus-nb-wmi/
root@sdd2-desktop:/sys/kernel/debug/asus-nb-wmi# ls -l
total 0
drwxr-xr-x  2 root root 0  — 17 11:54 ./
drwx----- 37 root root 0  — 17 11:54 ../
-r--r--r--  1 root root 0  — 17 11:54 call
-rw-r--r--  1 root root 0  — 17 11:54 ctrl_param
-rw-r--r--  1 root root 0  — 17 11:54 dev_id
-r--r--r--  1 root root 0  — 17 11:54 devs
-r--r--r--  1 root root 0  — 17 11:54 dsts
-rw-r--r--  1 root root 0  — 17 11:54 method_id
root@sdd2-desktop:/sys/kernel/debug/asus-nb-wmi#
```

For example, if you want to read the DIO0 logical level, please type the following commands.

```
root@sdd2-desktop:/sys/kernel/debug/asus-nb-wmi# echo 0x00010001 > method_id
root@sdd2-desktop:/sys/kernel/debug/asus-nb-wmi# echo 0x0 > dev_id
root@sdd2-desktop:/sys/kernel/debug/asus-nb-wmi# cat call
0x10001(0x0, 0x0) = 0x1
```

1. echo "method_id" > method_id

2. echo "DIO pin number" > dev_id
3. cat call
4. Return 0x1: high

"0x10001 (0x0, 0x0) = 0x1"
"method_id (dev_id, ctrl_param) = return value"

Note: For more examples, please refer to the following section.

2.2 BFPI Version

Note: Please check the AAEON BFPI version of your mainboard. To check the version, please use the following method to read the version number from your mainboard. For BFPI version 0.3, please refer to chapter 3. For BFPI version 0.4 and 0.5, please refer to chapter 4. If there is any question about the supported functions, please contact the regional FAE for getting more details.

- **Get Revision of AAEON BFPI Spec.**

e.g.
 echo 0x00000000 > method_id
 echo 0x00 > dev_id
 cat call

The result would be "0x0(0x0, 0x0) = 0x3" or "0x0(0x0, 0x0) = 0x5".

	method_id	dev_id	return value
Get Revision of AAEON BFPI Spec.	0x00000000	0x00	Integer: V[Major].[Minor] Bit [15:0]: Minor Version of spec Bit[31:16]: Major Version of spec

3. Method for AAEON BFPI v0.3

3.1 DIO Functions

- **Get Digital I/O Level**
- **Set Digital I/O Level**
- **Get Digital I/O Direction**
- **Set Digital I/O Direction**

	method_id	dev_id	return value
Report DIO Capability	0x00010000	Arg0 : Select capability data to return - 0x10: Supported pin number on this instance 0x11: Supported GPIO Input Bit Map 0x12: Supported GPIO Output Bit Map	Return data according to input Arg0: 0x10: Interger 0x11: Interger, each bit represents correspond GPIO supports input mode or not. 1:supported/0:not supported. 0x12: Interger, each bit represents correspond GPIO supports output mode or not. 1:supported/0:not supported.
Get Digital I/O Level	0x00010001	Arg0 : DIO number to get 0: DIO0 1: DIO1 ... 63: DIO63	Integer: 0 - Low level 1 - High level INVALID_PARAMETER: an invalid number is given
Set Digital I/O Level	0x00010002	Arg0 : Bit [7:0] - DIO number to set 0: DIO0 1: DIO1 ... 63: DIO63 Bit [16] - 0:low/1:high	Integer: 0 - Success INVALID_PARAMETER: an invalid number is given
Get Digital I/O Direction	0x00010003	Arg0: DIO number to get 0: DIO0 1: DIO1 ... 63: DIO63	Integer 0: Output 1: Input INVALID_PARAMETER: an invalid number is given
Set Digital I/O Direction	0x00010004	Arg0 : Bit [7:0] - DIO number to set 0: DIO0 1: DIO1 ...	Integer: 0 - Success INVALID_PARAMETER: an invalid number is given

		63: DIO63 Bit [16] - 0:Output/1:Input Other bits are reserved.	
--	--	--	--

3.2 WDT Functions

- **Get watchdog timeout value**

e.g.
echo 0x00020001 > method_id
cat call

- **Set watchdog timeout value and start/stop watchdog**

e.g.
Set timeout (60 s) and start watchdog:

```
echo 0x00020002 > method_id
echo 0x3C > dev_id
cat call
```

e.g.
Stop watchdog:

```
echo 0x00020002 > method_id
echo 0 > dev_id
cat call
```

	method_id	dev_id	return value
Report capabilities	0x00020000	Arg0: 0x10: Return Max. supported timeout in seconds	Integer: Max. supported timeout in seconds.
Get watchdog timeout value	0x00020001		Integer: Remaining time to time-out in seconds.
Set watchdog timeout value and start/stop watchdog	0x00020002	Arg0: timeout value in seconds 0: stop watchdog, Others: Set a watchdog timeout value and start watchdog. The watchdog timeout value support range is from 0x00 to 0xff seconds.	Integer: 0 - Success INVALID_PARAMETER - Input time exceeds the maximum supported timeout.

3.3 HWM Functions

- **Get Temperature**

e.g.

Get CPU Temperature

```
echo 0x00030001 > method_id
echo 0x2000 > dev_id
cat call
```

Get SYS Temperature

```
echo 0x00030001 > method_id
echo 0x1000 > dev_id
cat call
```

- **Get Fan Speed**

e.g.

Get CPU Fan Speed

```
echo 0x00030001 > method_id
echo 0x2100 > dev_id
cat call
```

- **Get Voltage**

e.g.

Get +12V Voltage

```
echo 0x00030001 > method_id
echo 0x4200 > dev_id
cat call
```

Then multiply the obtained value by 96.

	method_id	dev_id	return value
Read Sensor	0x00030001	Arg0: Select sensor to read Bit[11:8]: Sensor type - 0 - Temperature 1 - Fan 2 - Voltage Bit[15:12]: Sensor number - Temperature: 1: SYS Temperature 2: CPU Temperature FAN: 1: Chassis FAN 2: CPU FAN Voltage: 0: V0 1: V1 2: V2 3: V3 4: V4 5: V5 6: V6 7: V7 8: V8	Reading value report by sensor Temperature: Signed integer, temperature in degree-C FAN : Integer, fan speed in rpm Voltage : Integer, these returned values (V0, V1... V8) are raw data, using the below formulas to translate to actual voltage value (mV). $VCORE = V0 \times 16$ $+5V = V1 \times 2008 / 50$ $AVCC = V2 \times 16$ $+3.3V = V3 \times 16$ $+12V = V4 \times 96$ $VCOREREFIN = V5 \times 552 / 41$ $VIN4 = V6 \times 8$ $3VSB = V7 \times 16$ $VBAT = V8 \times 16$ NOT_SUPPORTED - Correspond sensor is not present

3.4 Smart Fan Functions

- **Get Fan Mode and PWM Value**

e.g.

Get SYS1 Fan Mode and PWM Value for Manual Mode

```
echo 0x00050001 > method_id
echo 0x1 > dev_id
cat call
```

- **Set Fan Mode and PWM Value**

e.g.

Set SYS1 Fan Mode - Linear Mode

```
echo 0x00050002 > method_id
echo 0x011 > dev_id
cat call
```

e.g.

Set SYS1 Fan Mode - Manual Mode

```
echo 0x00050002 > method_id
echo 0x640001 > dev_id
cat call
```

	method_id	dev_id	return value
Get fan mode	0x00050001	Arg0 : Bit[3:0]: Fan Instance 0: Controller - 0 (CPU Fan) 1: Controller - 1 (SYS1 Fan)	Return data Bit[3:0]: Fan Mode 0 - Manual mode 1 - Linear mode (Points-linear mode) Bit[7:4]: Reserved Bits Bit[15:8]: PWM value
Set fan mode	0x00050002	Arg0 : Bit[3:0]: Fan Instance 0: Controller - 0 (CPU Fan) 1: Controller - 1 (SYS1 Fan) Bit[7:4]: Mode to set 0 - Manual mode 1 - Linear mode (Points-linear mode) Bit[8]: Reserved, should be 0 for basic protocol. Bit[23:16]: PWM value for manual mode	Integer: 0 - Success Others - Fail with status code INVALID_PARAMETER : Given duty or slope is larger than Max. supported NOT_SUPPORTED

4. Method for AAEON BFPI v0.4 and v0.5

4.1 DIO Functions

- **Get Digital I/O Level**
- **Set Digital I/O Level**
- **Get Digital I/O Direction**
- **Set Digital I/O Direction**

	method_id	dev_id	return value
Get Digital I/O Level	0x00010001	Arg0 : DIO number to get 0: DIO0 1: DIO1 ... 64: DI O65	Integer: 0 - Low level 1 - High level INVALID_PARAMETER: an invalid number is given
Set Digital I/O Level	0x00010002	Arg0 : Bit [7:0] - DIO number to set 0: DIO0 1: DIO1 ... 64: DIO65 Bit [16] - 0:low/1:high	Integer: 0 - Success INVALID_PARAMETER: an invalid number is given
Get Digital I/O Direction	0x00010003	Arg0: DIO number to get 0: DIO0 1: DIO1 ... 64: DIO65	Integer 0: Output 1: Input INVALID_PARAMETER: an invalid number is given
Set Digital I/O Direction	0x00010004	Arg0 : Bit [7:0] - DIO number to set 0: DIO0 1: DIO1 ... 64: DIO65 Bit [16] - 0:Output/1:Input Other bits are reserved.	Integer: 0 - Success INVALID_PARAMETER: an invalid number is given

4.2 WDT Functions

- **Get Max. supported timeout in ms**

e.g.
 echo 0x00020000 > method_id
 echo 0x10 > dev_id
 cat call

- **Get watchdog timeout value**

e.g.
 echo 0x00020001 > method_id
 cat call

- **Set watchdog timeout value and start/stop watchdog**

e.g.
 Set timeout (60000 ms) and start watchdog:

```
echo 0x00020002 > method_id
echo 0xea60 > dev_id
cat call
```

e.g.
 Stop watchdog:

```
echo 0x00020002 > method_id
echo 0x0 > dev_id
cat call
```

	method_id	dev_id	return value
Report capabilities	0x00020000	Arg0: 0x10: Return Max. supported timeout in ms	Integer: Max. supported timeout in ms.
Get watchdog timeout value	0x00020001		Integer: Remained time to time-out in ms
Set watchdog timeout value and start/stop watchdog	0x00020002	Arg0: timeout value in ms 0: stop watchdog, Others: Set a watchdog timeout value and start watchdog.	Integer: 0 - Success INVALID_PARAMETER - Input time exceeds the maximum supported timeout.

4.3 HWM Functions

- **Get Temperature**

e.g.

Get CPU Temperature

```
echo 0x00030001 > method_id
echo 0x0000 > dev_id
cat call
```

- **Get Fan Speed**

e.g.

Get SYS1 Fan Speed

```
echo 0x00030001 > method_id
echo 0x1100 > dev_id
cat call
```

- **Get Voltage**

e.g.

Get VMEM Voltage

```
echo 0x00030001 > method_id
echo 0x1200 > dev_id
cat call
```

	method_id	dev_id	return value
Read Sensor	0x00030001	Arg0: Select sensor to read Bit[11:8]: Sensor type - 0 - Temperature 1 - Fan 2 - Voltage Bit[15:12]: Sensor number - Temperature: 0: CPU Temperature 1: SYS1 Temperature 2: SYS2 Temperature FAN: 0: CPU FAN 1: SYS1 FAN 2: SYS2 FAN 3: Chasis1 FAN 4: Chasis2 FAN	Reading value report by sensor Temperature: Signed integer, temperature in millidegree celsius FAN : integer, fan speed in rpm Voltage : integer, voltage in millivolt NOT_SUPPORTED - Correspond sensor is not present

		Voltage: 0: VCORE 1: VMEM 2: +12V 3: +5V 4: +3.3V 5: +1.8V 6: 5VSB 7: 3VSB 8: VBAT	
--	--	---	--

4.4 Smart Fan Functions

- **Get Fan Mode and PWM Value**

e.g.

Get SYS1 Fan Mode and PWM Value for Manual Mode

```
echo 0x00050001 > method_id
echo 0x1 > dev_id
cat call
```

- **Set Fan Mode and PWM Value**

e.g.

Set SYS1 Fan Mode - Slope-linear Mode

```
echo 0x00050002 > method_id
echo 0x021 > dev_id
cat call
```

e.g.

Set SYS1 Fan Mode - Manual Mode

```
echo 0x00050002 > method_id
echo 0x640001 > dev_id
cat call
```

	method_id	dev_id	return value
Get fan mode	0x00050001	Arg0 : Bit[3:0]: Fan Instance 0: Controller - 0 (CPU Fan) 1: Controller - 1 (SYS1 Fan)	Return data Bit[3:0]: Fan Mode 0 - Manual mode

		<p>2: Controller - 2 (SYS2 Fan) 3: Controller - 3 (Chasis1 Fan) 4: Controller - 4 (Chasis2 Fan)</p>	<p>1 - Linear mode (Points-linear mode) 2 - Slope-linear mode Bit[7:4]: Reserved Bits Bit[15:8]: PWM value</p>
Set fan mode	0x00050002	<p>Arg0 : Bit[3:0]: Fan Instance 0: Controller - 0 (CPU Fan) 1: Controller - 1 (SYS1 Fan) 2: Controller - 2 (SYS2 Fan) 3: Controller - 3 (Chasis1 Fan) 4: Controller - 4 (Chasis2 Fan) Bit[7:4]: Mode to set 0 - Manual mode 1 - Linear mode (Points-linear mode) 2 - Slope-linear mode Bit[8]: Reserved, should be 0 for basic protocol. Bit[23:16]: Duty cycle PWM value for manual mode</p>	<p>Interger: 0 - Success Others - Fail with status code INVALID_PARAMETER : Given duty or slope is larger than Max. supported NOT_SUPPORTED</p>

4.5 Backlight controller Functions

- **Get backlight brightness Value**

e.g. Get Panel - 1 backlight brightness Value

```
echo 0x00040001 > method_id
echo 0x1 > dev_id
cat call
```

- **Set backlight brightness Value to 100**

e.g. Set Panel - 2 backlight brightness Value

```
echo 0x00040002> method_id
echo 0x264 > dev_id
cat call
```

	method_id	dev_id	return value
Get backlight brightness	0x00040001	Arg0: Panel instance Bit[3:0]: 0 - Panel - 0 1 - Panel - 1 2 - Panel - 2 3 - Panel - 3	Integer: Current brightness level
Set backlight brightness	0x00040002	Arg0: Brightness level to set 0~255 - 0 is lowest brightness, 255 is highest brightness. Arg0: Bit[7:0] PWM value, 0~255 Bit[9:8]: Panel instance 0 - Panel - 0 1 - Panel - 1 2 - Panel - 2 3 - Panel - 3 Arg1: Brightness level (when it is larger than 255)	integer: 0 - Success Others - Fail with status code

5. Method for AAEON BFPI v0.6

5.1 DIO Functions

- Get Digital I/O Level
- Set Digital I/O Level
- Get Digital I/O Direction
- Set Digital I/O Direction
- Get Digital I/O Driving
- Set Digital I/O Driving

Note: (TBD) If the hardware implementation of GPIO does not support selected driving level, next lower level is selected.

	method_id	dev_id	return value
Get Digital I/O Level	0x00010001	Arg0 : DIO number to get 0: DIO0 1: DIO1 ... 64: DIO65	Integer: 0 - Low level 1 - High level INVALID_PARAMETER: an invalid number is given
Set Digital I/O Level	0x00010002	Arg0 : Bit [7:0] - DIO number to set 0: DIO0 1: DIO1 ... 64: DIO65 Bit [16] - 0:low/1:high	Integer: 0 - Success INVALID_PARAMETER: an invalid number is given
Get Digital I/O Direction	0x00010003	Arg0: DIO number to get 0: DIO0 1: DIO1 ... 64: DIO65	Integer 0: Output 1: Input INVALID_PARAMETER: an invalid number is given
Set Digital I/O Direction	0x00010004	Arg0 : Bit [7:0] - DIO number to set 0: DIO0 1: DIO1 ... 64: DIO65 Bit [16] - 0:Output/1:Input Other bits are reserved.	Integer: 0 - Success INVALID_PARAMETER: an invalid number is given

Get Digital I/O Driving	0x00010005	Arg0: DIO number to get0: DIO0 1: DIO1 ... 64: DIO65	integer: 0: Open drain 1: Push pull / Internal pull-up 20K 2: Internal pull-up 10K 3: Internal pull-up 5K 4: Internal pull-up 1K INVALID_PARAMETER: a invalid number is given
Set Digital I/O Driving	0x00010006	Arg0: Bit [7:0] - DIO number to set Bit [20:16] - 0: Open drain 1: Push pull / Internal pull-up 20K 2: Internal pull-up 10K 3: Internal pull-up 5K 4: Internal pull-up 1K Other bits are reserved.	Integer: 0 - Success INVALID_PARAMETER: an invalid number is given

5.2 WDT Functions

- **Get Max. supported timeout in ms**

e.g.
echo 0x00020000 > method_id
echo 0x10 > dev_id
cat call

- **Get watchdog timeout value**

e.g.
echo 0x00020001 > method_id
cat call

- **Set watchdog timeout value and start/stop watchdog**

e.g.
Set timeout (60000 ms) and start watchdog:

```
echo 0x00020002 > method_id
echo 0xea60 > dev_id
cat call
```

e.g.
Stop watchdog:

```
echo 0x00020002 > method_id
echo 0x0 > dev_id
cat call
```

	method_id	dev_id	return value
Report capabilities	0x00020000	Arg0: 0x10: Return Max. supported timeout in ms 0x12: Supported WatchDog Sensor Instance Bitmap Arg1: When Arg0 (0x10), Select WDT instance to return 0: Controller - 0 1: Controller - 1 2: Controller - 2 3: Controller - 3 ...	Return data according to input Arg0 (0x10): integer: Max. supported timeout in ms. Arg0 (0x12): integer, each bit represents correspond WatchDog sensor instance is supported or not. 1: supported/0: not supported. Bit0: Sensor Controller - 0 Bit1: Sensor Controller - 1 Bit2: Sensor Controller - 2 Bit3: Sensor Controller - 3 ... Arg0 (Others): INVALID_PARAMETER: a invalid number is given

Get watchdog timeout value	0x00020001	Arg1: Select WDT instance to get 0: Controller - 0 1: Controller - 1 2: Controller - 2 3: Controller - 3 ...	Integer: Remained time to time-out in ms
Set watchdog timeout value and start/stop watchdog	0x00020002	Arg0: timeout value in ms 0: stop watchdog, Others: Set as watchdog timeout value and start watchdog. Arg1: Select WDT instance to set 0: Controller - 0 1: Controller - 1 2: Controller - 2 3: Controller - 3 ...	Integer: 0 - Success INVALID_PARAMETER - Input time exceeds the maximum supported timeout.

5.3 HWM Functions

- **Get Temperature**

e.g.

Get CPU Temperature

```
echo 0x00030001 > method_id
echo 0x0000 > dev_id
cat call
```

- **Get Fan Speed**

e.g.

Get SYS1 Fan Speed

```
echo 0x00030001 > method_id
echo 0x1100 > dev_id
cat call
```

- **Get Voltage**

e.g.

Get VMEM Voltage

```
echo 0x00030001 > method_id
echo 0x1200 > dev_id
cat call
```

	method_id	dev_id	return value
Read Sensor	0x00030001	Arg0: Select sensor to read - Bit [11:8]: Sensor type - 0 - Temperature 1 - Fan 2 - Voltage Bit [15:12]: (Valid for standard mapping) Sensor number - Temperature: 0: CPU Temperature 1: SYS1 Temperature 2: SYS2 Temperature FAN: 0: CPU FAN 1: SYS1 FAN 2: SYS2 FAN 3: Chasis1 FAN 4: Chasis2 FAN Voltage:	Reading value report by sensor Temperature: Signed integer, temperature in millidegree Celsius FAN: integer, fan speed in rpm Voltage: integer, voltage in millivolt NOT_SUPPORTED - Correspond sensor is not present

		0: VCORE 1: VMEM 2: +12V 3: +5V 4: +3.3V 5: +1.8V 6: 5VSB 7: 3VSB 8: VBAT	
--	--	---	--

5.4 Smart Fan Functions

- **Get Fan Mode and PWM Value**

e.g.

Get SYS1 Fan Mode and PWM Value for Manual Mode

```
echo 0x00050001 > method_id
echo 0x1 > dev_id
cat call
```

- **Set Fan Mode and PWM Value**

e.g.

Set SYS1 Fan Mode - Slope-linear Mode

```
echo 0x00050002 > method_id
echo 0x021 > dev_id
cat call
```

e.g.

Set SYS1 Fan Mode - Manual Mode

```
echo 0x00050002 > method_id
echo 0x640001 > dev_id
cat call
```

	method_id	dev_id	return value
Get fan mode	0x00050001	Arg0 : Bit[3:0]: Fan Instance 0: Controller - 0 (CPU Fan) 1: Controller - 1 (SYS1 Fan)	Return data Bit[3:0]: Fan Mode 0 - Manual mode

		<p>2: Controller - 2 (SYS2 Fan) 3: Controller - 3 (Chasis1 Fan) 4: Controller - 4 (Chasis2 Fan)</p>	<p>3 - Linear mode (Points-linear mode) 4 - Slope-linear mode Bit [7:4]: Reserved Bits Bit[15:8]: PWM value</p>
Set fan mode	0x00050002	<p>Arg0: Bit [3:0]: Fan Instance 0: Controller - 0 (CPU Fan) 1: Controller - 1 (SYS1 Fan) 2: Controller - 2 (SYS2 Fan) 3: Controller - 3 (Chasis1 Fan) 4: Controller - 4 (Chasis2 Fan) Bit [7:4]: Mode to set 0 - Manual mode 1 - Linear mode (Points-linear mode) 3 - Slope-linear mode Bit [8]: Reserved, should be 0 for basic protocol. Bit [23:16]: Duty cycle PWM value for manual mode</p>	<p>Integer: 0 - Success Others - Fail with status code INVALID_PARAMETER: Given duty or slope is larger than Max. supported NOT_SUPPORTED</p>

5.5 Backlight controller Functions

- **Get backlight brightness Value**

e.g. Get Panel - 1 backlight brightness Value

```
echo 0x00040001 > method_id
echo 0x1 > dev_id
cat call
```

- **Set backlight brightness Value to 100**

e.g. Set Panel - 2 backlight brightness Value

```
echo 0x00040002> method_id
echo 0x264 > dev_id
cat call
```

	method_id	dev_id	return value
Get backlight brightness	0x00040001	Arg0: Panel instance Bit [3:0]: 4 - Panel - 0 5 - Panel - 1 6 - Panel - 2 7 - Panel - 3	Integer: Current brightness level
Set backlight brightness	0x00040002	Arg0: Brightness level to set 0~255 - 0 is lowest brightness, 255 is highest brightness. Arg0: Bit [7:0] PWM value, 0~255 Bit [9:8]: Panel instance 4 - Panel - 0 5 - Panel - 1 6 - Panel - 2 7 - Panel - 3 Arg1: Brightness level (when it is larger than 255)	integer: 0 - Success Others - Fail with status code

5.6 LED

- **Get LED Brightness Value**

e.g. Get LED - 1 status Value

```
echo 0x00060001 > method_id
echo 0x1 > dev_id
cat call
```

- **Set LED Brightness Value**

e.g. Set LED - 2 status Value

```
echo 0x00060002 > method_id
echo 0x10002 > dev_id
cat call
```

	method_id	dev_id	return value
Report Capabilities	0x00060000	Arg0: Bit [7:0]: Select capability data to return - 0x00: (Extension) Label of this function 0x10: Supported LED number	Return data according to input Arg0 (0x10): integer, LED number
Get LED Brightness	0x00060001	Arg0: LED number to get 0 - LED1 1 - LED2 ... 15 - LED16 ...	integer: 0 - Off 1 - On INVALID_PARAMETER: a invalid number is given

Set LED Brightness	0x00060002	Arg0: Bit [7:0] - E34:E38LED number to set 0 - LED1 1 - LED2 ... 16 - LED16 ... Bit [16] - 0: Off/1: On	integer: 0 - Success INVALID_PARAMETER: a invalid number is given
-----------------------	------------	---	--