Privileged API User Manual

V0.3

Canaan Creative CO,. Ltd

Revision Record

Version	Date	Amendments
V0.1	2019-10-30	Modify the interface name and prefix and add interface such as save settings and enter privilege mode.
V0.2	2019-10-31	Add enable fan interface,rename the disable fan interface
V0.3	2019-11-1	Add query API such as get_DH,Get_GHSmm.etc. Add return value for each API.

1 Overall

1.1 Basic Adjustment Framework

1.1.1 Adjustment Objective

The machine can achieve certain desired goals by setting different voltage, frequency or temperature etc, and the goal include one or more of HASH rate, such as energy efficiency, temperature, and power consumption.

1.2 Description of temperature and heat sink/cooling system

1.2.1 Temperature upper limit

A single ASIC chip of 104X series could withstand up to 115℃, The system will cut off the HASH board power supply to avoid the HASH board being damaged when the chip temperature is out of limit. (only when original power supply is used.)

The protection measures above is also needed when customer use their own PSU.

1.2.2 Original FAN Cooling System

The firmware could track the setting temperature(average temperature of all ASIC) through self -regulating fan speed.

The system could not keep the setting temperature in some circumstance even the fan reach at 100% speed .That usually caused by a high ambient temperature or the power consumption. The system temperature is uncertain in these circumstances.

In extreme cases, if the fan cannot suppress the temperature , the system may enter a positive self-excited state of temperature and power consumption, that makes the temperature rising continually. As a result, the PSU will cut off the HASH board power or even enter the overload protection mode. (if any)

1.2.3 Customer Cooling System

The firmware will not be able to automatically adjust the temperature. Depending on the application scenario, the customer may need to set up an external feedback regulation loop to keep the chip at the appropriate temperature.

In this mode, a disable fan API should be invoked to disable fan system in the current firmware.

1.2.4 Heating system with chip as heat source

For applications that the operating goal is a certain chip temperature, such as a HASH-based heating system, the voltage and frequency can be adjusted to keep the system at the required temperature.

1.3 Typical Adjustment Process

a) Set the target temperature [average temperature, the current HASH board has the upper limit 105℃, otherwise it is easy to trigger single-chip over-temperature]
b) Set the voltage [see the API description and the description of the corresponding miner; for the 104x model, the set voltage range is 12.00-13.88V]
   i. Power communication uses an external bus. To avoid configuration failure caused by electromagnetic interference, you need to check the information returned by the “set voltage” API to confirm your settings.
c) Set Frequency[See API description for details].
d) Monitoring the DH of the HASH board, in some experience, if DH in the range of 0.4%~1.6%,the HASH rate and energy efficiency will reach an ideal status.
   i. Some HASH boards are difficult to reach this interval (0.4%~1.6%) under certain circumstances, just try to be as close as possible to it.
   ii. After the change of V and or F, the adjustment unit embedded in the chip will take 5 min~10min to make the DH reach a steady state.
   iii. If DH is too high, try lowering the frequency or increasing the voltage. If DH is too low, you can lower the voltage or increase the frequency.
e) To get the expected HASH rate, use the formula GHSmm*(1-DH). Use API “get_GHSmm” to get current theoretical HASH rate. It assumed that there is no mistake in HASH calculation. DH is the mistake calculation rate.
f) Monitor HASH board chip temperature
   i. If the chip temperature exceeds the target range, reduce the voltage and / or reduce the frequency
   ii. There are many temperature sensors distributed on HASH boards, which can help you querying the lowest, highest and average temperature.
g) Monitor Power supply output
   i. Some kinds of PSU has the ability to measure the output power to HASH boards, and “Pout” information can be collected as a reference (due to the complexity of the external electromagnetic environment, it is generally necessary to read “Pout” 3 times at intervals of 1 second and elimination erroneous data).
   ii. An external power meter can also be used to monitor power changes

If necessary, you can try the above steps several times to make the mining

machine work at the desired target.

1.4 Other Consideration

a) Some improper parameter adjustment may cause the communication between the main control board and the HASH board disconnect. The communication detection API (see Table 3.8) can be called intermittently during the adjustment process to confirm the communication status between the current main control board and the HASH boards. If there is a communication problem, you need to reboot the miner to recover it.
b) Since Flash is a vulnerable device, frequent write operations (invoke “save settings” API) are not recommended. Typically, you can save a set of arguments that makes your miner boot up stability in your scenario. Then , you can connect miners into your adjusting system .
c) The original PSU has an overcurrent protection. When the power consumption exceeds the power supply capacity, the PSU will cut off the HASH board power. This will cause the communication between the main control board and the HASH board interrupted. You can also invoke general API to get PSU states (the first field is error code, it means a PSU problem when it is not 0). In this case, it is necessary to cut off the input power of the PSU , wait for 1 second and then turn the power back on to recover.
d) Warning: Do not rely on the overcurrent protection of the original power supply to achieve work safety. The power supply should be installed with current limiting protection devices such as leakage protectors. During the whole adjustment period, it is necessary to pay attention to the status of the mining machine and related power systems to prevent fires and electric shock accidents.

2 Warranty terms for API applications

The privileged API is developed for customers who have the need to self-regulate the working state of the mining machine. The use of a privileged API will allow the miner to operate outside of the factory-set security boundaries, so use such APIs means that the user voluntarily abandons the mine's warranty and bears the resulting risks.

Customers are advised to fully test to reduce risk before applying the corresponding parameters in large numbers.

Please contact our sales staff before using .

3 The detail of privileged API

3.1 Overview

All privileged APIs start with “privilege|0,“ and the miner must be brought into privileged mode before calling such APIs (see Section xx).

3.2 Enter Privilege Mode

You must enter privileged mode to make all privileged APIs can be successfully invoked.
Note: Entering the privileged mode will result in the mine warranty failing. Using this interface means accepting the risk by default and voluntarily giving up the warranty.

description	Configure the mining machine to enter the privileged mode (real-time effective, restart mining machine failure).
format	privilege|0,enable_privilege
Parameter	None
Example（ linux socat）	Enable IP 192.168.189.135 miner privileged mode: echo -n “privilege|0,enable_privilege” | socat -t 30 stdio tcp:192.168.189.135:4028,shut-none && echo
Return value	STATUS=I,When=1910,Code=118,Msg=ASC 0 set info: **Privilege mode enabled! Enjoy it!**,Description=cgminer 4.11.1|

3.3 Set Frequency

description	Configure the operating frequency of the mining machine chip (real-time effective, restart mining machine failure).
format	privilege|0,setfreq, <freq1>,< freq2>,< freq3>,< freq4>
Parameter	<freq1> point 1 frequency, in MHz. < freq2> point 2 frequency, in MHz. < freq3> point 3 frequency, in MHz. < freq4> point 4 frequency, in MHz. Note: The frequency parameter must be selected in the frequency list and any one of the frequency points cannot be 0. The frequencies must be in **ascending order**. See Appendix 1 for a list of frequencies.

Example（linux socat）	Set 4 frequency points of IP 192.168.189.135 miner to 500,525,550,575（MHz）: echo -n “privilege|0,setfreq,500,525,550,575“ | socat -t 30 stdio tcp:192.168.189.135:4028,shut-none && echo
Return value	STATUS=S,When=1865,Code=119,Msg=ASC 0 set OK,Description=cgminer 4.11.1|

3.4 Set voltage

Command	Configure the mine voltage (effective in real time, restart the mine machine failure). After calling this interface, the miner's voltage will be adjusted to the specified voltage until the next time the interface is called to change the voltage setting.
format	privilege|0,setvolt, <voltage>
Parameter	< voltage > set target voltage in **10mV**. Note: Minimum voltage is 12V,which is 12000mV,stepping is 40mV,Maximum voltage 13880mV.
Example（linux socat）	Set voltage of IP 192.168.189.135 miner to 12.6V（12600 mV）: echo -n “privilege|0,setvolt,1260“ | socat -t 30 stdio tcp:192.168.189.135:4028,shut-none && echo
Return value	STATUS=I,When=1498,Code=118,Msg=ASC 0 set info: **PS[0 1205 1381 166 2201 1260]**,Description=cgminer 4.11.1| PS[0 1205 1381 166 2201 1260]: The last field (here is 1260) is the goal of setting voltage. Please check this value to make sure it is the one you want. If not, please invoke this API to set it again.

3.5 Disable FAN

Command description	Disable miner fan(Real-time effective, restart mining machine failure)
Command format	privilege|0, disable_fan

Parameter description	None
Example （linux socat）	Disable FAN of IP 192.168.189.135 miner: echo -n “privilege|0,disable_fan“ | socat -t 30 stdio tcp:192.168.189.135:4028,shut-none && echo
Return value	STATUS=I,When=1456,Code=118,Msg=ASC 0 set info: **Fan cooling system disabled!**,Description=cgminer 4.11.1|

3.6 Enable FAN

description	Enable FAN(Real-time effective, restart mining machine failure)
format	privilege|0, enable_fan
Description	None
Example（linux socat）	Enable fan cooling system of IP 192.168.189.135 minier: echo -n “privilege|0,enable_fan“ | socat -t 30 stdio tcp:192.168.189.135:4028,shut-none && echo
Return value	STATUS=I,When=1428,Code=118,Msg=ASC 0 set info: **Fan cooling system enabled!**,Description=cgminer 4.11.1|

3.7 Set target temperature of miner chip

Command description	Configure the target operating temperature of the miner's chip (real-time effective, restart the mine machine failure). Once calling this interface, the target of the mine temperature is set to the specified value until the mine is restarted.
Command format	privilege|0,settemp, <temp>
Parameter description	< temp> Sets the target of temperature in degrees Celsius. Note: The maximum temperature is set to 105 ° C.
Example（linux socat）	Set target temperature of IP 192.168.189.135 to 105℃: echo -n “privilege|0,settemp,105“

	| socat -t 30 stdio tcp:192.168.189.135:4028,shut-none && echo
Return value	STATUS=S,When=1400,Code=119,Msg=ASC 0 set OK,Description=cgminer 4.11.1|

3.8 Querying the status of the HASH board communication

description	Query the communication status of HASH board and controller board.
format	privilege|0,get_hash_comm_state
Parameter	None
Example （linux socat）	Query the status of mm board and HASH board communication of IP192.168.189.135 miner: echo -n “privilege|0,get_hash_comm_state“ | socat -t 30 stdio tcp:192.168.189.135:4028,shut-none && echo
Return value	STATUS=I,When=1289,Code=118,Msg=ASC 0 set info: Hash comm status:[0 OK! 1 OK!],Description=cgminer 4.11.1| 0 OK : HASH board 0 communication status. 1 OK : HASH board 1 communication status.

3.9 Save privileged configuration

description	Save the frequency/voltage/temperature target/fan enable configuration in the current privileged mode. The next time the miner is powered on, these configurations are automatically active.
format	privilege|0,savesettings
Parameter	None
Example（linux socat）	Save privileged configuration of IP 192.168.189.135 miner: echo -n “privilege|0,savesettings“

	| socat -t 30 stdio tcp:192.168.189.135:4028,shut-none && echo
Return value	STATUS=I,When=1713,Code=118,Msg=ASC 0 set info: **Save privilege arguments success!** ,Description=cgminer 4.11.1|

3.10 Get DH Value

description	Get DH value from miner. DH is the rate of calculation mistakes.
format	privilege|0, get_DH
Parameter	None
Example（linux socat）	Get IP 192.168.189.135 miner’s DH value: echo -n “privilege|0,get_DH” | socat -t 30 stdio tcp:192.168.189.135:4028,shut-none && echo
Return value	STATUS=I,When=1186,Code=118,Msg=ASC 0 set info: **DH[0.115% 0.138%]**,Description=cgminer 4.11.1| **0.115%** : HASH board 0 DH. **0.138%** : HASH board 1 DH.

3.11 Get GHSmm Value

description	Get GHSmm value from miner.
format	privilege|0, get_GHSmm
Parameter	None
Example（linux socat）	Get IP 192.168.189.135 miner’s GHSmm value: echo -n “privilege|0, get_GHSmm” | socat -t 30 stdio tcp:192.168.189.135:4028,shut-none && echo
Return value	STATUS=I,When=508,Code=118,Msg=ASC 0 set info: **GHSmm[18703.54 18706.64]**,Description=cgminer 4.11.1|

	18703.54 : HASH board 0 HASH rate in GHS. 18706.64 : HASH board 1 HASH rate in GHS. Note: To get the expected HASH rate, use the formula GHSmm*(1-DH).

3.12 Get temperature status

description	Get temperature status from miner.
format	privilege|0, get_temp_status
Parameter	None
Example（linux socat）	Get IP 192.168.189.135 miner’s temp status: echo -n “privilege|0,get_temp_status” | socat -t 30 stdio tcp:192.168.189.135:4028,shut-none && echo
Return value	STATUS=I,When=1648,Code=118,Msg=ASC 0 set info: **TMax[66] Tmin[50] TAvg[58]**,Description=cgminer 4.11.1| TMax: Max chip temperature. Tmin: Min chip temperature. TAvg: Average temperature of all chips

3.13 Get fan status

description	Get fans status from miner.
format	privilege|0, get_fan_status
Parameter	None
Example（linux socat）	Get IP 192.168.189.135 miner’s temp status: echo -n “privilege|0,get_fan_status” | socat -t 30 stdio tcp:192.168.189.135:4028,shut-none && echo
Return value	STATUS=I,When=1668,Code=118,Msg=ASC 0 set info: **Fan1[3368] Fan2[3368] FanR[50%]**,Description=cgminer 4.11.1|

	Fan1: The first fan speed in RPM. Fan2: The second fan speed in RPM. FanR: Current fan speed percentage.

4 General API description

The general API includes setting the mining machine to the IDLE state, querying state of the mining machine, voltage, temperature, DH value, mine pool information, and HASH rate information of all the chips. Please refer to:
https://github.com/Canaan-Creative/avalon10-docs

5 APPENDIX 1 Frequency list :

	25	300	312	325	337	350	362	375	387	400	408	412	416
425	433	437	441	450	458	462	466	475	483	487	491	500	508
512	516	525	533	537	550	562	575	587	600	612	625	637	650
662	675	687	700	712	725	737	750	762	775	787	800		