LIBSOPHON User Guide

Release 0.4.13

SOPHGO

Oct 27, 2025

contents

CHAPTER 1

Disclaimer

Legal Disclaimer

Copyright © SOPHGO 2022. All rights reserved.
No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of SOPHGO .

Notice

The purchased products, services and features are stipulated by the contract made between SOPHGO and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied. The information in

this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Technical Support

Address
Floor 6, Building 1, Yard 9, FengHao East Road, Haidian District, Beijing, 100094, China

Website
https://www.sophgo.com/

Email
sales@sophgo.com

Phone
+86-10-57590723 +86-10-57590724

CHAPTER 2

Release note

version	release date	description
V0.1.0	2022.07.12	First release, including bmlib, bm-smi and tpu runtime.
V0.2.0	2022.07.30	Added bmvid and supplemented document.
V0.3.0	2022.08.30	Added soc mode support and bmcv support. Supported bm1684.
V0.4.0	2022.09.15	Improved bm1684 support, added cross-compilation guide of soc mode and supported SC7 accelerator card.
V0.4.1	2022.09.21	Improved soc mode support of bm1684 and fixed some opencv bug.

CHAPTER 3

Install libsophon

The libsophon offers different types of installations on different Linux versions. Please select the corresponding method according to your system. Do not mix different installations on the same machine. Depending on the current situation, the installed version may change. "0.4.1" in the following description is only an example.

Please uninstall the old BM1684 SDK driver first:

```
#Enter the SDK installation directory and execute:
sudo ./remove_driver_pcie.sh
#or:
sudo rmmod bmsophon
sudo rm -f /lib/modules/$(uname -r)/kernel/drivers/pci/bmsophon.ko
```

If the Debian/Ubuntu system is used:

the installation package consists of three files:

- sophon-driver_0.4.1_amd64.deb
- sophon-libsophon_0.4.1_amd64.deb
- sophon-libsophon-dev_0.4.1_amd64.deb

Where: sophon-driver contains the PCIe accelerator card driver; sophon-libsophon contains the runtime environment (library files, tools, etc.); sophon-libsophon-dev contains the development environment (header files, etc.). If the system is only installed on a deployment environment, the sophon-libsopho-dev does not need to be installed.

The installation may be done by following steps:

```
#To install the dependency library, execute the following commands only once:
sudo apt install dkms libncurses5
#Install libsophon:
sudo dpkg -i sophon-*.deb
#Bm-smi can be used after executing the following command in the terminal,
#or after logging out and then logging in to the current user:
source /etc/profile
```

Installation position:

```
/opt/sophon/
├─── driver-0.4.1
├─── libsophon-0.4.1
│    ├─── bin
│    ├─── data
│    ├─── include
│    └─── lib
└─── libsophon-current -> /opt/sophon/libsophon-0.4.1
```

The deb package installation method does not allow you to install multiple different versions of the same package, but you may have placed several different versions under /opt/sophon in other ways. When using the deb package for installation, /opt/sophon/libsophon-current will point to the last version installed. After uninstallation, it will point to the remaining latest version (if any).

When using the deb package to install, if the following prompt message appears:

```
modprobe: FATAL: Module bmsophon is in use.
```

You need to manually execute the following command to install the driver in the new deb package after manually stopping the program that is using the driver.

```
sudo modprobe -r bmsophon
sudo modprobe bmsophon
```

Uninstallation method:

```
sudo apt remove sophon-driver sophon-libsophon
#or:
sudo dpkg -r sophon-driver
sudo dpkg -r sophon-libsophon-dev
sudo dpkg -r sophon-libsophon
```

If there is any trouble in uninstallation, you can try the following operations:

```
#Uninstall the driver manually:
dkms status
#Check the output, usually as follows:
bmsophon, 0.4.1, 5.15.0-41-generic, x86_64: installed
#Remember the first two fields and apply them to the following command:
```

(continues on next page)

(continued from previous page)

```
sudo dkms remove -m bmsophon -v 0.4.1 --all
#Then uninstall the driver again:
sudo apt remove sophon-driver
sudo dpkg --purge sophon-driver
#Completely clear libsophon:
sudo dpkg --purge sophon-libsophon
```

If another Linux system is used:

the installation package consist of one file:

- libsophon_0.4.1_x86_64.tar.gz

The installation may be done by following steps:

```
tar -xzvf libsophon_0.4.1_x86_64.tar.gz
sudo cp -r libsophon_0.4.1_x86_64/* /
sudo ln -s /opt/sophon/libsophon-0.4.1 /opt/sophon/libsophon-current
```

Next, please build the driver compilation environment according to the requirements of the Linux Releases you are using, and then do the following operations:

```
sudo ln -s /opt/sophon/driver-0.4.1/$bin /lib/firmware/bm1684x_firmware.bin
sudo ln -s /opt/sophon/driver-0.4.1/$bin /lib/firmware/bm1684_ddr_firmware.bin
sudo ln -s /opt/sophon/driver-0.4.1/$bin /lib/firmware/bm1684_tcm_firmware.bin
cd /opt/sophon/driver-0.4.1
```

Here "$bin" is the full name of the bin file with version number, for bm1684x, e.g. **bm1684x.bin_v3.1.0-9734c1da-220802**, for bm1684, e.g. **bm1684_ddr.bin_v3.1.1-63a8614d-220906** and **bm1684_tcm.bin_v3.1.1-63a8614d-220906**.

After that you can compile the driver (here it does not depend on dkms):

```
sudo make SOC_MODE=0 PLATFORM=asic SYNC_API_INT_MODE=1 \
      TARGET_PROJECT=sg_x86_pcie_device FW_SIMPLE=0 \
      PCIE_MODE_ENABLE_CPU=1
sudo cp ./bmsophon.ko /lib/modules/$(uname -r)/kernel/
sudo depmod
sudo modprobe bmsophon
```

Finally, some configuration work should be done:

```
# Add library and executable file path:
sudo cp /opt/sophon/libsophon-current/data/libsophon.conf /etc/ld.so.conf.d/
sudo ldconfig
sudo cp /opt/sophon/libsophon-current/data/libsophon-bin-path.sh /etc/profile.d/
source /etc/profile
# Add cmake config files:
sudo mkdir -p /usr/lib/cmake/libsophon
sudo cp /opt/sophon/libsophon-current/data/libsophon-config.cmake /usr/lib/cmake/
↪libsophon/
```

Copyright © SOPHGO

Uninstallation method:

```
sudo rm -f /etc/ld.so.conf.d/libsophon.conf
sudo ldconfig
sudo rm -f /etc/profile.d/libsophon-bin-path.sh
sudo rm -rf /usr/lib/cmake/libsophon
sudo rmmod bmsophon
sudo rm -f /lib/modules/$(uname -r)/kernel/bmsophon.ko
sudo depmod
sudo rm -f /lib/firmware/bm1684x_firmware.bin
sudo rm -f /lib/firmware/bm1684_ddr_firmware.bin
sudo rm -f /lib/firmware/bm1684_tcm_firmware.bin
sudo rm -f /opt/sophon/libsophon-current
sudo rm -rf /opt/sophon/libsophon-0.4.1
sudo rm -rf /opt/sophon/driver-0.4.1
```

Copyright © SOPHGO

CHAPTER 4

Development using libsophon

4.1 PCIE MODE

After libsophon is installed, it is recommended to use cmake to link the library in libsophon to your own program. You can add the following paragraphs in CMakeLists.txt of your own program:

```
find_package(libsophon REQUIRED)
include_directories(${LIBSOPHON_INCLUDE_DIRS})

add_executable(${YOUR_TARGET_NAME} ${YOUR_SOURCE_FILES})

target_link_libraries(${YOUR_TARGET_NAME} ${the_libbmlib.so})
#The ${the_libbmlib.so} above means that the libbmlib.so library should be linked, and other␣
↪libraries are the same.
```

In your code you can call functions in libbmlib:

```
#include <bmlib_runtime.h>
#include <stdio.h>

int main(int argc, char const *argv[])
{
  bm_status_t ret;
  unsigned int card_num = 0;

  ret = bm_get_card_num(&card_num);
  printf("card number: %d/%d\n", ret, card_num);
```

(continues on next page)

(continued from previous page)

```
  return 0;
}
```

4.2 SOC MODE

4.2.1 Compiling program on the SOC board

If you want to develop in Linux environment running on a SOC-mode board, you need to install the sophon-soc-libsophon-dev_0.4.1_arm64.deb toolkit, using the following command:

```
sudo dpkg -i sophon-soc-libsophon-dev_0.4.1_arm64.deb
```

After installation, you can refer to the PCIE MODE development method and use cmake to link the libraries in libsophon to your own program.

4.2.2 x86 cross-compiling programs

If you want to build cross-compiling environment using the SOPHON SDK, you will need the gcc-aarch64-linux-gnu toolchain and libsophon_soc_0.4.1_aarch64.tar.gz in libsophon relesae package. Install the gcc-aarch64-linux-gnu toolchain with the following command:

```
sudo apt-get install gcc-aarch64-linux-gnu g++-aarch64-linux-gnu
```

Next, extract libsophon_0.4.1_aarch64.tar.gz from the libsophon relesae package.

```
1. mkdir -p soc-sdk
2. tar -zxf libsophon_soc_0.4.1_aarch64.tar.gz
3. cp -rf libsophon_soc_0.4.1_aarch64/opt/sophon/libsophon-0.4.1/lib  soc-sdk
4. cp -rf libsophon_soc_0.4.1_aarch64/opt/sophon/libsophon-0.4.1/include  soc-sdk
```

Libsophon contains bmrt, bmlib and bmcv libraries.

Opencv and ffmpeg libraries are in sophon-mw-soc_0.4.1_aarch64.tar.gz, which the directory structure is similar to libsophon, just copy all the contents in lib and include to soc-sdk.

If you need to use third-party libraries, you can use qemu to build a virtual environment on x86 and then copy the header and library files into the soc-sdk directory with the following command:

```
# Build qemu virtual environment:

sudo apt-get install qemu-user-static
mkdir rootfs
cd rootfs

# Build Ubuntu 20.04 roofs:

sudo qemu-debootstrap --arch=arm64 focal .
sudo chroot . qemu-aarch64-static /bin/bash

# After entering qemu, install the software.
# gflag are used as an example:

apt-get update
apt-get install libgflags-dev

# The installed ".so" files are usually under /usr/lib/aarch64-linux-gnu/libgflag*, just copy␣
↪them to soc-sdk/lib.
```

Following code is an example of how to use the soc-sdk cross-compilation produced by SOPHON SDK in your code, to call functions in libbmlib:

```
#include <bmlib_runtime.h>
#include <stdio.h>

int main(int argc, char const *argv[])
{
  bm_status_t ret;
  unsigned int card_num = 0;

  ret = bm_get_card_num(&card_num);
  printf("card number: %d/%d\n", ret, card_num);

  return 0;
}
```

First create a new working directory as follows:

```
mkdir -p workspace && pushd workspace
touch CMakeLists.txt
touch get_dev_count.cpp
```

Import the above c++ code into get_dev_count.cpp and add the following paragraph to CMakeLists.txt:

```
cmake_minimum_required(VERSION 2.8)

set(TARGET_NAME "test_bmlib")

project(${TARGET_NAME} C CXX)
```

(continues on next page)

(continued from previous page)

```
set(CMAKE_C_COMPILER aarch64-linux-gnu-gcc)
set(CMAKE_ASM_COMPILER aarch64-linux-gnu-gcc)
set(CMAKE_CXX_COMPILER aarch64-linux-gnu-g++)

# This demo linked to bmlib library, so opened BM_LIBS.
set(BM_LIBS bmlib bmrt)
# You can open JPU_LIBS if you need to link jpu related libraries.
# set(JPU_LIBS bmjpuapi bmjpulite)
# You can open OPENCV_LIBS if you need to link opencv related libraries.
# set(OPENCV_LIBS opencv_imgproc opencv_core opencv_highgui opencv_imgcodecs␣
↪opencv_videoio)
# To link external libraries, see the following method for importing gflags, and open EXTRA_
↪LIBS.
# set(EXTRA_LIBS gflags)

include_directories("${SDK}/include/")

link_directories("${SDK}/lib/")

set(src get_dev_count.cpp)
get_filename_component(target ${src} NAME_WE)
add_executable(${target} ${src})
target_link_libraries(${target} ${BM_LIBS})
# Libraries such as OPENCV and FFMPEG are not used, so there is no need to add the␣
↪following link path.
# target_link_libraries(${target} ${BM_LIBS} ${OPENCV_LIBS} ${FFMPEG_LIBS} ${JPU_LIBS} $
↪{EXTRA_LIBS} pthread)
```

Then build program using cmake:

```
mkdir -p build && pushd build
cmake -DSDK=/path_to_sdk/soc-sdk ..
make
```

In this way, you can compile an aarch64 architecture program running in soc mode on an x86 machine.

the above example only links to the bmlib library, other libraries such as opencv, ffmpeg, and other libs are the same.

CHAPTER 5

Use of libsophon tools

There are several tools in libsophon, and they will be introduced below.

5.1 Operation instructions for bm-smi

5.1.1 Interpretation of terms

Term	Interpretation
BM1684	The third-generation tensor processor launched by SOPHGO for the field of deep learning.
BM1684X	The fourth-generation tensor processor launched by SOPHGO for the field of deep learning.
TPU	On-chip neural network processing unit.
SOC mode	A working mode, the SDK runs on A53 AARCH64 platform, and TPU is used as the platform bus device.
PCIe mode	A working mode, SDK runs on the X86 platform, BM1684 and BM1684X are at deep learning computing accelerator cards in PCIe interface.
Drivers	Drivers are the channels through which the API interface accesses the hardware.
Gmem	DDR memory on card for TPU acceleration.
F	FAULT, fault status.
N/A	This parameter is not supported.

5.1.2 Introduction of bm-smi

The bm-smi tool displays device status information in the form of interface or text, such as device temperature, fan speed; it can also enable, disable or set some device functions, such as led and ecc.

Bm-smi functions mainly include:

1) Viewing device parameters and runtime status.
    - Viewing device working mode(PCIe/SOC).
    - Viewing physical board ID.
    - Viewing the device chip ID, and the PCIe bus ID where it is located.
    - Viewing device temperature and power consumption.
    - Viewing whether the device ECC is enabled and the number of corrections.
    - Viewing the total number and usage of gmem.
    - Viewing the usage of tpu.
    - Viewing the work frequency information of the device.
    - Viewing the size of gmem occupied by each process during running.
    - Viewing device fan status.
2) Modifying board parameters.
    - Disable or enable ECC.
    - Turn on or turn off led indicator light on the board.
3) Performing the recovery operation on the fault device.

    The following table lists the device information that bm-smi can obtain and the support information in PCIe and SOC modes:

Device Information	PCIe Mode	SOC Mode
Displaying time and date	Supported	Supported
SDK Version	Supported	Supported
Driver Version	Supported	Supported
Physical board id number	Supported	Supported
Tpu device number	Supported	Supported
Board name	Supported	Supported
Board status	Supported	Supported
Board temperature	Supported	Not supported
Chip temperature	Supported	Not supported
Board power consumption	Supported	Not supported
Module power consumption	Supported	Not supported
Module voltage	Supported	Not supported
Supply current	Supported	Not supported
Whether DDR ECC is enabled	Supported	Not supported
DDR enabled, number of correction times	Supported	Not supported
Board serial number	Supported	Not supported
domain:b:d.f in PCIe mode	Supported	Not supported
PCIe or SOC mode	Supported	Supported
Minimum work frequency	Supported	Supported
Maximum work frequency	Supported	Supported
Current work frequency	Supported	Supported
Maximum power consumption of board	Supported	Not supported
Working current of module	Supported	Not supported
Byte numbers of gmem totals and used	Supported	Supported
Instantaneous usage of tpu	Supported	Supported
Fan speed	Supported	Not supported
Byte numbers of gmem occupied by each process (or thread)	Supported	Not supported
Text mode	Supported	Not supported
Parameters	Supported	Only file, lms and loop are supported

Copyright © SOPHGO

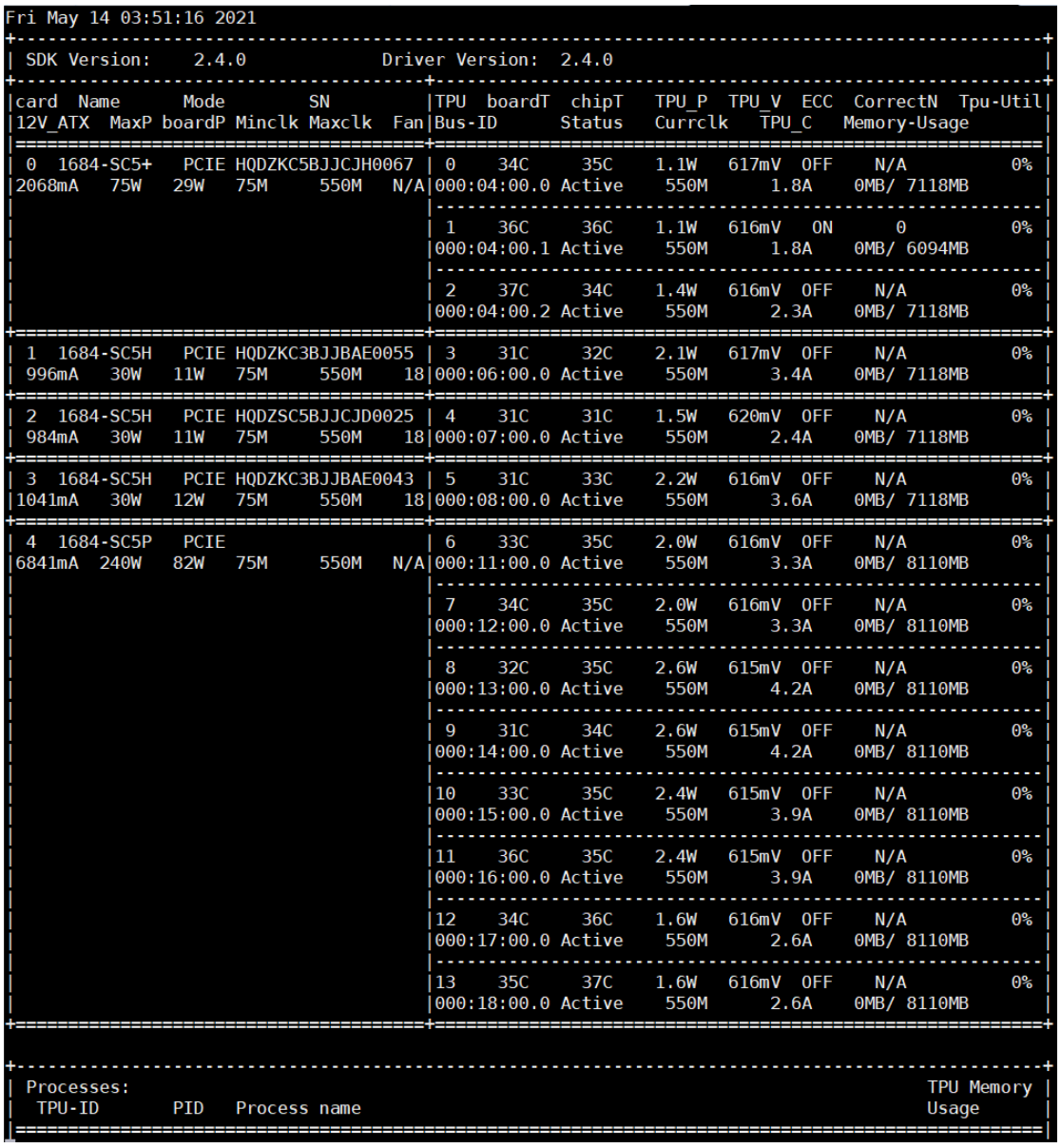


Figure 1 shows the display status of SC5+ (three-core)/SC5H/SC5P (eight-core), each card is separated by =======, the board attributes are displayed on the left, and the state of a single chip is displayed on the right and middle.

bm-smi is an executable file that does not depend on other dynamic libraries, and it is located under /opt/sophon/libsophon-current/bin directory. The above figure is a schematic diagram about the execution of bm-smi.

Copyright © SOPHGO

5.1.3 Meanings of parameters

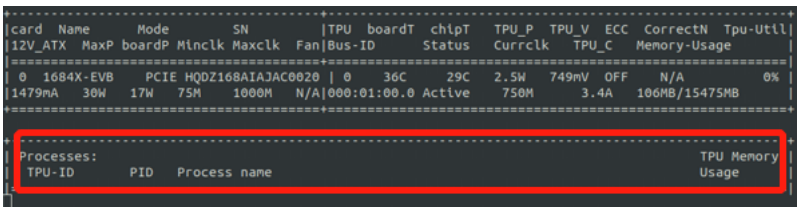
The meaning of each part is introduced one by one below:

- Fri Aug 7 14:18:57 2020: time and date when bm-smi is executed. This is just an example, and the actual execution may be different from what is shown here.
- SDK Version: 2.3.2: version number of sdk. This is just an example, and the actual execution may be different from what is shown here.
- Driver Version: 2.3.2: version number of the driver. This is just an example, and the actual execution may be different from what is shown here.
- Card: physical board id number.
- Name: board name.
- Mode: PCIe or SOC mode.
- SN: board serial number (total of 17 bits).
- TPU: device number of tpu.
- BoardT: board temperature.
- chipT: chip temperature.
- TPU_P: power consumption of TPU module.
- TPU_V: voltage of TPU module.
- ECC: whether DDR ECC is enabled.
- CorrectNum: the number of correction times if DDR is enabled.
- Tpu-Util: instantaneous usage of tpu.
- 12V_ATX: 12V board supply current.
- MaxP: maximum power consumption of board.
- boardP: board power consumption.
- Minclk: minimum work frequency of tpu.
- Maxclk: maximum work frequency of tpu.
- Fan: fan speed, N/A means the card has no fan, and F means there is a failure in the fan.
- Bus-ID: domain:b:d.f in PCIe mode.
- Status: board status. Active means active status; and Fault means fault status.
- Curclk: current work frequency of tpu. The color of the displayed value varies according to the current work frequency. 550M (bm1684) or 1000M (bm1684x) is displayed in white, 75M in red, and other frequencies in yellow; red and yellow are used to indicate to the user that the current work frequency is not the maximum

work frequency. Displaying different colors are only available in version 2.1.0 and above.

- TPU_C: work current of tpu module.
- Memory-Usage: byte numbers of gmem totals and used. The 106M indicates the memory size of the VPU firmware by default. The memory on the board may be distributed in different address spaces. All the memory we allocate is continuous address, and because of the different size of each allocation, it will lead to fragmentation of the memory, so the usage may not reach 100%.

The figure below shows the byte numbers of gmem occupied by each process (or thread) of each device:



Notices:

1. Because our board can be used by multiple users for multiple tasks at the same time, so theoretically, there can be unlimited processes creating unlimited handles to apply for global memory. You can use the up and down arrow keys and the page up and down keys to view the information of gmem occupied by all processes, save them as files through tag, which also contains all process information.
2. For the gmem information occupied by the process, each line displays the gmem corresponding to a handle created by the process. If the process creates multiple handles, the gmem information occupied by each handle is displayed on a separate line.

5.1.4 Specific operation methods and parameters

The parameters supported by bm-smi include:

- dev (which dev is selected to query, 0xff is for all.) type: int32, default: 255

  Used to select the parameters of which device to query or modify, and all devices are selected by default.

  This function is not supported in SOC mode.

- ecc (ECC on DDR is on or off)

  type: string default: ""

  Used to configure Enable and Disable of DDR ECC, and the example is as follows:

```
bm-smi --dev=0x0 --ecc=on

bm-smi --dev=0x0 --ecc=off
```

When executing this command, do not allow any process to use this device. After setting, restart the host to make it valid.

When executing this command, please do not use it with other parameters, for example:

```
bm-smi --dev=0x0 --ecc=on --file=~/a.txt
# the --file=~/a.txt may be ignored. This command can only execute ecc-related␣
↪actions.
```

If the dev parameter is not specified, the operation will be performed on all devices by default.

This function is not supported in SOC mode.

- file (target file to save smi log)

```
type: string default: ""
```

The device status can be redirected to a text with the following method:

```
bm-smi --dev=0x0 --file=./bm-smi.log
```

This function is supported in SOC mode.

- led (pcie card LED status: on/off/blink)

```
type: string default: "on"
```

Used to configure on and off of the board LED, and the examples are as below:

```
bm-smi --dev=0x0 --led=on

bm-smi --dev=0x0 --led=off
```

Note: This function support on/off/blink on SC5+ and SC5P, on/off on SC5H, does not support other board types. For the SC5+ board, only the first chip can control the status of LED. SC5P has 8 LEDs, each device corresponds to one LED, and each LED supports setting status separately.

This function is not supported in SOC mode.

- lms (sample interval in loop mode.) type: int32 default: 500

  Used to set the time interval for querying the device status when running bm-smi, the default is 500ms, and the minimum value of this parameter is 300ms. This function is supported in SOC mode.

- loop (true is for loop mode, false is for only once mode.) type: bool, default: true

  Used to set single mode or cycle mode when running bm-smi, and the cycle mode is set by default. In single mode, bm-smi exits after querying the device status once; in cycle mode, it queries the device status repeatedly according to lms. Examples are as follows:

```
bm-smi --loop

bm-smi --noloop
```

  This function is supported in SOC mode.

- recovery, the use method is as follows: when x function of a certain device is found to be faulty, the user removes all services from this card, reaching to a state where no upper-layer services and applications use this card, and the following is executed:

```
bm-smi --dev=0x(0/1/2/3…..) --recovery
```

  The three-core card SC5+ and the eight-core card SC5P only support the recovery of the entire card. Recoverying any device on the card can led to the entire card recoveried, so you need to stop the tasks on the entire card during recovery.

  Note: Do not perform this operation when the board card is working normally. Some servers do not support this function, and executing this function will cause the server to restart. Currently known servers that do not support this function are Dell R740, Dell R940, Inspur 5468 and Sugon X785-G30.

  This function is not supported in SOC mode.

- opmode and opval, operation method is: select the mode and mode value of bm-smi execution, compatible with the previous tags, for example:

```
bm-smi  --opmode=display
#bm-smi have the same effects.

bm-smi --opmode=ecc --opval=on
#bm-smi --ecc=on have the same effects. And so on for other tags.
```

  At present, opmode has four operation modes: display, ecc (enable), led (indicator), and recovery. Subsequent new functions will be used in this way. In order to take care of users which have the operating habits of the old version , the operation method of the old version is still supported in the new version. (Note: At present, only when the opmode is ecc or led, you need to use it with opval to assign values).

  We added 2.5.0 display mode with the memory monitor display for heap and vpu, and the operation method is:

```
bm-smi --opmode=display_memory_detail
```

```
Mon Sep 13 10:47:45 2021
+-----------------------------------------------------------------------------------------------------+
| SDK Version:    2.5.0            Driver Version:  2.5.0                                             |
+-------------------------------------+---------------------------------------------------------------+
|card  Name      Mode        SN       |TPU  boardT  chipT   TPU_P  TPU_V  ECC  CorrectN  Tpu-Util|
|12V_ATX  MaxP boardP Minclk Maxclk  Fan|Bus-ID      Status   Currclk   TPU_C   Memory-Usage       |
|                                     |     Heap0         Heap1        Heap2     Vpu-Memory-Usage|
|=====================================+===============================================================|
| 0  1684-SC5+   PCIE HQDZKC5AIABAH0001 | 0    53C     52C    1.8W   615mV   ON       0          0% |
|2516mA   75W   34W   75M     550M   N/A|000:01:00.0 Active    550M      2.9A  170MB/ 9646MB       |
|                                     |   0MB/3503MB    0MB/2559MB    0MB/32MB    170MB/ 3552MB  |
|                                     |---------------------------------------------------------------|
|                                     | 1    53C     52C    1.8W   615mV  OFF    N/A          0% |
|                                     |000:01:00.1 Active    550M      3.0A  170MB/11182MB       |
|                                     |   0MB/4015MB    0MB/3071MB    0MB/32MB    170MB/ 4064MB  |
|                                     |---------------------------------------------------------------|
|                                     | 2    52C     52C    2.0W   614mV  OFF    N/A          0% |
|                                     |000:01:00.2 Active    550M      3.2A  170MB/11182MB       |
|                                     |   0MB/4015MB    0MB/3071MB    0MB/32MB    170MB/ 4064MB  |
+=====================================+===============================================================+

+-----------------------------------------------------------------------------------------------------+
| Processes:                                                                                TPU Memory |
|  TPU-ID       PID   Process name                                                          Usage      |
|=====================================================================================================|
```

Meanwhile, the usage display of vpu and jpu is also added, and the operation method is as follows:

```
bm-smi --opmode=display_util_detail
```

```
Mon Sep 13 10:49:10 2021
+-----------------------------------------------------------------------------------------------------+
| SDK Version:    2.5.0            Driver Version:  2.5.0                                             |
+-------------------------------------+---------------------------------------------------------------+
|card  Name      Mode        SN       |TPU  boardT  chipT   TPU_P  TPU_V  ECC  CorrectN  Tpu-Util|
|12V_ATX  MaxP boardP Minclk Maxclk  Fan|Bus-ID      Status   Currclk   TPU_C   Memory-Usage       |
|                                     |Vpu-Dec-Util       Vpu-Enc-Util       Jpu-Core-Util      |
|=====================================+===============================================================|
| 0  1684-SC5+   PCIE HQDZKC5AIABAH0001 | 0    53C     52C    1.8W   615mV   ON       0          0% |
|2563mA   75W   35W   75M     550M   N/A|000:01:00.0 Active    550M      2.9A    0MB/ 6094MB       |
|                                     |     0%                0%                 0%               |
|                                     |---------------------------------------------------------------|
|                                     | 1    53C     52C    1.8W   615mV  OFF    N/A          0% |
|                                     |000:01:00.1 Active    550M      3.0A    0MB/ 7118MB       |
|                                     |     0%                0%                 0%               |
|                                     |---------------------------------------------------------------|
|                                     | 2    52C     52C    1.9W   614mV  OFF    N/A          0% |
|                                     |000:01:00.2 Active    550M      3.1A    0MB/ 7118MB       |
|                                     |     0%                0%                 0%               |
+=====================================+===============================================================+
```

5.1.5 Introduction of text mode

The output of bm-smi is a simple graphical interface, which describes the status of the board. In order to meet the needs of some users for text information (easy to use scripts to parse some parameters), it supports text mode (SOC mode does not support text mode), and the operation method is as follows:

```
bm-smi --start_dev=0 --last_dev=2 --text_format

1684-SC5+ PCIE chip0: 0 000:01:00.0 Active 56C 55C 2W 615mV OFF N/A 0% 75M 550M 550M 3.
↪3A 0MB 7086MB
1684-SC5+ PCIE chip1: 1 000:01:00.1 Active 56C 55C 2W 613mV OFF N/A 0% 75M 550M 550M 4.
```

(continues on next page)

(continued from previous page)

```
↪1A 0MB 7086MB
1684-SC5+ PCIE chip2: 2 000:01:00.2 Active 54C 53C 1W 615mV OFF N/A 0% 75M 550M 550M 2.
↪6A 0MB 7086MB
```

A line of text information output by the above command divided into three areas:

First area:

```
1684-SC5+ PCIE chip0: 0 000:01:00.0 Active 56C 55C 2W 615mV OFF N/A 0% 75M 550M 550M 3.
↪3A 0MB 7086MB
```

Status of the 0th chip on the three-core card, 1684-SC5+ PCIE chip0:

```
TPU Bus-ID Status boardT chipT TPU_P TPU_V ECC CorrectN Tpu-Util Minclk Maxclk␣
↪Curclk TPU_C Memory-Usage
```

Second area:

```
1684-SC5+ PCIE chip1: 1 000:01:00.1 Active 56C 55C 2W 613mV OFF N/A 0% 75M 550M 550M 4.
↪2A 0MB 7086MB
```

Status of the 1st chip on the three-chip card, 1684-SC5+ PCIE chip1:

The following information corresponds in sequence to parameters in bm-smi:

```
TPU Bus-ID Status boardT chipT TPU_P TPU_V ECC CorrectN Tpu-Util Minclk Maxclk␣
↪Curclk TPU_C Memory-Usage
```

Third area:

```
1684-SC5+ PCIE chip2: 2 000:01:00.2 Active 54C 53C 1W 615mV OFF N/A 0% 75M 550M 550M 2.
↪6A 0MB 7086MB
```

Status of the 2nd chip on the three-chip card, 1684-SC5+ PCIE chip2:

The following information corresponds in sequence to parameters in bm-smi:

```
TPU Bus-ID Status boardT chipT TPU_P TPU_V ECC CorrectN Tpu-Util Minclk Maxclk␣
↪Curclk TPU_C Memory-Usage
```

Notes:

1. --start_dev=0 --last_dev=2 indicates the device numbers corresponding to the 0th chip **and**␣ ↪the last chip of a certain card displayed **in** bm-smi;

2. --start_dev --last_dev --text_format should be used together.

Copyright © SOPHGO

5.1.6 help information of bm-smi :

5.1.6.1 help information of bm-smi under PCIe mode

```
bm-smi --help

bm-smi: command line brew

usage: bm-smi [--ecc=on/off] [--file=/xx/yy.txt] [--dev=0/1...][--start_dev=x] [--last_dev=y] [--
↪text_format] [--lms=500] [--recovery] [-loop] [--led=on/off/blink]

ecc:

set ecc status, default is off

file:

the target file to save smi log, default is empty.

dev:

which device to be selected to query, default is all.

start_dev:

the first device to be selected to query, must chip0 of one card, default is invalid.

last_dev:

the last device to be selected to query, default is invalid.

lms:

how many ms of the sample interval, default is 500.

loop:

if -loop (default): smi sample device every lms ms.

if -noloop: smi sample device only once.

recovery:

recovery dev from fault to active status.

text_format:

if true only display attr value from start_dev to last_dev.

led:
```

(continues on next page)

(continued from previous page)

```
pcie card LED status: on/off/blink.

New usage: bm-smi [--opmode=display/ecc/led/recovery][--opval=on/off/...] [--file=/xx/yy.txt][--
↪dev=0/1...] [--start_dev=x] [--last_dev=y] [--text_format][--lms=500] [-loop]

opmode(default null):

choose different mode,example:display, ecc, led, recovery

display: means open bm-smi window and check info, use like ./bm-smi

ecc: means enable or disable ecc, collocation opval=on/off

led: means modify led status, collocation opval=on/blink/off

recovery: means recovery dev from fault to active status.

opval(default null):

set mode value, use with opmode!

off: for led/ecc

on: for led/ecc

blink: for led

other flags have same usage, Both usage can be used!
```

bm-smi supports all parameters listed above in help in PCIe mode.

5.1.6.2 help information of bm-smi in SOC mode

```
bm-smi --help

bm-smi: command line brew

usage: bm-smi [--opmode=display] [--file=/xx/yy.txt] [--lms=500] [-loop]

opmode:

SOC mode only use display for bm-smi.

file:

the target file to save smi log, default is empty.

lms:
```

(continues on next page)

(continued from previous page)

```
how many ms of the sample interval, default is 500.

loop:

if -loop (default): smi sample device every lms ms.

if -noloop: smi sample device only once.
```

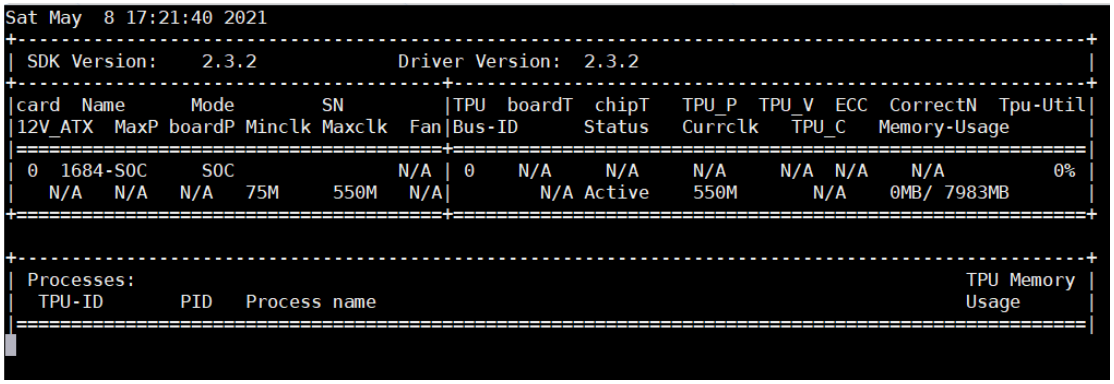
SOC mode only supports opmode=display, file, lms and loop parameters, and other parameters are invalid.

5.1.7 bm-smi used in SOC mode

In PCIe mode, bm-smi supports all the above functions; and in SOC mode, bm-smi interface supports the functions shown in Fig.2. N/A indicates this function is not supported; the supported parameters are only opmode=display, file, lms and loop.

In SOC mode, bm-smi operation method: run bm-smi directly after logging in soc.

```
bm-smi or bm-smi --opmode=display
```



5.2 Operation instructions for proc file system

5.2.1 Introduction of proc file system

The proc file system interface creates a device information node under the /proc node. Users can read the relevant nodes such as device temperature and version, through cat command or programming to obtain information.

Bm-smi focuses on displaying device information visually using interface, and the proc file system interface focuses on providing users with an interface for programming to

obtain device information. The table below lists the device information that can be obtained by the proc file system and its support information in PCIe and SOC modes:

Device Information	PCIe Mode	SOC Mode
card_num	Supported	Not supported
chip_num	Supported	Not supported
chip_num_on_card	Supported	Not supported
board_power	Supported	Not supported
board_temp	Supported	Not supported
chipid	Supported	Not supported
chip_temp	Supported	Not supported
dbdf	Supported	Not supported
dynfreq	Supported	Not supported
ecc	Supported	Not supported
maxboardp	Supported	Not supported
mode	Supported	Not supported
pcie_cap_speed	Supported	Not supported
pcie_cap_width	Supported	Not supported
pcie_link_speed	Supported	Not supported
pcie_link_width	Supported	Not supported
pcie_region	Supported	Not supported
tpuid	Supported	Not supported
tpu_maxclk	Supported	Not supported
tpu_minclk	Supported	Not supported
tpu_freq	Supported	Not supported
tpu_power	Supported	Not supported
firmware_info	Supported	Not supported
sn	Supported	Not supported
boot_loader_version	Supported	Not supported
board_type	Supported	Not supported
driver_version	Supported	Not supported
board_version	Supported	Not supported
mcu_version	Supported	Not supported
versions	Supported	Not supported
cdma_in_time	Supported	Not supported
cdma_in_counter	Supported	Not supported
cdma_out_time	Supported	Not supported
cdma_out_counter	Supported	Not supported
tpu_process_time	Supported	Not supported
completed_api_counter	Supported	Not supported
send_api_counter	Supported	Not supported
tpu_volt	Supported	Not supported
tpu_cur	Supported	Not supported
fan_speed	Supported	Not supported
media	Supported	Not supported

continues on next page

Table 5.1 – continued from previous page

a53_enable	Supported	Not supported
arm9_cache	Supported	Not supported
bmcpu_status	Supported	Not supported
bom_version	Supported	Not supported
boot_mode	Supported	Not supported
clk	Supported	Not supported
ddr_capacity	Supported	Not supported
dumpreg	Supported	Not supported
heap	Supported	Not supported
location	Supported	Not supported
pcb_version	Supported	Not supported
pmu_infos	Supported	Not supported
status	Supported	Not supported
vddc_power	Supported	Not supported
vddphy_power	Supported	Not supported
jpu	Not supported	Supported
vpu	Not supported	Supported

When the driver is installed, the system creates the /proc/bmsophon/card0…n directory according to the number of board, among which the card0 directory corresponds to the information of the 0th board card, the card1 directory corresponds to the information of the first card, and so on.

The /proc/bmsophon/cardn/bmsophon0…x directories are created in sequence under the board directory according to the number of devices on the current board, where x in bmsophonx corresponds to the device information with device id x of board n. For example, only SC5+ is inserted in the device, /proc/bmsophon/card0 will be generated accordingly after the driver is installed, and the bmsophon0/1/2 directory will be generated under the card0 directory to store the information of device 0/1/2, respectively.

In SOC mode, only JPU and VPU support the proc file system interface, and the nodes are /proc/jpuinfo and /proc/vpuinfo.

5.2.2 Meaning of parameters

In SOC mode, the parameters are only /proc/jpuinfo and /proc/vpuinfo; and PCIe mode, the directories and file nodes in the proc file system are structured as follows:

```
bitmain@weiqiao-MS-7B46:~/work/bm168x$ ls /proc/bmsophon/ -l

total 0

-r--r--r--.
1 root root 0 May 6 23:06 card_num
```

(continues on next page)

(continued from previous page)

```
-r--r--r--.
1 root root 0 May 6 23:06 chip_num

-r--r--r--.
1 root root 0 May 6 23:06 driver_version

dr-xr-xr-x 2 root root 0 May 6 13:46 card0 //there are the information of board 0 under the folder,
↪ as follows:

bitmain@weiqiao-MS-7B46:~/work/bm168x$ ls /proc/bmsophon/card0/ -l

total 0

-r--r--r--.
1 root root 0 May 6 23:06 board_power

-r--r--r--.
1 root root 0 May 6 23:06 board_temp

-r--r--r--.
1 root root 0 May 6 23:06 board_type

-r--r--r--.
1 root root 0 May 6 23:06 board_version

-r--r--r--.
1 root root 0 May 6 23:06 bom_version

-r--r--r--.
1 root root 0 May 6 23:06 chipid

-r--r--r--.
1 root root 0 May 6 23:06 chip_num_on_card

-rw-r--r--.
1 root root 0 May 6 23:06 fan_speed

-r--r--r--.
1 root root 0 May 6 23:06 maxboardp

-r--r--r--.
1 root root 0 May 6 23:06 mode

-r--r--r--.
1 root root 0 May 6 23:06 pcb_version

-r--r--r--.
1 root root 0 May 6 23:06 sn

-r--r--r--.
1 root root 0 May 6 23:06 tpu_maxclk
```

(continues on next page)

(continued from previous page)

```
-r--r--r--.
1 root root 0 May 6 23:06 tpu_minclk

-r--r--r--.
1 root root 0 May 6 23:06 versions

dr-xr-xr-x.
2 root root 0 May 6 23:06 bmsophon0//there are the information of board 0 under the folder,␣
↪as follows:

bitmain@weiqiao-MS-7B46:~/work/bm168x$ ls /proc/bmsophon/card0/bmsophon0 -l

total 0

-r--r--r--.
1 root root 0 May 6 23:11 a53_enable

-r--r--r--.
1 root root 0 May 6 23:11 arm9_cache

-r--r--r--.
1 root root 0 May 6 23:11 bmcpu_status

-r--r--r--.
1 root root 0 May 6 23:11 boot_loader_version

-r--r--r--.
1 root root 0 May 6 23:11 boot_mode

-r--r--r--.
1 root root 0 May 6 23:11 cdma_in_counter

-r--r--r--.
1 root root 0 May 6 23:11 cdma_in_time

-r--r--r--.
1 root root 0 May 6 23:11 cdma_out_counter

-r--r--r--.
1 root root 0 May 6 23:11 cdma_out_time

-r--r--r--.
1 root root 0 May 6 23:11 chip_temp

-r--r--r--.
1 root root 0 May 6 23:11 clk

-r--r--r--.
1 root root 0 May 6 23:11 completed_api_counter
```

(continues on next page)

(continued from previous page)

```
-r--r--r--.
1 root root 0 May 6 23:11 dbdf

-r--r--r--.
1 root root 0 May 6 23:11 ddr_capacity

-rw-r--r--.
1 root root 0 May 6 23:11 dumpreg

-rw-r--r--.
1 root root 0 May 6 23:11 dynfreq

-r--r--r--.
1 root root 0 May 6 23:11 ecc

-r--r--r--.
1 root root 0 May 6 23:11 heap

-rw-r--r--.
1 root root 0 May 6 23:11 jpu

-r--r--r--.
1 root root 0 May 6 23:11 location

-r--r--r--.
1 root root 0 May 6 23:11 mcu_version

-rw-r--r--.
1 root root 0 May 6 23:11 media

-r--r--r--.
1 root root 0 May 6 23:11 pcie_cap_speed

-r--r--r--.
1 root root 0 May 6 23:11 pcie_cap_width

-r--r--r--.
1 root root 0 May 6 23:11 pcie_link_speed

-r--r--r--.
1 root root 0 May 6 23:11 pcie_link_width

-r--r--r--.
1 root root 0 May 6 23:11 pcie_region

-r--r--r--.
1 root root 0 May 6 23:11 pmu_infos

-r--r--r--.
1 root root 0 May 6 23:11 sent_api_counter
```

(continues on next page)

(continued from previous page)

```
-r--r--r--.
1 root root 0 May 6 23:11 status

-r--r--r--.
1 root root 0 May 6 23:11 tpu_cur

-rw-r--r--.
1 root root 0 May 6 23:06 tpu_freq

-r--r--r--.
1 root root 0 May 6 23:11 tpuid

-r--r--r--.
1 root root 0 May 6 23:11 tpu_power

-r--r--r--.
1 root root 0 May 6 23:11 firmware_info

-r--r--r--.
1 root root 0 May 6 23:11 tpu_process_time

-rw-r--r--.
1 root root 0 May 6 23:11 tpu_volt

-rw-r--r--.
1 root root 0 May 6 23:11 vddc_power

-rw-r--r--.
1 root root 0 May 6 23:11 vddphy_power
```

Note: If the PCIe mode uses SC5P, the mcu_version will be created under the /proc/bmsophon/card/ board directory.

If other type of board is used, mcu_version will be created under the /proc/bmsophon/card/bmsophon/ device directory.

5.2.3 Meanings and Operation Methods of Parameters

5.2.3.1 Detailed Information of Devices in PCIe Mode

- card_num

  Readwrite property: read only.

  Meaning: number of system boards.

- chip_num

  Readwrite property: read only.

  Meaning: number of system devices.

- chip_num_on_card

  Readwrite property: read only.

  Meaning: number of devices on the corresponding board.
- board_power

  Readwrite property: read only.

  Meaning: board power consumption.
- board_temp

  Readwrite property: read only.

  Meaning: board temperature.
- chipid

  Readwrite property: read only.

  Meaning: chip id (0x1684x/0x1684/0x1682).
- chip_temp

  Readwrite property: read only.

  Meaning: chip temperature.
- dbdf

  Readwrite property: read only.

  Meaning: domain:bus:dev.function.
- dynfreq

  Readwrite property: read and write.

  Meaning: Enable or disable the dynamic tpu frequency modulation function; 0/1 is valid, other values are invalid.
- ecc

  Readwrite property: read only.

  Meaning: Enable or disable ECC function.
- maxboardp

  Readwrite property: read only.

  Meaning: maximum board power consumption.
- mode

  Readwrite property: read only.

  Meaning: work mode, PCIe/SOC.
- pcie_cap_speed

Readwrite property: read only.

Meaning: The maximum speed of PCIe supported by the device.

- pcie_cap_width

  Readwrite property: read only.

  Meaning: PCIe interface’ s maximum lane width supported by the device.

- pcie_link_speed

  Readwrite property: read only.

  Meaning: PCIe interface’ s speed of the device.

- pcie_link_width

  Readwrite property: read only.

  Meaning: lane width of PCIe interface of the device.

- pcie_region

  Readwrite property: read only.

  Meaning: size of PCIe bar.

- tpuid

  Readwrite property: read only.

  Meaning: tpu ID (0/1/2/3……).

- tpu_maxclk

  Readwrite property: read only.

  Meaning: maximum work frequency of tpu.

- tpu_minclk

  Readwrite property: read only.

  Meaning: minimum work frequency of tpu.

- tpu_freq

  Readwrite property: read and write.

  Meaning: work frequency of the tpu, which can be changed by writing parameters. 0 should be written into dynfreq to turn off the dynamic TPU frequency modulation before writing parameters.

- tpu_power

  Readwrite property: read only.

  Meaning: instantaneous power of tpu.

- firmware_info

Readwrite property: read only.

Meaning: version information of firmware, including commit id and compiling time.

- sn

  Readwrite property: read only.

  Meaning: number of board product.

- boot_loader_version

  Readwrite property: read only.

  Meaning: bootloader version number in spi flash.

- board_type

  Readwrite property: read only.

  Meaning: type of board.

- driver_version

  Readwrite property: read only.

  Meaning: version number of drive.

- board_version

  Readwrite property: read only.

  Meaning: version number of board hardware.

- mcu_version

  Readwrite property: read only.

  Meaning: version number of mcu software.

- versions

  Readwrite property: read only.

  Meaning: a collection of board software and hardware versions.

- cdma_in_time

  Readwrite property: read only.

  Meaning: the total time spent by cdma to move data from the host to the board.

- cdma_in_counter

  Readwrite property: read only.

  Meaning: the total number of times that cdma moves data from the host to the board.

- cdma_out_time

  Readwrite property: read only.

  Meaning: the total time spent by cdma to move data from the board to the host.

- cdma_out_counter

  Readwrite property: read only.

  Meaning: the total number of times that cdma moves data from the board to the host.

- tpu_process_time

  Readwrite property: read only.

  Meaning: the time spent in tpu processing.

- completed_api_counter

  Readwrite property: read only.

  Meaning: the number of times the api has been completed.

- send_api_counter

  Readwrite property: read only.

  Meaning: the number of times the api has been sent.

- tpu_volt

  Readwrite property: read and write.

  Meaning: tpu voltage, the voltage can be changed by writing a parameter.

- tpu_cur

  Readwrite property: read only.

  Meaning: tpu current.

- fan_speed

  Readwrite property: read only.

  Meaning: fan actual speed.

- media

  Readwrite property: read only.

  total_mem_size : total size of memory used by vpu and jpu.

  used_mem_size : memory being used by vpu and jpu.

  free_mem_size : free memory.

  id : number of vpu core.

link_num : number of encoding/decoding channels.

- a53_enable

  Readwrite property: read only.

  Meaning: a53 enable status.

- arm9_cache

  Readwrite property: read only.

  Meaning: cache enable status of arm9.

- bmcpu_status

  Readwrite property: read only.

  Meaning: bmcpu status.

- bom_version

  Readwrite property: read only.

  Meaning: bom version number.

- boot_mode

  Readwrite property: read only.

  Meaning: start mode.

- clk

  Readwrite property: read only.

  Meaning: clocks of modules.

- ddr_capacity

  Readwrite property: read only.

  Meaning: ddr capacity.

- dumpreg

  Readwrite property: read and write.

  Meaning: Dump register, input 1 is dumped to tpu register, input 2 is dumped to gdma register.

- heap

  Readwrite property: read only.

  Meaning: displaying the size of each heap.

- location

  Readwrite property: read only.

  Meaning: showing it is on which device currently.

- pcb_version

  Readwrite property: read only.

  Meaning: pcb version number.

- pmu_infos

  Readwrite property: read only.

  Meaning: More detailed current and voltage information.

- status

  Readwrite property: read only.

  Meaning: board status.

- vddc_power

  Readwrite property: read only.

  Meaning: vddc power.

- vddphy_power

  Readwrite property: read only.

  Meaning: vddphy power.

5.2.3.2 Detailed information of devices in SOC mode

In SOC mode, only JPU and VPU supports proc interface, and the corresponding proc nodes are /proc/jpuinfo and /proc/vpuinfo.

- jpuinfo

  Readwrite property: read only.

  JPU loadbalance : recording JPU0-JPU1(1684x),JPU0-JPU3(1684) encoding/decoding times, JPU* is JPEG encoder/decoder inside chip, value range: 0~2147483647

- vpuinfo

  Readwrite property: read only.

  id: vpu core number, value range: 0~2(1684x), 0~4(1684).

  link_num: number of encoding/decoding channels; value range: 0~32.

5.3 Operation Instructions for Tpu Drive Sysfs File System

5.3.1 Introduction of TPU Drive Sysfs File System

The sysfs file system interface is used to obtain information such as the usage of TPU. The table below lists the device information that can be obtained by TPU Drive Sysfs File System and the support information in PCIe and SOC modes:

Device Information	PCIe Mode	SOC Mode
npu_usage	Supported	Supported
npu_usage_enable	Supported	Supported
npu_usage_interval	Supported	Supported

After the driver is installed, some properties are created for each device to view and modify some parameters. these properties are located in :

PCIe mode: /sys/class/bm-sophon/bm-sophon0/device

SOC mode: /sys/class/bm-tpu/bm-tpu0/device

5.3.2 Meanings of Parameters

The meaning of each part is introduced below.

- npu_usage, the percentage of time TPU (NPU) is working over a period of time (window width)
- npu_usage_enable, npu_usage_enable, whether to enable statistics on NPU usage, enabled by default.
- npu_usage_interval, the time window width of statistics on NPU usage (in ms). The default is 500ms and the value range is [200,2000].

5.3.3 Specific use method of TPU Drive Sysfs File System interface

Examples are as follows:

Change the time window width (only under superuser):

```
root@bitmain:/sys/class/bm-sophon/bm-sophon0/device# cat npu_usage_interval

"interval": 600

root@bitmain:/sys/class/bm-sophon/bm-sophon0/device# echo 500 > npu_usage_interval

root@bitmain:/sys/class/bm-sophon/bm-sophon0/device# cat npu_usage_interval
```

(continues on next page)

(continued from previous page)

```
"interval": 500
```

Turn off the enable function of statistics on the usage of NPU:

```
root@bitmain:/sys/class/bm-sophon/bm-sophon0/device# cat npu_usage_enable
"enable": 1
root@bitmain:/sys/class/bm-sophon/bm-sophon0/device# echo 0 > npu_usage_enable
root@bitmain:/sys/class/bm-sophon/bm-sophon0/device# cat npu_usage_enable
"enable": 0
root@bitmain:/sys/class/bm-sophon/bm-sophon0/device# cat npu_usage
Please, set [Usage enable] to 1
root@bitmain:/sys/class/bm-sophon/bm-sophon0/device# echo 1 > npu_usage_enable
root@bitmain:/sys/class/bm-sophon/bm-sophon0/device# cat npu_usage_enable
"enable": 1
root@bitmain:/sys/class/bm-sophon/bm-sophon0/device# cat npu_usage
"usage": 0, "avusage": 0
root@bitmain:/sys/class/bm-sophon/bm-sophon0/device#
```

View the usage of NPU:

```
root@bitmain:/sys/class/bm-sophon/bm-sophon0/device# cat npu_usage
"usage": 0, "avusage": 0
```

Usage indicates the usage of NPU in a past time window.

Avusage indicates the usage of NPU since the drive was installed.

Copyright © SOPHGO

CHAPTER 6

Using test environment by Docker

Customers also can use docker to build the testing environment by following the next steps.

Before starting to build it, we recommand following the content of 安装libsophon to build it on the host firstly!

6.1 Build test environment on Docker

6.1.1 Install Docker

Please read the instructions on Docker official website and install docker step by step:

```
# uninstall docker
sudo apt-get remove docker docker.io containerd runc

# install dependencies
sudo apt-get update
sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release

# get the PIN
sudo mkdir -p /etc/apt/keyrings
curl -fsSL \
  https://download.docker.com/linux/ubuntu/gpg | \
```

(continues on next page)

(continued from previous page)

```
  gpg --dearmor -o docker.gpg && \
  sudo mv -f docker.gpg /etc/apt/keyrings/

# add docker libraries to system
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] \
  https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# install docker
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

6.1.2 Make the Dockerfile and create the docker image

6.1.2.1 Make the dockerfile

Using dockerfile to build the docker image with testing environment. The dockerfile is in the libsophon directory. Before building the image, please check whether the system time is accurate, as an incorrect system time can result in certificate verification failure during the process of updating the package list.

```
# convert the following path to the path where the actual libsophon is
cd path/to/libsophon
sudo docker build --platform linux/arm64/v8 -t image_name:image_version -f libsophon_
↪dockerfile .
```

You can named image_name and image_version as you wish.

After all the above work have been done, checking if the docker image created sucessfully by executing the command docker images:

```
$ sudo docker images
REPOSITORY  TAG           IMAGE ID    CREATED       SIZE
image_name    image_version  image_id      create_time      1.74GB
```

The image_name and image_version should be the same as what customer sets up. image_id and create_time should be when and what image customer uses.

Copyright © SOPHGO

6.1.2.2 Create the docker image

libsophon dynamic library is necessary to make a test in the docker. There are two ways to install it:

1. Install sophon-libsophon and sophon-libsophon-dev on host. Then map them into docker.

```
# sophon-libsophon and sophon-libsophon-dev are installed on host
sudo docker run \
  -td \
  --privileged=true \
  -v /opt/sophon:/opt/sophon \
  -v /etc/profile.d:/etc/profile.d \
  -v /etc/ld.so.conf.d:/etc/ld.so.conf.d \
  --name container_name image_name:image_version bash
```

2. Directly install them in docker

```
# Directly install sophon-libsophon and sophon-libsophon-dev in docker
sudo docker run \
  -td \
  --privileged=true \
  --name container_name image_name:image_version bash
```

Please read the related content in 安装libsophon, such as instll sophon-driver, sophon-libsophon and sophon-libsophon-dev before starting install them in docker

The image_name and image_versio is what users decided before. The name of the image is customize, either.

6.2 Checkout the environment

To make sure libsophon is working in docker, run the following commands in the docker:

```
# start the docker image and get into it
sudo docker exec -it container_name bash

# run this command to let libsophon can be found
ldconfig

# update environment variables so that libsophon tools can be straightly to use
for f in /etc/profile.d/*sophon*; do source $f; done
```

After that, opening bm-smi to check if the output is the same as the one in bm-smi使用说明.

CHAPTER 7

Instruction of Out_of_band interface

7.1 SMBUS protocol interface definition

7.1.1 Commands

Firstly write 1 byte of CMD to the i2c slave, then read n bytes data. Here is an instance of geting the temperature of CHIP0:

```
# write 0x0 to 0x60 which is slave address
i2c write 0x60 (slave addr) 0x0 (cmd)
# then read 1 byte data from 0x60
i2c read 0x60 (slave addr) 0x1 (n byte)
```

7.2 The out-of-band interface of SC5 series

7.2.1 Instruction

Server manufacturer BMC control

- The slave address of SC5 series card with multiple cpu are: CHIP0 -> 0x60, CHIP1 -> 0x61, CHIP2 -> 0x62
- While returning integer data, the higher bit is sent first (for example, integer 0x16841e30, return 0x16, 0x84, 0x1e, 0x30)

7.2.2 SC5+ MCU protocol command

Meaning	Address	Attribu	Description
Chip temperature	0x00	RO	unsigned byte, unit: centigrade
Card temperature	0x01	RO	unsigned byte, unit: centigrade
Card power	0x02	RO	unsigned byte, unit:w
Fan speed percentage	0x03	RO	unsigned byte, 0xff means no fan
Vendor ID	0x10	RO	unsigned int;[31:16]:Device ID 0x1684;[15:0]:Vendor ID 0x1e30;
Hardware Version	0x14	RO	unsigned byte
Firmware Version	0x18	RO	unsigned int;[7:0]Minor version;[15:8]Major version;[31:16]chip version
Kind of card	0x1c	RO	unsigned byte(Kind of card, SC5+ is 7)
Sub Vendor ID	0x20	RO	unsigned int;[15:0]:sub Vendor ID 0x0;[31:16]:sub system ID 0x0

Note:

- Can only read from 0x02 of 0x60 for the power of each card
- unsigned int means this address has 4 bytes valid data
- unsigned byte means this address has 1 bytes valid data

7.3 The out-of-band interface of SC7 series

7.3.1 Instruction

Server manufacturer BMC control

- The slave address of SC7 series card with multiple cpu are: CHIP0 -> 0x60, CHIP1 -> 0x62, CHIP2 -> 0x72
- While returning integer data, the higher bit is sent first (for example, integer 0x16841f1c30, return 0x16, 0x84, 0x1f, 0x1c)

Copyright © SOPHGO

7.3.2 SC7Pro MCU protocol command

Meaning	Address	Attribu	Description
Chip temperature	0x00	RO	unsigned byte, unit: centigrade
Card temperature	0x01	RO	unsigned byte, unit: centigrade
Card power	0x02	RO	unsigned byte, unit:w
Fan speed percentage	0x03	RO	unsigned byte, 0xff means no fan
Vendor ID	0x10	RO	unsigned int;[31:16]:Device ID 0x1686;[15:0]:Vendor ID 0x1f1c;
Hardware Version	0x14	RO	unsigned byte
Firmware Version	0x18	RO	unsigned int;[7:0]Minor version;[15:8]Major version;[31:16]chip version
Kind of card	0x1c	RO	unsigned byte(Kind of card,sc7pro is 0x21)
Sub Vendor ID	0x20	RO	unsigned int;[15:0]:sub Vendor ID 0x0;[31:16]:sub system ID 0x0
SN ID	0x24	RO	product SN code
MCU Version	0x36	RO	unsigned byte; MCU version:0

Note:

- Can only read from 0x02 of 0x60 for the power of each card
- unsigned int means this address has 4 bytes valid data
- unsigned byte means this address has 1 bytes valid data

Copyright © SOPHGO