

Linux WMI Driver User Guide

Most Confidential	
Confidential	
Internal	
Public	V

12/12/2024
Revision 2.1

Revision History.....	3
1 Introduction.....	5
2 Features	5
2.1 Basic Access Method	5
2.2 DIO Functions	7
2.3 WDT Functions.....	9
2.4 HWM Functions.....	11
2.5 Smart Fan Functions	13
2.6 Backlight Controller Functions	15
2.7 LED.....	16

Revision History

Version Number	Description	Revision Date
V2.1	Verified on Ubuntu 24.04.1	12/12/2024
V2.0	<ul style="list-style-type: none"> ● Verified on Ubuntu 22.04.4 ● Removed chapters 3 to 5 ● Moved the content of 'Method for AAEON BFPI v0.7' to 'Features' 	08/06/2024
V1.9	Support AAEON BFPI v0.7	09/18/2023
V1.8	<ul style="list-style-type: none"> ● fix some issues ● Add content to Chapter 5.5 	09/14/2023
V1.7	<ul style="list-style-type: none"> ● Supported Ubuntu 22.04.2 ● Fix format error of fan and backlight 	05/25/2023
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, 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"

Note1: If you got "0xFFFFFFFF" as the return value, it means "NOT_SUPPORTED", this device does not support this function. if the value is "0xFFFFFFFFE", it means "INVALID_PARAMETER ", the parameter that you input is invalid.

Note2: For more examples, please refer to the following section.

2.2 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	Integer: 0 – Success INVALID_PARAMETER: an invalid number is given

		Other bits are reserved.	
Get Digital I/O Driving	0x00010005	Arg0: DIO number to get 0: 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: an 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

2.3 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

			<p>Bit2: Sensor Controller – 2 Bit3: Sensor Controller – 3 ...</p> <p>Arg0 (Others): INVALID_PARAMETER – an invalid number is given NOT_SUPPORTED – Corresponding controller is not present</p>
Get watchdog timeout value	0x00020001	<p>Arg1: Select WDT instance to get 0: Controller – 0 1: Controller – 1 2: Controller – 2 3: Controller – 3 ...</p>	<p>Integer: Remained time to time-out in ms</p>
Set watchdog timeout value and start/stop watchdog	0x00020002	<p>Arg0: timeout value in ms 0: stop watchdog, Others: Set as watchdog timeout value and start watchdog.</p> <p>Arg1: Select WDT instance to set 0: Controller – 0 1: Controller – 1 2: Controller – 2 3: Controller – 3 ...</p>	<p>integer: 0 – Success NOT_SUPPORTED – Corresponding controller is not present INVALID_PARAMETER – Input time exceeds maximum supported timeout.</p>
Get watchdog expired status	0x00020003	<p>Arg1: Select WDT instance to get 0 : Controller – 0 1 : Controller – 1 2 : Controller – 2 3 : Controller – 3 ...</p>	<p>integer: 0 – Not expired 1 – Expired NOT_SUPPORTED – Corresponding controller is not present, or it does not support this feature</p>

Clear watchdog expired status	0x00020004	Arg1: Select WDT instance to clear 0 : Controller – 0 1 : Controller – 1 2 : Controller – 2 3 : Controller – 3	integer: 0 – Success NOT_SUPPORTED – Corresponding controller is not present, or it does not support this feature
-------------------------------	------------	--	---

2.4 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	Reading value report by sensor or Temperature: Signed integer, temperature in millidegree Celsius

		<p>Bit [15:12]: (Valid for standard mapping) Sensor number –</p> <p>Temperature: 0: CPU Temperature 1: SYS1 Temperature 2: SYS2 Temperature</p> <p>FAN: 0: CPU FAN 1: SYS1 FAN 2: SYS2 FAN 3: Chasis1 FAN 4: Chasis2 FAN</p> <p>Voltage: 0: VCORE 1: VMEM 2: +12V 3: +5V 4: +3.3V 5: +1.8V 6: 5VSB 7: 3VSB 8: VBAT</p>	<p>FAN: integer, fan speed in rpm</p> <p>Voltage: integer, voltage in millivolt</p> <p>NOT_SUPPORTED – Correspond sensor is not present</p>
--	--	---	---

2.5 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) 2: Controller – 2 (SYS2 Fan) 3: Controller – 3 (Chasis1 Fan) 4: Controller – 4 (Chasis2 Fan)	Return data Bit[3:0]: Fan Mode 0 – Manual mode 1 – Linear mode(Points-linear mode) 2 – Slope-linear mode Bit [7:4]: Reserved Bits Bit [15:8]: PWM value

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

2.6 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

2.7 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: an 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: an invalid number is given