Sophgo Edge Produc BSP Development Reference Manual

Release master

SOPHGO

Sep 03, 2024

Content

CHAPTER 1

Statement

Notice

The products, services or features you purchase shall be subject to the computing business contract and terms, All or some of the products, services, or features described in this document may not be within the scope of your purchase or use. Unless otherwise agreed in the contract, we make no representations or warranties of any kind, express or implied, regarding the content of this doc ument. This document may be updated from time to time due to product version upgrade or other reasons. Unless otherwise agreed, this document is

intended as a guide for use only, and all statements, information and recommendations in this document do not constitute any express or implied warranty.

Technical Supports

Address
Building 1, Zhongguancun Integrated Circuit Design Park (ICPARK), No.9 Fenghao East Road, Haidian District, Beijing

Postcode
100094

Website
https://www.sophgo.com/

E-mail
sales@sophgo.com

Telephone
+86-10-57590723 +86-10-57590724

CHAPTER 2

Preface

2.1 Overview

This document describes in detail the appearance, application scenarios, device parameters, electrical characteristics, supporting software, and operating environment of Athena2 series AI computing modules (including development boards). It helps users and developers of the device have a comprehensive and in-depth understanding of Athena2 series AI computing modules (including development boards). Device users and developers can install, debug, deploy, and maintain the device according to this manual.

2.2 Intended Audience

This document is intended for:

- FAE Engineer, pre-sales engineer
- Developers of Eco Partners
- R&d engineers and pre-sales engineers of user enterprises

2.3 The Interpretation of the Agreed Symbols, Signs and Expressions

The following symbols and symbols may appear in this document, and their meanings are as follows:

 Danger	Indicates a hazard with a high level of risk which, if not avoided, could result in death or serious injury.
 Warning	Indicates a hazard with a medium or low level of risk which, if not avoided, could result in minor or moderate injury
 Attention	Indicates a potential hazard, which if not avoided, could result in equipment damage, data loss and other consequences
 ESD Protection	This indicates electrostatic sensitive equipment or operation
 Warning Electric Shock	This indicates a high pressure hazard
 Trick	This may help you solve a problem or save you time
 Explanation	Provides additional information to emphasize or supplement important points in the main text

2.4 Abbreviation

JPU	JPEG Process Unit
VPP	Video Post Process
VPU	Video Process Unit

2.5 Modification Records

2.6 Declaration

reserves the right to pursue legal responsibility for any infringement of the Company' s copyright and other intellectual property rights and interests.

This product series will be constantly updated and iterated, and we will regularly check the contents of this product manual, which will inevitably be amended, deleted and supplemented in subsequent versions.

We reserve the right to make technical improvements, documentation changes, product improvements and upgrades, add or subtract product models and features without prior notice.

CHAPTER 3

Software Installation

3.1 Software Update

BM1688 is pre-installed with system software at the factory. You can check its version under Ubuntu with the following command:

```
linaro@sophon:~$ bm_version
SophonSDK(BM1688) 1.7
sophon-soc-libsophon : 0.4.9
sophon-soc-libsophon-dev : 0.4.9
sophon-media-soc-sophon-ffmpeg : 1.7.0
sophon-media-soc-sophon-opencv : 1.7.0
BL2 bm1688:g 2024-07-23T07:28:45+00:00
BL31 bm1688:gc3fb9f5-dirty 2024-07-22T15:27:39+08:00
U-Boot 2021.10 (Jul 23 2024 -07:28:41 +0000) cvitek_sophgo
Kernel version : Linux sophon 5.10.4-sophon-custom #1 SMP Tue Jul 23 07:28:49F
↪UTC 2024 aarch64 aarch64 aarch64 GNU/Linux
Mode: soc mode
MCU Version: 0.17.19
```

BM1688 currently provides two update methods: SD card flashing and USB burning. The SD card flashing will rewrite the entire eMMC, meaning all data stored on the eMMC will be lost. This method is the cleanest and most reliable; theoretically, as long as your BM1688 has no hardware damage, you can perform SD card flashing. The file replacement method refers to upgrading the bootloader, kernel, and other software by replacing the corresponding files under Ubuntu. This method carries certain risks, such as version compatibility between different software components and file corruption. The following describes the operations for both software update methods:

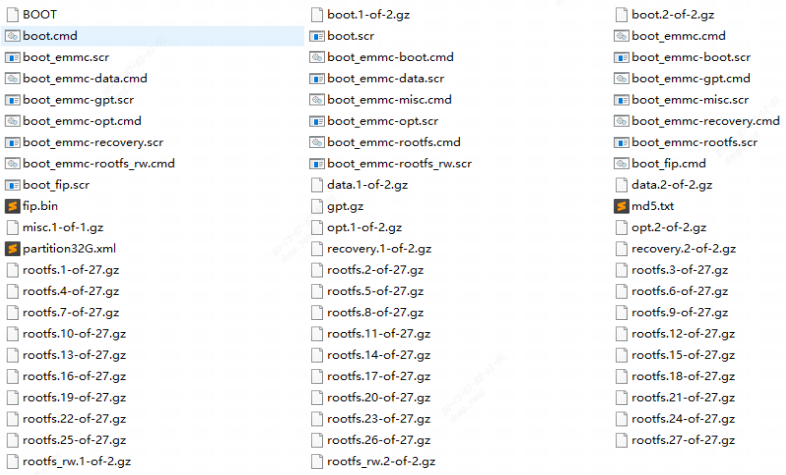
a. SD Card Flashing

Please format the SD card to FAT32 format (if the SD card has multiple partitions, only the first partition can be used), with a size of at least 1GB.

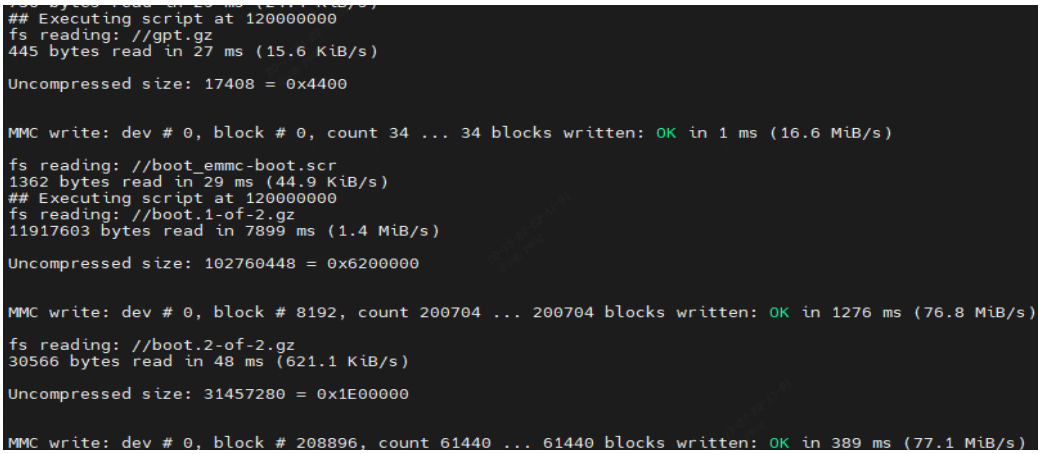
Please download the latest flashing package for BM1688 from the official website:

https://developer.sophgo.com/site/index/material/all/all.html

Please extract the downloaded sdcard.tgz to the root directory of the SD card. Confirm that the files are as follows (the quantity may vary):



Please power off BM1688, insert the SD card, and connect to the serial terminal, then power on BM1688. You will see BM1688 automatically enter the flashing process:



Flashing usually takes about 3 minutes. After it ends, you will see a prompt to remove the SD card and restart BM1688; please follow the instructions:

```
MMC write: dev # 0, block # 28602368, count 200704 ... 200704 blocks written: OK in 2556 ms (38.3 MiB/s)

fs reading: //data.2-of-2.gz
10588 bytes read in 9 ms (1.1 MiB/s)

Uncompressed size: 10866688 = 0xA5D000


MMC write: dev # 0, block # 28803072, count 21224 ... 21224 blocks written: OK in 143 ms (72.5 MiB/s)

eMMC update done
Unknown command 'bm_savelog' - try 'help'
all done
Unknown command 'led' - try 'help'
Unknown command 'led' - try 'help'
Unknown command 'set' - try 'help'
Unknown command 'led' - try 'help'
Unknown command 'set' - try 'help'
Please remove the installation medium, then reboot
Unknown command 'led' - try 'help'
Unknown command 'set' - try 'help'
Please remove the installation medium, then reboot
```

Please note: After flashing, the Ubuntu system will perform critical actions such as file system initialization during its first boot. Please do not power off indiscriminately; use the sudo poweroff command to shut down after entering the command line.

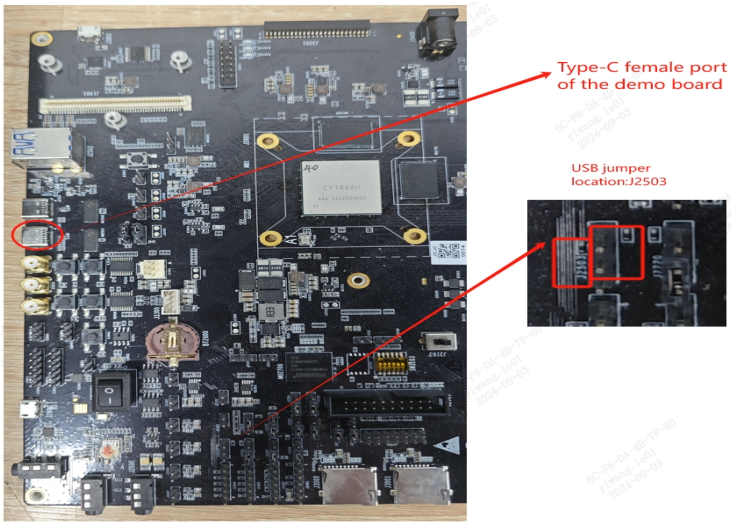
b. USB Burning

Precautions

1. Do not insert the SD card during USB burning.
2. usb_dl.exe must be in the same directory as cv_dl_magic.bin.
3. The path of usb_dl.exe and the path of the firmware to be burned must not contain Chinese characters.

Hardware Connection

1. Use a jumper cap to short the position as shown (short J2503 by connecting the two pins on the right side of the text).
2. Prepare a Type-C USB cable, connecting one end to the Type-C female port on the board and the other end to the PC.



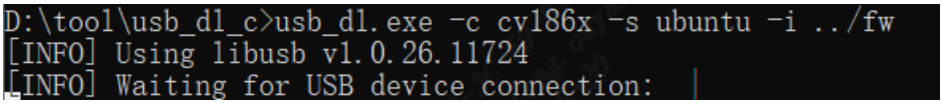
Install Windows Driver

Run CviUsbDownloadInstallDriver.exe, click Next, and then click Finish.

Burning Steps

1. Open the Windows console.
2. Place the USB burning tool in a local directory (the path of the USB burning tool is <SDK>/build/tools/<chip>/usb_dl).
3. In the same directory as usb_dl, create a folder named fw and place the files to be upgraded inside it.
4. Execute the upgrade command: usb_dl.exe -c cv186x -s ubuntu -i ../fw.

After entering the command, the following will be printed:



This indicates that it is waiting for the board to connect.

5. Connect the device to the computer using the USB data cable.
6. Power cycle the device (if the USB data cable also provides power, then when powering off, you must also unplug the USB cable).

3.2 Building Installation Package

In addition to using existing installation packages for upgrades, you can also build your own installation package.

3.2.1 Ubuntu Installation

It is recommended to install Ubuntu 20.04.

3.2.2 Installing and Configuring Repo

1. Configure the repo installation path:

```
mkdir ~/bin
```

2. Add environment variable:

```
vim ~/.profile
```

Add PATH=~/bin:$PATH at the end of the file.

3. Install repo:

```
curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
```

3.2.3 Docker Import

1. Install related software packages:

```
sudo apt-get update

sudo apt-get install -y build-essential ninja-build automake autoconf libtool wget \
curl git gcc libssl-dev bc slib squashfs-tools android-sdk-libsparse-utils \
android-sdk-ext4-utils jq cmake python3-distutils tcl scons parallel \
openssh-client tree python3-dev python3-pip ssh libncurses5 pkg-config \
lzop bison flex rsync kmod cpio sudo fakeroot dpkg-dev device-tree-compiler \
u-boot-tools uuid-dev libxml2-dev debootstrap qemu qemu-user-static kpartx \
binfmt-support git-lfs libisl-dev
```

2. Upgrade Python libraries:

```
pip3 install dfss --upgrade
```

3. Run dfss; if Python is not available, use Python3:

```
python -m dfss --url=open@sophgo.com:/gemini-sdk/docker/bm1688_docker.tar
```

4. Import the Docker image; if already imported, this step can be skipped:

```
docker load -i bm1688_docker.tar
```

5. Add the following command to ~/.bashrc and execute source ~/.bashrc:

```
function run_docker() {
 docker run  -e LOCAL_USER_ID=`id -u $USER_ID` --privileged -itd -v \$2:/project/\
↪$1 --name \$1  bm1688_docker:latest /bin/bash
}
```

6. Entering the Docker directory will change the working directory to /project/user.name/. Please download the code to /your/workspace/path before entering Docker:

```
run_docker sophon /your/workspace/path
```

7. Enter the shell in the Docker container:

```
docker exec -it sophon /bin/bash
```


3.2.4 Getting Source Code

Two methods are provided for obtaining the source code.

a. From GitHub

Create a personal account on GitHub and configure the SSH key. You will need your personal GitHub account to download the code.

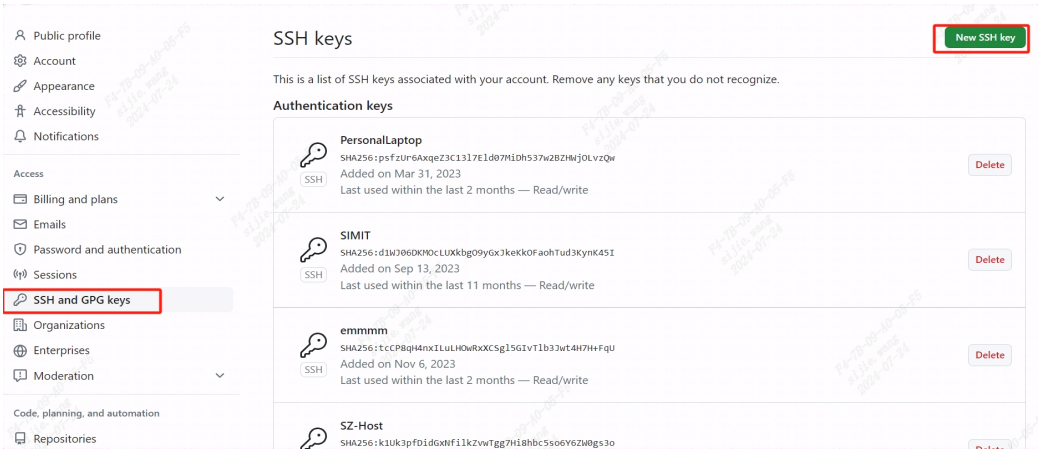
1. Set the account email:

```
git config --global user.name "your_name"

git config --global user.email "your_email@example.com"
```

2. Configure the key:

```
ssh-keygen -t ed25519 -C "your_email@example.com"

cat ~/.ssh/id_ed25519.pub
```

Add the public key to GitHub:



Verify if SSH is configured successfully:

```
ssh -T git@github.com
```

3. Use repo to pull the latest SDK source code:

```
repo init -u https://github.com/sophgo/manifest.git -m release/edge.xml

repo sync
```

b. From the Official Website

Download the SDK from the official website (https://developer.sophgo.com/site/index/material/81/all.html), then execute the following commands to obtain the latest version of the SDK:

```
unzip sophonsdk_edge_v1.6_offical_release.zip

cd sophonsdk_edge_v1.6_offical_release/sdk_release

tar -xvf bm1688_v1.6_source.tar.gz

repo sync
```

3.2.5 File Structure

The SDK file structure is described as follows:

```
top
├── build       ## Build scripts
├── fsbl        ## ATF
├── host-tools  ## Compilation toolchain
├── isp_tuning  ## ISP tuning
├── libsophon   ## Multimedia and TPU libraries
├── linux_5.10  ## Linux kernel
├── middleware  ## ISP libraries
├── osdrv       ## Drivers
├── oss         ## Open-source codebase
├── ramdisk     ## RAM disk
├── sophon_media ## FFmpeg and OpenCV libraries
├── u-boot-2021.10 ## U-Boot
└── ubuntu
```

3.2.6 Compiling BM1688 Firmware

```
source build/envsetup_soc.sh

defconfig  bm1688_wevb_emmc

build_bm1688_all
```

Note: Root privileges or Docker is required. The generated upgrade package will be located in install/soc_bm1688_wevb_emmc/ and package_edge/sdcard.

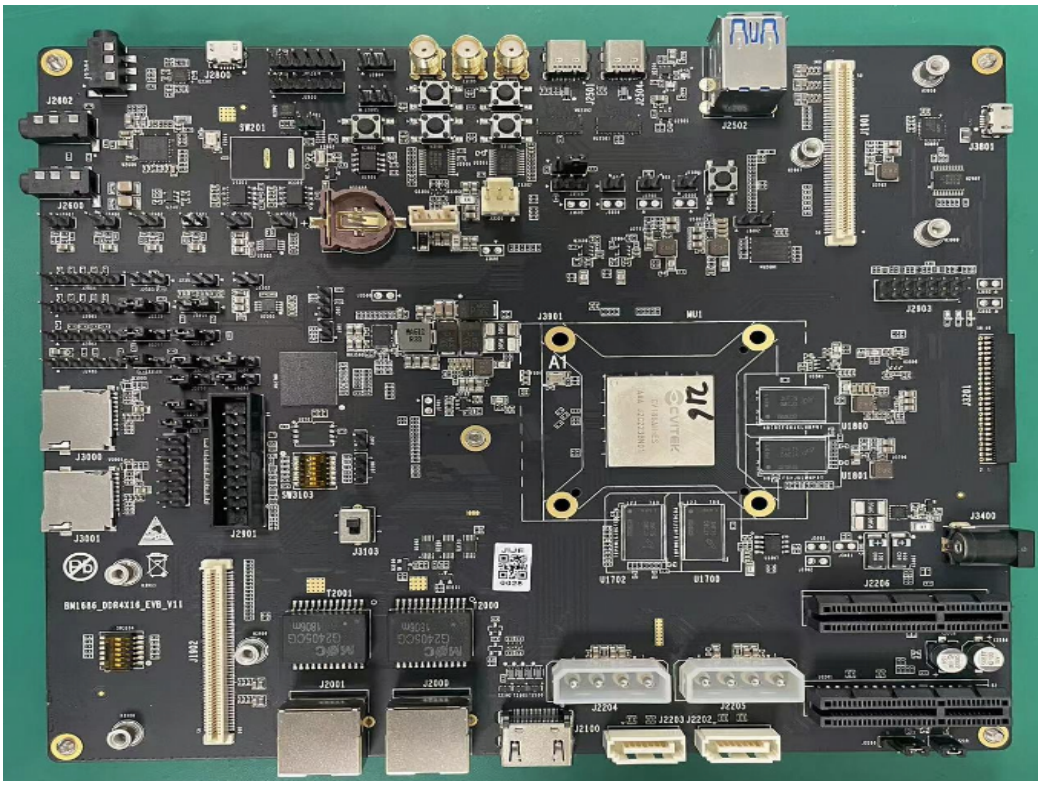
CHAPTER 4

Hardware Installation

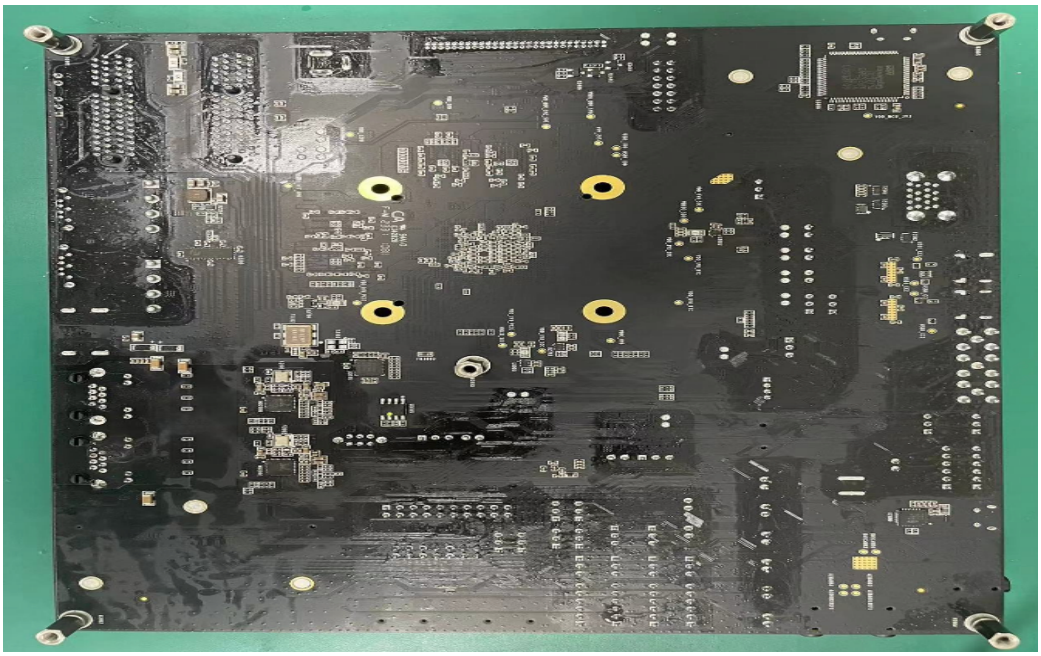
4.1 Board Installation

The BM1688 AI computing module refers only to the board that includes core components such as BM1688, LPDDR4X, eMMC, etc., as shown in the images below:

BM1688 EVB Front



BM1688 EVB Back



For convenience in the following descriptions, this document will refer to this board as the "core board."

4.2 Accessory Installation

To facilitate debugging, it is recommended that you prepare the following accessories:

a. A USB to UART cable: The UART0 (UART for BM1688) exposed by the core board serves as the debug port, with TTL levels, baud rate of 115200, 8 data bits, 1 stop bit, no parity, and no hardware flow control.

b. An Ethernet cable: Connect to Ethernet port 0 (eth0). The pre-installed system is set to DHCP by default, so it is more convenient to have the BM1688 and your debugging machine deployed under the same router via eth0.

c. An SD card: Used for flashing or debugging; it is recommended to use an 8GB/class10 or higher specification card.

d.  A power supply matching your baseboard design: If you are using our provided reference baseboard, the power adapter' s default specifications are 220V AC IN and 12V 5A OUT.

e. Cooling: Please install necessary cooling devices such as heat sinks and fans to prevent overheating shutdowns and other anomalies.

4.3 Power On

Once everything is ready, you can power on the baseboard. If you are using our provided reference baseboard for serial connection, prepare a Micro B USB cable, connecting one end to the Micro B female port on the board and the other end to the PC. After connecting the serial port, refer to the serial port diagram in section 5.11, figure 5.1. Please plug in the power supply (at this point, you should also see log printing from the serial terminal). Please check your serial terminal; the BM1688 is pre-installed with the Ubuntu 20.04 system, with the initial username and password both set to "linaro" (the root account has no initial password; you need to set a password for it using the sudo passwd root command after logging in with the linaro account):

```
Ubuntu 20.04 LTS sophon ttyS0

sophon login: linaro
Password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.10.4-tag--g09c5e5af6d9b aarch64)

* Documentation:  https://help.ubuntu.com
* Management:     https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
just raised the bar for easy, resilient and secure K8s cluster deployment.
```

(continues on next page)

(continued from previous page)

```
https://ubuntu.com/engage/secure-kubernetes-at-the-edge
overlay / overlay rw,relatime,lowerdir=/media/root-ro,upperdir=/media/root-rw/
↪overlay,workdir=/media/root-rw/overlay-workdir 0 0
/dev/mmcblk0p5 /media/root-rw ext4 rw,relatime 0 0
/dev/mmcblk0p4 /media/root-ro ext4 ro,relatime 0 0

Last login: Fri Sep  8 20:07:47 CST 2023 on ttyS0
linaro@sophon:~$
```

To check the IP address, use the ifconfig or ip a command:

```
ifconfig
ip a
```

If you need to manually configure a static IP, you can modify the /etc/netplan/01-netcfg.yaml configuration file as follows and enable the modified configuration file:

```
$ cat /etc/netplan/01-netcfg.yaml
network:
        version: 2
        renderer: networkd
        ethernets:
                eth0:
                        dhcp4: no                       # Change to no for static IP, yes for
↪dynamic IP
                        addresses: [192.168.1.100/24]    # Add IP; leave empty for dynamic
↪IP
                        optional: yes
                        dhcp-identifier: mac            # Remove this line for static IP
                eth1:
                        dhcp4: no
                        addresses: [192.168.150.1/24]
                        optional: yes
                enp3s0:
                        dhcp4: yes
                        addresses: []
                        dhcp-identifier: mac
                        optional: yes
$ sudo netplan try      # Test if the configuration is available
$ sudo netplan apply    # Enable the latest configuration
```

Once you have obtained the IP address, you can use SSH to log in, with the port number being 22 and the username and password both being "linaro."

```
ssh linaro@your_ip
```

It is recommended to use the sudo poweroff command when shutting down to avoid directly cutting off power, which may damage the file system. 

The core board has two network cards; eth0 is set to DHCP by default, so you need to obtain

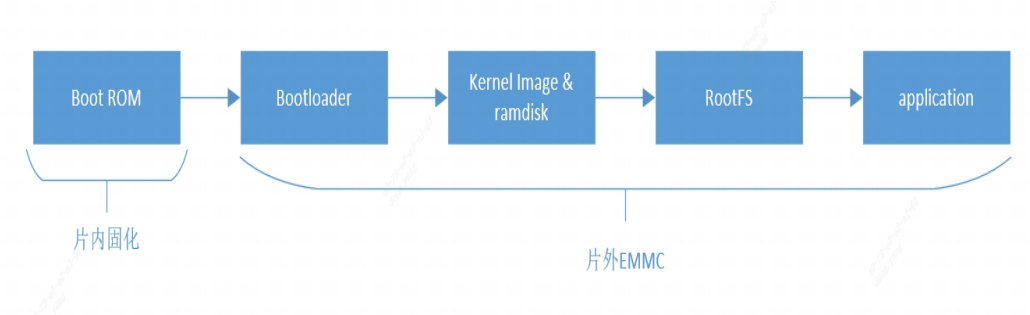
the IP address using the method described above. Eth1 is configured with a static IP of 192.168.150.1.

CHAPTER 5

System Software Constitution

5.1 Startup Process

The Athena2 system software is a typical embedded ARM64 Linux, which consists of bootloader, kernel, ramdisk, and Ubuntu 20.04. After the system is started, run the following command in sequence:

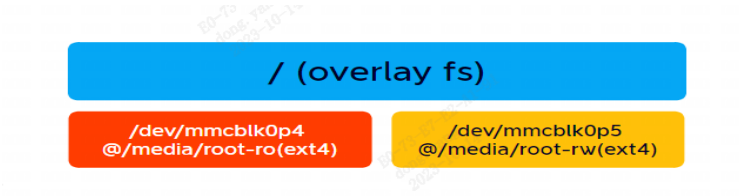


Boot ROM and bootloader are built based on fsbl and u-boot. kernel is built on the 5.10 branch of Linux. Ubuntu 20.04 is built on the official Ubuntu arm64 source and does not contain GUI-related components.

5.2 EMMC Partition

Device file	Mount point	File system	Content
/dev/mmcblk0p1	/boot	FAT32	Kernel and ramdisk mirrors
/dev/mmcblk0p2	/recovery	EXT4	Recovery mode image.
/dev/mmcblk0p3	none	none	Not currently in use.
/dev/mmcblk0p4	/media/root-ro	EXT4	Read-only part of the u-boot system.
/dev/mmcblk0p5	/media/root-rw	EXT4	Read-write part of the u-boot system.
/dev/mmcblk0p6	/data	EXT4	Driver and runtime environment for sophon sdk.

Note on the fourth and fifth partitions: The fourth partition stores key parts of the Ubuntu 20.04 system and is mounted as read-only; The fifth partition stores the files generated during the running of Ubuntu 20.04 and is mounted as readable and writable. The two partitions are aggregated by overlayfs and then mounted as the root directory of the system, as shown in the following figure:



Users usually do not need to pay attention to this detail, for daily use is transparent, normal operation of the root directory files can be, but when using df and other commands to view the partition usage and other operations, please be aware of this, as shown below:

```
linaro@sophon:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
overlay         8.7G  167M  8.2G   2% /
devtmpfs        2.4G     0  2.4G   0% /dev
tmpfs           2.4G     0  2.4G   0% /dev/shm
tmpfs           488M  2.0M  486M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           2.4G     0  2.4G   0% /sys/fs/cgroup
/dev/mmcblk0p1  128M   15M  114M  12% /boot
/dev/mmcblk0p6   17G   23M   17G   1% /data
/dev/mmcblk0p4  2.4G  2.3G   18M 100% /media/root-ro
/dev/mmcblk0p5  8.7G  167M  8.2G   2% /media/root-rw
/dev/mmcblk0p2  108M   19M   85M  19% /recovery
tmpfs           488M     0  488M   0% /run/user/1000
linaro@sophon:~$
```

5.3 Docker

The docker service is pre-installed on the core board system. You can run the docker info command to check the status. Note that the root directory of docker is configured under /data/docker, which is different from the default setting.

5.4 File System Support

If you use the reference baseboard, the storage device is identified as /dev/sdb1 or a similar node after a USB flash drive or portable hard drive is inserted (taking into account the USB power supply capability), which is the same as in the desktop PC Linux environment. The file system supports FAT, FAT32, EXT2/3/4, and NTFS. Athena2 does not support automatic mounting. Therefore, you need to mount sudo mount /dev/sdb1/mnt. When access to NTFS format storage devices, pre-loaded kernel version supports only read, if you need to write, need to manually install NTFS-3g package, please refer to the https://packages.ubuntu.com/search?keywords=ntfs-3g. After data is written, run the sync or umount command in a timely manner. Run the sudo poweroff command to avoid forcible power-off and data loss.

5.5 Ethernet Operation Guide

5.5.1 Operation Example

Ethernet modules are programmed into the kernel by default and do not require additional loading operations.

The procedure for using network ports in the kernel is as follows:

- Configure the ip address and subnet mask

  sudo ifconfig eth0 xxx.xxx.xxx.xxx netmask xxx.xxx.xxx.xxx up

- Setting the default gateway

  sudo route add default gw xxx.xxx.xxx.xxx

- Mounting the NFS

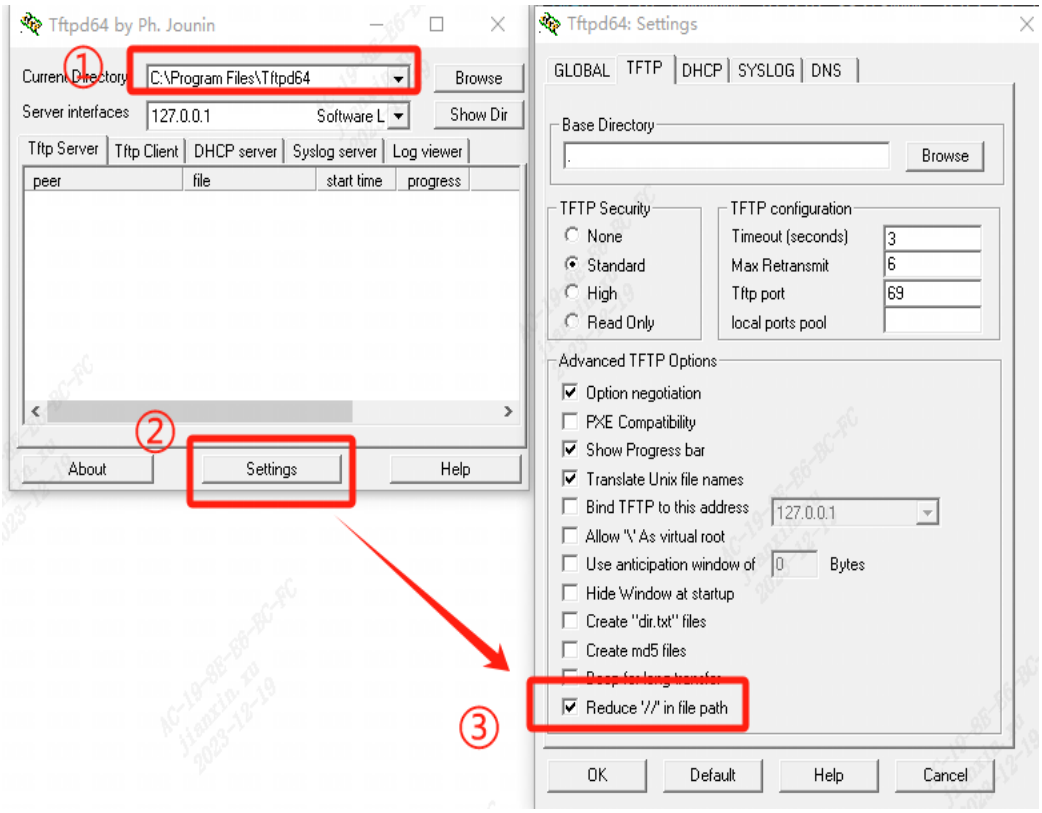
  sudo busybox mount -t nfs -o nolock xxx.xxx.xxx.xxx:/your/path /mount-dir

  Ex: sudo busybox mount -t nfs -o nolock 192.168.2.10:/NFS /mnt/sd

5.5.1.1 Tftp Instructions

· Start by opening the Tftp server in Window

1. If there is no TFTP tool, please download tftpd64 tool, link to https://pjo2.github.io/tftpd64/ .
2. Click Download page, download the tFTpd64-bit software package, and then install the tftpd tool.
3. Configure the tftpd64 tool as follows.



You can click the Browse button to set the Current Directory in the diagram to the tftp file transfer path for example: C:Program FilesTftpd64;

Set Server interfaces to the local network interface, for example, 192.168.137.1.

On the setting screen, select Reduce '//' in file path on the TFTP page.

Note: Be sure to use 64-bit software and check "Reduce '//' in file path" .

After the Tftp server is enabled in windows, you can download files from the Tftp server on the PC in the uboot and kernel.

· Download from the uboot through tftp

1. Connect the board to the PC through a network cable, use a serial cable to connect the PC to the board, and open the serial port terminal of the PC.

2. After power-on, when the serial port terminal prompts you to Hit any key to stop autoboot, press Enter to enter the u-boot command line mode and see the prompt "sophon#"

3. Enter the following command to configure the network:

```
setenv ipaddr 192.168.137.11
setenv gatewayip 192.168.337.1
setenv serverip 192.168.137.1;
```

   Ipaddr is the board IP address, serverip is the IP address of the PC where the tftp server resides, and gatewayip is the gateway address.

4. Enter the following command to start the tftp download:

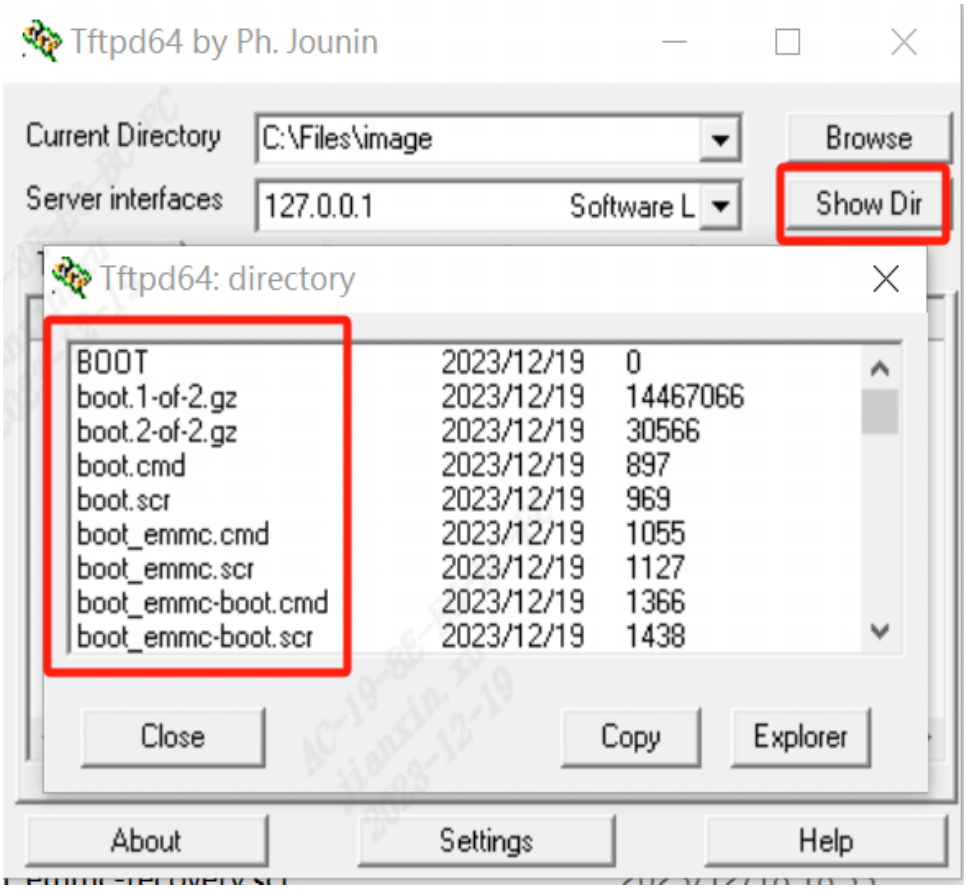
```
tftp xxxxxxxxx filename
```

   This command downloads a file named filename from the tftp server to the address xxxxxxxxx.

5. Upgrade the system through tftp

   The product supports upgrading the system through tftp during the uboot phase. You need to download the system installation package tftp.tgz. After decompressing the installation package, copy all installation files to the Current Directory on the PC server.

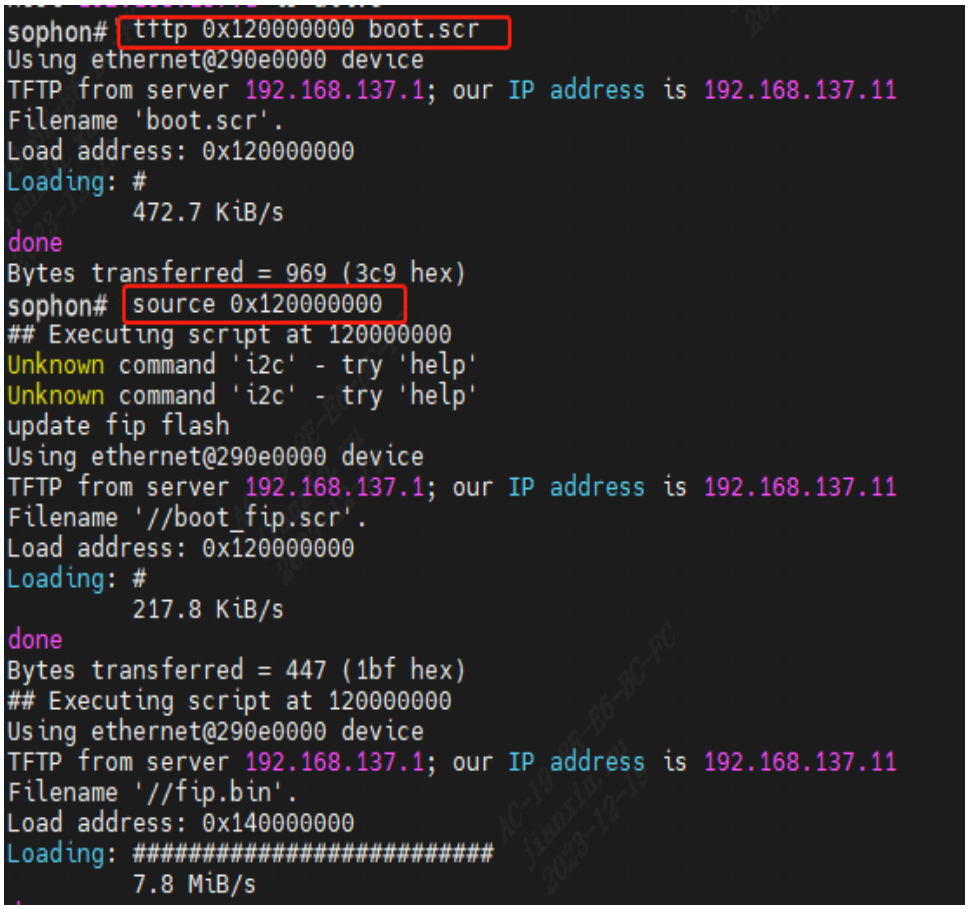
   Click the Show Dir button on the PC server to display the system upgrade file in the path.



After configuring the network, enter the following command to upgrade the system:

```
tftp 0x120000000 boot.scr    // Download boot.scr to 0x120000000
source 0x120000000           // Execute the boot.scr file
```

The following information is displayed on the serial port and the upgrade progress is displayed with a #.



Wait until the serial port displays "Please remove the installation medium, then reboot" , and the system is successfully upgraded.

- Upload and download the kernel through tftp

  After the Tftp server is running on the PC, connect the board to the PC through a network cable. Enter the following commands to download and upload files.

  Download files:

  ```
  sudo busybox tftp –g -r [remote file name] [server ip]
  ```

  Note: remote file name indicates the name of the file to be downloaded, and server ip indicates the ip address of the server where the file is to be downloaded (ex: sudo busybox tftp -g -r test.txt 192.168.0.11).

  Upload files:

  ```
  sudo busybox tftp –p -l [local file name] [server ip]
  ```

  Note: local file name indicates the name of the local file to be uploaded, and server ip indicates the ip address of the target server (ex: sudo busybox tftp -p -l test.txt 192.168.0.11).

5.5.2 IPv6 Instructions

The IPv6 environment can be configured as follows:

- Configure the ip address and gateway

```
sudo ip -6 addr add <ipv6 address>/ipv6 prefixlen dev <port name>
```

  Ex:sudo ip -6 addr add 3FFE:FFFF:7654:FEDA:1245:BA98:3210:4564/64 dev eth1

- Ping the specified IPv6 address

```
#ping -6 <ipv6 address>
```

  Ex: ping -6 3FFE:FFFF:7654:FEDA:1245:BA98:3210:4563

5.5.3 IEEE 802.3x Flow Control Function

5.5.3.1 Flow Control Function Description

Athena2 Ethernet supports the flow control function defined by IEEE 802.3x for flow control purposes by sending flow control frames and receiving flow control frames sent from the peer end.

- Send flow control frames:

  When receiving packets from the peer end, the local end sends a flow control frame to the peer end and asks the peer end not to send packets for a period of time. In this way, traffic control is implemented.

- Receive flow control frames:

  When receiving a flow control frame from the peer end, the local end delays sending the packet to the peer end according to the flow control time in the frame. After the flow control delay time expires, the local end sends the packet to the peer end. If a flow control frame sent by the peer end is received during the waiting process and the flow control time described is 0, the transmission is directly started.

5.5.3.2 Flow Control Function Configuration

Send flow control frame function related files are configured in linux/drivers/net/ethernet/stmicro/stmmac/stmmac_main.c

```
static int flow_ctrl = FLOW_AUTO;
module_param(flow_ctrl, int, 0644);
MODULE_PARM_DESC(flow_ctrl, "Flow control ability [on/off]");
```

(continues on next page)

(continued from previous page)

```
static int pause = PAUSE_TIME;
module_param(pause, int, 0644);
MODULE_PARM_DESC(pause, "Flow Control Pause Time");
```

If you want to change the default pause time, you can configure pause to a target.

5.5.3.3 Ethtool configures the interface flow control function

Users can enable the flow control function through the standard ethtool interface.

The ethtool -a eth0 command is used to check the flow control status of eth0. The following information is displayed:

```
# ethtool -a eth0
Pause parameters for eth0:
Autonegotiate: on
RX: off
TX: off
```

RX flow control is off, TX flow control is off; TX flow control can be turned on or off by the following command:

```
# ethtool -A eth0 tx off
# ethtool -A eth0 tx on
```

Note: The ethtool tool is not compiled into the file system by default. You can add it to the file system when necessary.

5.6 USB Instructions

5.6.1 Preparation

Perform the following operations on the USB 2.0 Host/Device:

- U-boot and Linux Kernel U-boot and Linux kernel are released using the SDK.
- The file system can use the local file system ext4 or squashfs, or NFS.
- run_usb.sh customizes USB device devices using the kernel' s USB ConfigFS function. Users can refer to and modify run_usb.sh to change PID/VID and function parameters. Detailed operation can be reference kernel file "linux/Documentation/usb/gadget_configfs.txt" .

5.6.2 Uboot Operating Process

5.6.2.1 Configure the USB Host in the Uboot

The Uboot supports only USB flash drives and hard disks. USB host is disabled by default in uboot. You need to enable the related config.

Step 1. Enable the USB-related driver in the uboot.

```
CONFIG_USB=y
CONFIG_DM_USB=y
CONFIG_USB_STORAGE=y
CONFIG_CMD_USB=y
```

Step 2.

For athena2 series, modify include/configs/athena2-asic.h and add a new definition:

```
#define CONFIG_USB_DWC2
#define CONFIG_USB_DWC2_REG_ADDR 0x04340000
```

Step 3. Compile the driver. Compiling uboot generates fip.bin

```
build_uboot
```

5.6.2.2 USB in Uboot

Preparations before starting the Uboot USB Host:

In the Uboot, USB Host does not support hot insertion. Therefore, you must plug in the device before starting the USB host. If a USB Hub is installed on the platform, ensure that the Hub is powered on and the Switch in the USB path is switched to the Host connector.

Take athena2 for example:

Power on the platform, access the uboot CLI, run "usb start" command, and check whether the identification is successful.

```
phobos_c906# usb start
starting USB...
USB0: Core Release: 4.00a
scanning bus 0 for devices...Device NOT ready
   Request Sense returned 02 3A 00
2 USB Device(s) found
      scanning usb for storage devices...2 Storage Device(s) found
```

If an enumeration error occurs after usb start or the device cannot be detected, run the setenv usb_pgood_delay XXX command on the uboot CLI to change the timeout value for a device that is preheated slowly or connected to the Hub in the middle. The recommended value ranges from 1000 to 3000.

After the recognition is complete, run "usb tree" command to view the recognition rate. The following is an example of connecting a USB host to a Hub and a storage device.

```
phobos_c906# usb tree
USB device tree:
  1 Hub (480 Mb/s, 0mA)
  | U-Boot Root Hub
  |
  +-2 Mass Storage (480 Mb/s, 500mA)
    Generic USB3.0 Card Reader 000000001532
```

Initialization and application:

After the identification is complete, you can proceed to the following operations.

Step 1. View device information

- Run "usb info [dev]" command to view information about all devices on the controller. Examples are as follows:

```
phobos_c906# usb info 1
config for device 1
2: Mass Storage, USB Revision 2.10
- Generic USB3.0 Card Reader 000000001532
- Class: (from Interface) Mass Storage
- PacketSize: 64 Configurations: 1
- Vendor: 0x05e3 Product 0x0749 Version 21.50
Configuration: 1
- Interfaces: 1 Bus Powered 500mA
  Interface: 0
  - Alternate Setting 0, Endpoints: 2
  - Class Mass Storage, Transp.SCSI, Bulk only
  - Endpoint 1 In Bulk MaxPacket 512
  - Endpoint 2 Out Bulk MaxPacket 512
```

Step 2. Read the USB flash drive

- Run the command "usb read addr blk# cnt" to read the data from the storage device whose start address is blk and size is cnt to the location where DDR is addr. Examples are as follows:

```
phobos_c906# usb read 0x90000000 0 10

USB read: device 0 block # 0, count 16 ...16 blocks read: OK
```

Step 3. Write the USB flash drive

- Run the usb write addr blk# cnt command to write the data with the DDR address addr and the size of cnt to the blk start address of the storage device. Examples are as follows:

```
phobos_c906# usb write 0x90000000 2000 2000
USB read: device 0 block # 8192, count 8192 ...8192 blocks write: OK
```

5.6.3 linux Host

5.6.3.1 USB 2.0 Host Operating Process

Step 1. Start the platform and load ext4 or squashfs, or use NFS.

Step 2. Load the driver

insmod usb-common.ko

insmod usbcore.ko

insmod udc-core.ko

insmod roles.ko

insmod dwc2.ko

Step 3. Set a usb role

echo host > /proc/cviusb/otg_role

5.6.3.2 U disk operation example

Insert detection:

Insert the USB flash drive directly and check whether the enumeration is successful. Normally, the serial port is printed as follows:

```
[ 72.061964] usb 1-1: new high-speed USB device number 2 usingdwc2
[ 72.315816] usb-storage 1-1:1.0: USB Mass Storage device detected
[ 72.335934] scsihost0: usb-storage 1-1:1.0
[ 73.363027] scsi 0:0:0:0: Direct-Access Generic STORAGE DEVICE1532 PQ: 0 ANSI: 6
[ 73.374407] sd 0:0:0:0: Attached scsigeneric sg0 type 0
[ 73.558597] sd 0:0:0:0: [sda] 30253056 512-byte logical blocks:(15.5 GB/14.4 GiB)
[ 73.566961] sd 0:0:0:0: [sda] Write Protect is off
[ 73.571922] sd 0:0:0:0: [sda] Mode Sense: 21 00 00 00
[ 73.577899] sd 0:0:0:0: [sda] Write cache: disabled, read cache:enabled, doesn't support DPO or F
↪FUA
[ 73.593961] sda: sda1[ 73.602607] sd 0:0:0:0: [sda] Attached SCSI removable disk
```

sda1 indicates the first partition on the USB flash drive or portable hard drive. If there are multiple partitions, sda1, sd2, and sda3 are displayed.

Initialization and application:

After insertion detection, perform the following operations:

In sdXY, X indicates the disk ID and Y indicates the partition ID. You can change the value based on the system environment.

- The device node operated by the partition command is sdX, example: ~$ fdisk /dev/sda。

- The specific partition formatted with the mkdosfs tool is sdXY: ~$ mkdosfs -F 32 /dev/sda1。
- The specific partition to be mounted is sdXY: ~$ mount /dev/sda1 /mnt。

1. Viewing partition information
   - Run "ls /dev" command to view the device file. If the partition information is not sdXY, there is no partition. Use fdisk to partition the disk and go to Step 2.
   - If the partition information is sdXY, the USB flash drive partition is detected. Go to Step 2.
2. View formatting information
   - If the format is not configured, run the mkdosfs command to format it and go to Step 3.
   - if configured, go to Step 3.
3. Mount directory
   - Run "mount /dev/sdaxy-mnt" command to mount the directory.
4. Read and write operations on hard disks

5.6.4 linux Device

5.6.4.1 USB 2.0 Device Operating Process

Step 1. Compile the kernel driver module related to the USB2.0 Device

- Configure using menuconfig

```
Device Driver --->
  [*] USB support --->
    <*> USB Gadget Support --->
      <M> USB functions configurable through configfs
      [*] Abstract Control Model (CDC ACM)
      [*] Mass storage


- Compile kernel modules and generate.ko files.
```

Step 2. Start the platform and load the ext4 or squashfs file system, or use NFS.

Step 3. When the platform is used as a Device, the USB2.0 Device module must be loaded to identify the platform as a USB device on the Host.

All USB 2.0 Device-related drivers are listed below.

5.6.4.2 USB Storage Device Operating Example

When the platform functions as a Device, it supports eMMC and SD storage media. The operations are as follows:

Step 1. Load the module.

insmod configfs.ko

insmod usb-common.ko

insmod udc-core.ko

insmod libcomposite.ko

insmod usbcore.ko

insmod roles.ko

insmod dwc2.ko

The paths of USB Device related modules in the kernel are as follows:

drivers/usb/gadget/libcomposite.ko

drivers/usb/gadget/function/usb_f_mass_storage.ko

fs/configfs/configfs.ko

Step 2. Switch the otg controller to device mode

Step 3. echo device > /proc/cviusb/otg_role

Step 4. run shell script "usb_usb.sh"

run_usb.sh probe msc /dev/mmcblkXY

run_usb.sh start

Step 5. The path of USB Device-related modules under rootfs is as follows: /etc/run_usb.sh

Step 6. After connecting a platform to a Host using USB, the Host identifies the platform as a USB storage device and generates device nodes in the /dev directory.

Step 7. At the Host end, the platform can be treated as an ordinary USB storage device, and it can be partitioned, formatted, read and write.

5.6.4.3 USB Terminal Equipment Operating Example

When the platform is used as a Device, it can be used as a terminal device, and the operation is as follows:

Step 1. Insert the module.

insmod configfs.ko
insmod libcomposite.ko

insmod u_serial.ko
insmod usb_f_acm.ko
insmod usb_f_serial.ko

The paths of USB Device related modules in the kernel are as follows:

- drivers/usb/gadget/libcomposite.ko
- drivers/usb/gadget/function/usb_f_serial.ko
- drivers/usb/gadget/function/usb_f_acm.ko
- drivers/usb/gadget/function/u_serial.ko
- fs/configfs/configfs.ko

Switch otg controller to device mode

echo device > /proc/cviusb/otg_role

run "run_usb.sh"
run_usb.sh probe acm
run_usb.sh start

The path of USB Device related modules under rootfs is as follows:
/etc/run_usb.sh

Step 2. Connect the platform to the Host by using USB. The Host identifies the platform as a USB terminal device, and generates the corresponding device node ttyACMX in the /dev directory, where X is the number of the same type terminal device. ttyGSY is generated in the /dev directory on the device side, where Y indicates the number of a terminal device of the same type.

Host and Device can transmit data through terminal devices.

5.6.4.4 USB RNDIS Device Operating Example

When the platform is used as a Device, it can be used as an RNDIS device. The operation is as follows:

Step 1. Insert the module.

insmod configfs.ko
insmod libcomposite.ko
insmod u_ether.ko
insmod usb_f_ecm.ko

insmod usb_f_eem.ko
insmod usb_f_rndis.ko

The paths of USB Device related modules in the kernel are as follows:

- drivers/usb/gadget/libcomposite.ko
- drivers/usb/gadget/function/usb_f_ecm.ko
- drivers/usb/gadget/function/usb_f_ecm.ko
- drivers/usb/gadget/function/usb_f_rndis.ko
- drivers/usb/gadget/function/u_ether.ko
- fs/configfs/configfs.ko

Switch otg controller to device mode

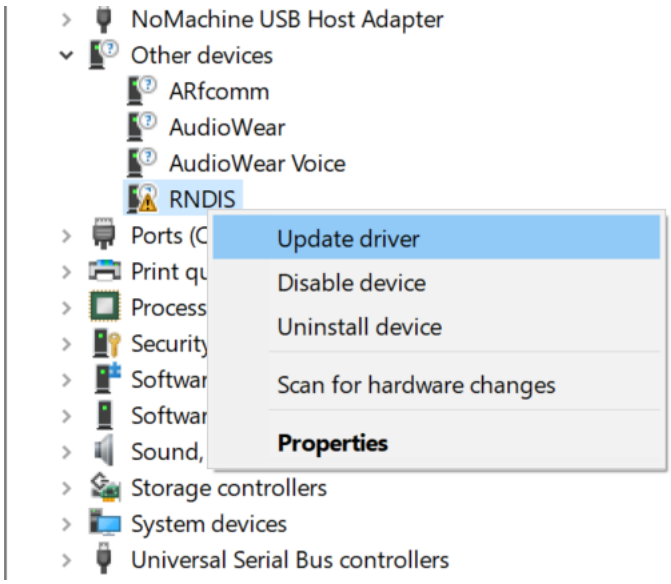
echo device > /proc/cviusb/otg_role

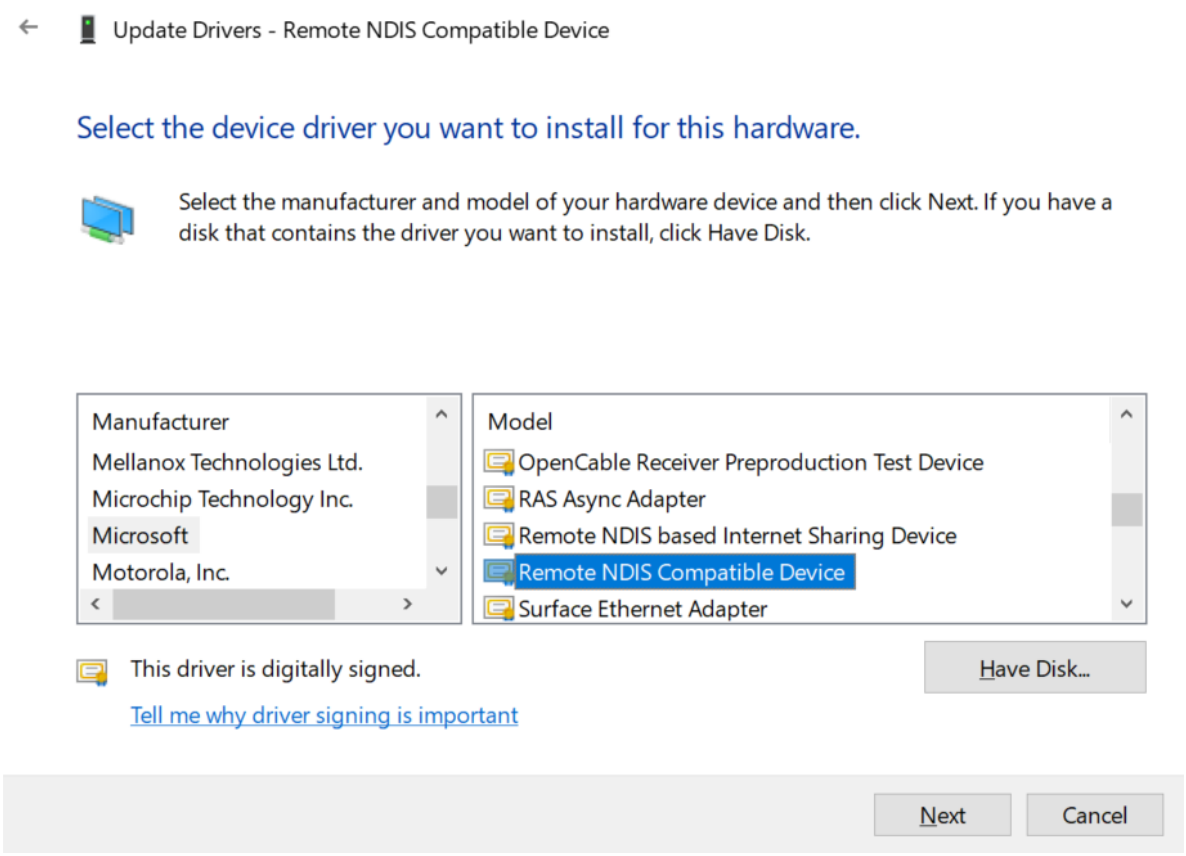
run "run_usb.sh"
run_usb.sh probe rndis
run_usb.sh start

The path of USB Device-related modules under rootfs is as follows:

/etc/run_usb.sh

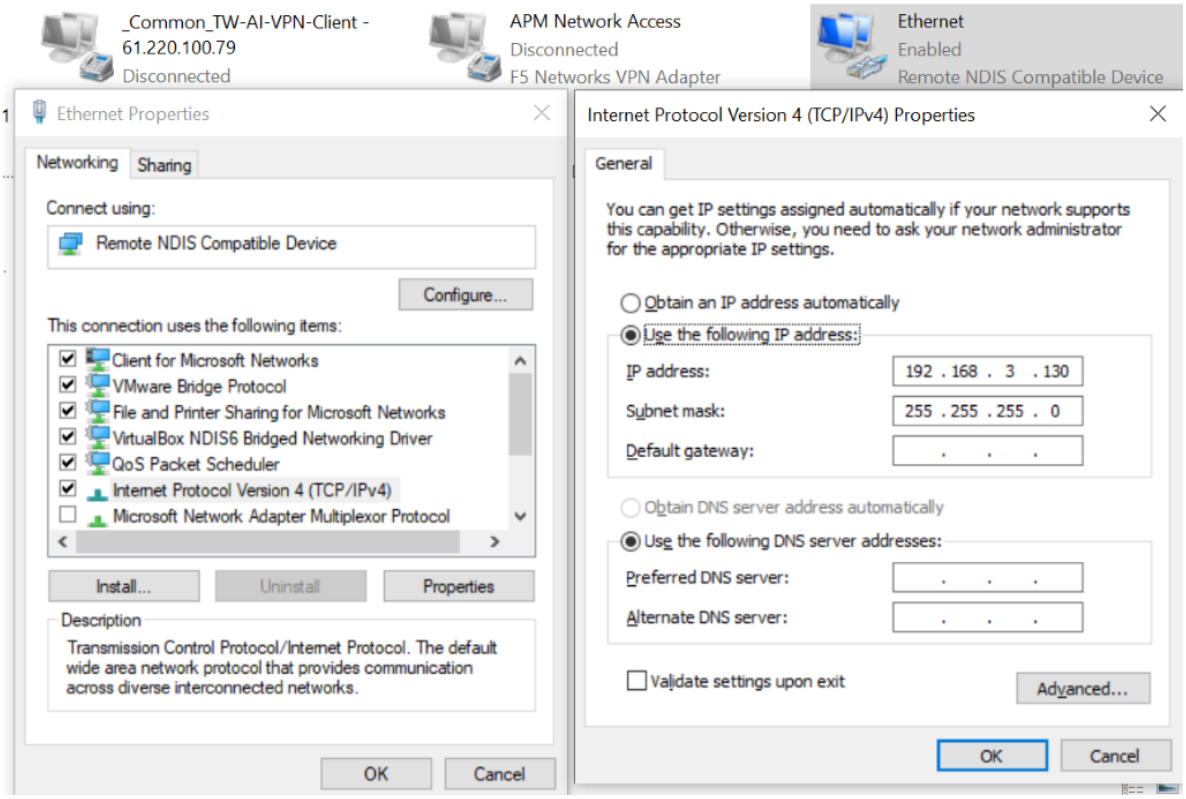
Step 2. Connect the platform to the Host using a USB port. The Host identifies the platform as a USB Remote NDIS Device, and install the Remote NDIS Compatible Device driver on Windows.





Step 3. Set an IP address on the board, for example, ifconfig usb0 192.168.3.101 up.

Step 4. Set the IP address on Windows.



Host and Device can transmit data through the RNDIS device.

5.6.4.5 USB Device CVITEK USB GADGET Operating Example

The platform can be used as a Device using a custom CVITEK USB Gadget(CVG) as follows:

Step 1. Insert the module

insmod configfs.ko
insmod libcomposite.ko
insmod usb_f_cvg.ko

The paths of USB Device related modules in the kernel are as follows:

- drivers/usb/gadget/libcomposite.ko
- drivers/usb/gadget/function/usb_f_cvg.ko
- fs/configfs/configfs.ko

Switch otg controller to device mode

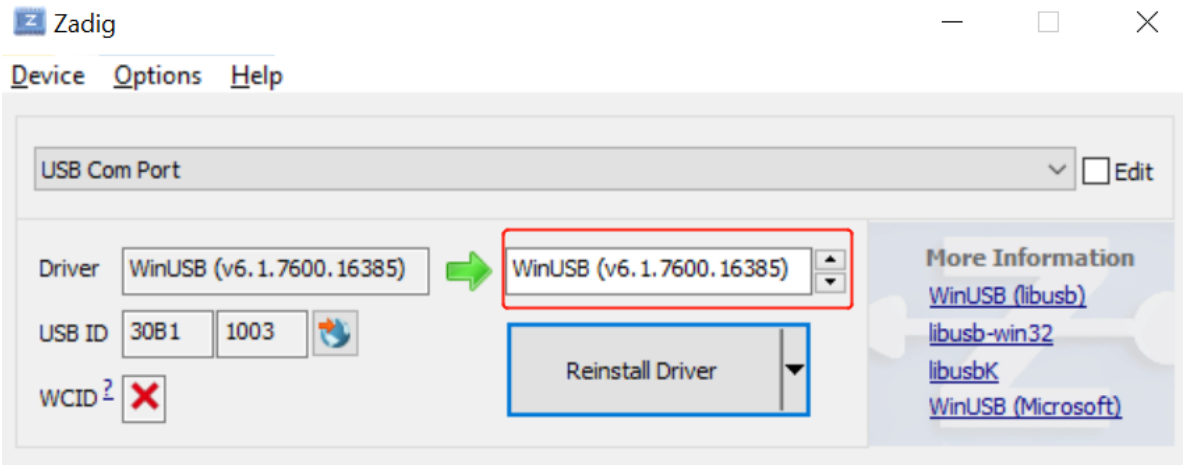
echo device > /proc/cviusb/otg_role

run "run_usb.sh"

```
run_usb.sh probe cvg
run_usb.sh start
```

/etc/run_usb.sh

Step 2. Connect the platform to the Host using USB. Use Zadig to install libusb(WinUSB) as the device driver.



Step 3. Run the TEST program sample_cvg [#TEST] on the board.

Step 4. Run "cvg/pctool/ gen_pattern.sh command" on the PC to generate test Patterns. Run "cvg/pctool/cvg_test.py" command to start the test.

Step 5. Refer to the CVITEK USB Gadget 使用指南.docx for details.

5.6.5 Matters Need Attention

Note the following:

- After the system starts, the default value is Host mode. To use the Device mode, load the module and execute the USB ConfigFS script. Before switching the Device, the user must confirm the following:
  - The USB Cable is not connected to the Host
  - The hardware on the platform must be switched to the corresponding USB mode. For example, before switching to Device mode, turn off the USB 5V power supply on the platform. If there is a Hub on the platform, power off the Hub and Switch the path to the Device mode connector.
- If you want to use Host mode again after switching to Device mode, you must restart the platform.
- When the platform is used as a terminal device, data may be lost if a large amount of data is transferred in a short period of time due to the characteristics of TTY terminals. Users must be aware of this limitation when using this feature.

· When using USB Host to read the USB flash drive in Uboot, if there is a Hub on the platform, power on the Hub and Switch the path switch to the correct Connector.

5.7 SD/MMC Card Instructions

5.7.1 Preparation

1. U-boot and kernel distributed with SDK.
2. File system:
   For SD/MMC cards, the SDK only supports the FAT file system, which can be read and written.
   After the kernel is started, mount it to the /mnt/sd directory or the required directory.
3. You can use the fdisk tool to implement partition work.
4. Athena2 SD supports 2.0 and 3.0：

At present, SD card supports 1.8/3.3V VDDIO, EMMC only supports 1.8V, users should pay attention to.

5.7.2 Operating Process

5. By default, all SD/MMC driver modules have been programmed into the kernel, and you do not need to run additional loading commands.
6. Insert the card and power it on. You can view the contents of the card through FAT-related commands in U-boot. After the platform is started to the kernel, the response nodes /dev/mmcblk0 and /dev/mmcblk0p1 are identified by card scanning.
7. The Uboot card does not support hot swap. The kernel card supports hot swap. After an SD card is inserted into the kernel, you can perform operations on the SD card. For details, see 3.3 Operation Examples.

5.7.3 Operating Example

The following is an example of reading and writing an SD card.

Initialization and application:

After the SD card is inserted, perform the following operations (X is the partition number, and its value is determined by the fdisk tool during partitioning):
Specify the specific directory for fdisk operations as: ~ $ fdisk /dev/mmcblk0

Step 1. Checking partition information

a. If p1 is not displayed, the SD card is not partitioned. Use the fdisk tool to partition the SD card in Linux or format the SD card in windows and go to Step 2.

b. If partition p1 is displayed, the SD card has been detected and partitioned. You can go to Step 2 to mount the SD card.

Step 2. Mounting

~ $ mount /dev/mmcblk1pX /mnt/sd. This command will mount the XTH partition on the SD card to the /mnt/sd directory.

5.7.4 Matters Need Attention

1. Ensure that the SD card is properly connected to the hardware pin of the card slot. If the SD card is improperly connected, a detection error or read/write error may occur, resulting in a read/write failure.
2. After you insert an SD card, you need to mount it once before you can read or write the SD card. If the SD card has been mounted to the file system, you must unmount the SD card before removing the SD card. Otherwise, the SD card partition may not be visible after the next insertion. In addition, abnormal card withdrawal also requires uninstallation.
3. You must ensure that the SD card has created a partition and formatted the partition as a FAT or FAT32 file system (using the fdisk command in LINUX or the disk management tool in Windows).
4. Operations that cannot be performed during normal operation:
   - Do not remove the SD card when reading or writing the SD card. Otherwise, some abnormal information will be printed, and the files or file system in the card may be damaged.
   - If the current directory is under the mount directory, such as /mnt/sd, the uninstallation cannot be performed. You must leave the current directory, such as /mnt/sd, to perform the uninstallation.
   - The process of reading and writing the mount directory in the system cannot be uninstalled until the process of reading and writing the mount directory is complete.
   - What to do when an exception occurs during operation:
     1. If the file system is damaged due to data reading or writing or other unknown reasons, a file system error message may be displayed when reading or writing the SD card. In this case, you need to uninstall the SD card, remove the card, insert and mount the card again, and then read or write the SD card again.
     2. Because the SD card registration, detection/cancellation process takes a certain amount of time, so if the card is inserted quickly after the card is removed, the SD card may not be detected.

3. If the card is removed abnormally during the test, the user needs to press ctrl+c to exit back to the kernel shell, otherwise the abnormal information will be continuously printed.
4. When there is more than one partition on the SD card, you can switch between different partitions by mounting them. However, it is important to ensure that the number of mount operations is equal to the number of unmount operations to ensure that all mounted partitions are completely unmounted.

5.8 I2C Instructions

5.8.1 Preparation

Preparations for I2C are as follows:

- Use the kernel released in the SDK.

5.8.2 Operating Process

- Load the kernel. By default, all I2C-related modules have been programmed into the kernel, and you do not need to run the loading command.
- By running I2C read and write commands under the console or writing I2C read and write programs in kernel or user mode, you can read and write peripheral devices mounted on the I2C controller.

5.8.3 Description of Interface Rate Settings

If you want to change the interface rate, you need to modify the clock_frequency of i2c node in build/boards/default/DTS/athena2 / athena2_base.dtsi, as shown below, and recompile the kernel.

```
i2c0: i2c@29000000 {
  compatible = "snps,designware-i2c";
  clocks = <&clk ATHENA2_CLK_I2C>;
  reg = <0x0 0x29000000 0x0 0x1000>;
  clock-frequency = <400000>;

  #size-cells = <0x0>;
  #address-cells = <0x1>;
  resets = <&rst RST_I2C0>;
  reset-names = "i2c0";
};
```

5.8.3.1 Example of I2C Read and Write

You can run IIC-related commands to detect the bus device and read and write the i2c device on the bus on the linux terminal.

1. i2cdetect -l

iic bus in the detection system (i2c-0~9 in athena2)

2. i2cdetect -y -r N To detect all device addresses connected to the i2c-N bus, check the devices on the i2c-2 as follows:

```
linaro@sophon:~$ sudo i2cdetect -y -r 2
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- 17 -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- 3c -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
linaro@sophon:~$
```

3. i2cdump -f -y N M // View the values of all registers in the device with address M on i2c-N.
4. i2cget -f -y 0 0x3c 0x00 // Reads the value of register 0x00 on a device with address 0x3c on i2c-0.
5. i2cset -f -y 0 0x3c 0x40 0x12 // Write to register 0x40 on device with address 0x3c on i2c-0.

5.8.3.2 Kernel-mode I2C Read and Write Program Example

This example shows how to read and write to an I2C peripheral in kernel mode using an I2C reader.

Step 1. Assuming the peripheral is known to be mounted on I2C controller 0, call the i2c_get_adapter() function to get the I2C controller structure adapter:

adapter = i2c_ger_adapter(0);

Step 2. The I2C controller is connected to the I2c peripheral through the i2c_new_device() function to obtain the client structure of the I2C peripheral:

client = i2c_new_device(adapter, &info)

Note: The info structure provides the device address of the i2c peripheral

Step 3. Call the standard read and write functions provided by the I2C core layer to read and write peripheral devices:

ret = i2c_master_send(client, buf, count);

ret = i2c_master_recv(client, buf, count);

Note: client is the structure obtained in step 2, buf is the register address and data to be read and written, and count is the length of buf.

The code example is as follows:

```
// Declare a peripheral named "dummy" and the device address 0x3c
static struct i2c_board_info info = {
        I2C_BOARD_INFO("dummy", 0x3C),
};
static struct i2c_client *client;

static int cvi_i2c_dev_init(void) {
  // Allocate i2c controller Pointers
  struct i2c_adapter \*adapter;

  adapter =i2c_get_adapter(0);
  client = i2c_new_device(adapter, &info);
  i2c_put_adapter(adapter);
  return 0;
}

static int i2c_dev_write(char *buf, unsigned int count){
  int ret;

  ret = i2c_master_write(client, buf, count);
  return ret;
}

static int i2c_dev_read(char *buf, unsigned int count) {
  int ret;

  ret =i2c_master_recv(client, buf, count);
  return ret;
}
```

5.8.3.3 User Mode I2C Read and Write Program Example

In this example, the I2C read and write program is used to read and write I2C peripheral devices in user mode.

Step 1. Open the device file corresponding to the I2C bus and obtain the file descriptor:

```
i2c_file = open("/dev/i2c-0", O_RDWR);
  if (i2c_file < 0) {
    printf("open I2C device failed %d\n", errno);
    return -ENODEV;
  }
```

Step 2.
To read and write data:

```
ret = ioctl(file, I2C_RDWR, &packets);
  if (ret < 0) {
    perror("Unable to send data");
    return ret;
  }
```

Note: Read and write operations must be specified in flags.

```
struct i2c_msg messages[2];
  int ret;

  /*
  * In order to read a register, we first do a "dummy write" by writing
  * 0 bytes to the register we want to read from.  This is similar to
  * the packet in set_i2c_register, except it's 1 byte rather than 2.
  */
  outbuf = reg;
  messages[0].addr = addr;
  messages[0].flags = 0;
  messages[0].len = sizeof(outbuf);
  messages[0].buf = &outbuf;

  /* The data will get returned in this structure */
  messages[1].addr = addr;
  /* | I2C_M_NOSTART */
  messages[1].flags = I2C_M_RD;
  messages[1].len = sizeof(inbuf);
  messages[1].buf = &inbuf;
```


5.9 SPI Instructions

5.9.1 Preparation

The operation preparation of SPI is as follows:

- Use the kernel and file system distributed by the SDK. The file system can use squashFS or ext4 released with the SDK. It can also be mounted to NFS from the local file system over the network.

5.9.1.1 Operating Process

- Load the kernel. All SPI related modules are programmed into the kernel without the need to execute the loading command.
- By running SPI read and write commands under the console or writing SPI read and write programs in kernel or user mode, you can read and write peripheral devices mounted on the SPI controller.

5.9.2 Operation Example

5.9.2.1 Example of Kernel-mode SPI Read and Write PExample of kernel-mode SPI read-write programrogram

This operation example shows how to read and write SPI peripheral devices in kernel mode through SPI reader.

Step 1. Call the SPI core layer function spi_busnum_to_master() to get a description of the SPI controller structure:

master = spi_busnum_to_master(bus_num);

// bus_num indicates the number of the controller where the SPI peripheral device to be read and written resides.

// master is a pointer to the spi_master struct type describing the SPI controller.

Step 2. Call the spi core layer function by the name of the SPI peripheral in the core layer to get the structure mounted on the SPI controller describing the SPI peripheral:

snprintf(str, sizeof(str), "%s.%u" , dev_name(&master->dev), cs);

dev = bus_find_device_by_name(&spi_bus_type, NULL, str);

spi = to_spi_device(dev);

// spi_buf_type is the bus_type struct type variable describing the SPI bus.

// spi is a pointer to the type of the spi_device structure that describes the SPI peripheral device.

Step 3. Call the SPI core layer function to add spi_transfer to the spi_message queue.

spi_message_init(&m)

spi_message_add_tail(&t, &m)

// t is the spi_transfer struct type variable.

// m is the spi_message struct type variable.

Step 4. Call he read and write function of SPI core layer to read and write peripheral devices.

status = spi_sync(spi, &m);

status = spi_async(spi, &m)

// spi is a pointer to the type of the spi_device structure that describes the SPI peripheral.

// The spi_sync function is used for spi synchronous read and write operations.

// The spi_async function is used to perform spi asynchronous read and write operations.

The code example is as follows:

This code example is for reference only, not for actual application functionality.

```
// Pass in the SPI controller bus number and slice selection number
static unsigned int busnum;module_param(busnum, uint, 0);
MODULE_PARM_DESC(busnum, "SPI busnumber (default=0)");

static unsigned int cs;
module_param(cs, uint, 0);
MODULE_PARM_DESC(cs, "SPI chip select (default=0)");

extern struct bus_typespi_bus_type;

// Declares the structure of the SPI controller
static struct spi_master *master;

// Declares the structure of the SPI peripheral
static struct spi_device *spi_device;

static int __init spidev_init(void) {
  char *spi_name;
  struct device *spi;
  master =spi_busnum_to_master(busnum);
  spi_name = kzalloc(strlen(dev_name(&master->dev)), GFP_KERNEL);

  if (!spi_name)
    return -ENOMEM;
```

(continues on next page)

(continued from previous page)

```
  snprintf(spi_name,sizeof(spi_name), "%s.%u", dev_name(&master->dev),cs);
  spi = bus_find_device_by_name(&spi_bus_type, NULL, spi_name);
  if (spi == NULL)
    return -EPERM;

  spi_device = to_spi_device(spi);
  if (spi_device ==NULL)
    return -EPERM;

  put_device(spi);
  kfree(spi_name);

  return 0;
}
int spi_dev_write(, void \*buf,unsigned long len, int buswidth)
{
  struct spi_device *spi = spi_device;
  struct spi_transfer t = {
    .speed_hz = 2000000,
    .tx_buf = buf,
    .len = len,
  };
  struct spi_message m;
  spi->mode = SPI_MODE_0;

  if (buswidth == 16)
    t.bits_per_word = 16;
    else
      t.bits_per_word = 8;
    if (!spi) {
      return -ENODEV;
    }
    spi_message_init(&m);
    spi_message_add_tail(&t, &m);
    return spi_sync(spi, &m);
}
int spi_dev_read(unsigned char devaddr, unsigned char reg_addr,void*buf, size_t len)
{
  struct spi_device *spi = spi_device;
  int ret;
  u8 txbuf[4] = { 0, };
  struct spi_transfer t = {
    .speed_hz = 2000000,
    .rx_buf =buf,
    .len = len,
    };
  struct spi_message m;
  spi->mode = SPI_MODE_0;

  if (!spi) {
    return -ENODEV;
```

(continues on next page)

(continued from previous page)

```
  }
  txbuf[0] = devaddr;
  txbuf[1] = 0;
  txbuf[2] = reg_addr;
  t.tx_buf =txbuf;

  spi_message_init(&m);
  spi_message_add_tail(&t, &m);
  ret = spi_sync(spi, &m);

  return ret;
}
```

5.9.2.2 User Mode SPI Read and Write Program Example

This operation example implements read and write operations on SPI peripherals mounted on SPI controller 0 in user mode. (For details, see tools/spi/spidev_test.c)

Step 1: Open the device file corresponding to the SPI bus and obtain the file descriptor.

```
static const char *device = "/dev/spidev32766.0";
...
fd = open(device, O_RDWR);
if (fd < 0)
  pabort("can't open device");
```

Note: The default peripheral attached to SPI controller 1 is dev/spidev32765.0.

The default peripheral attached to SPI controller 2 is dev/spidev32764.0.

The default peripheral attached to SPI controller 3 is dev/spidev32763.0.

You only need to replace the node name, and the rest of the operations are the same as the peripheral mounted on SPI controller 0.

Step 2: The SPI transfer mode is set by ioctl.

```
/*
*spi mode
*/
ret = ioctl(fd, SPI_IOC_WR_MODE32, &mode);
if (ret == -1)
  pabort("can't set spi mode");
ret = ioctl(fd, SPI_IOC_RD_MODE32, &mode);
if(ret == -1)
  pabort("can't get spi mode");
```

Note: Please refer to the following figure for the mode value configuration or the kernel code include/linux/spi/ SPi.h.

Ex. mode = SPI_MODE_3 | SPI_LSB_FIRST;

```
#define SPI_CPHA 0x01 /* clock phase */
#define SPI_CPOL 0x02 /* clock polarity */
#define SPI_MODE_0 (0|0) /* (original MicroWire) */
#define SPI_MODE_1 (0|SPI_CPHA)
#define SPI_MODE_2 (SPI_CPOL|0)
#define SPI_MODE_3 (SPI_CPOL|SPI_CPHA)
```

Step 3: Set the SPI transmission bandwidth via ioctl.

```
/*
* bits per word
*/
ret = ioctl(fd, SPI_IOC_WR_BITS_PER_WORD, &bits);
if (ret == -1)
   pabort("can't set bits per word");
ret = ioctl(fd,SPI_IOC_RD_BITS_PER_WORD, &bits);
if (ret == -1)
   pabort("can't get bits per word");
```

Step 4: Set the SPI transfer speed through ioctl (speed = 25M is recommended).

```
/*
* max speed hz
*/
ret = ioctl(fd, SPI_IOC_WR_MAX_SPEED_HZ, &speed);
if (ret == -1)
   pabort("can't set max speed hz");

ret = ioctl(fd,SPI_IOC_RD_MAX_SPEED_HZ, &speed);
if (ret == -1)
   pabort("can't get max speed hz");
```

Step 5: Use ioctl to read and write data.

```
ret = ioctl(fd, SPI_IOC_MESSAGE(1), &tr);
if (ret < 1)
   pabort("can't send spi message");
```

Note: tr transfers the first address of the spi_ioc array of a frame message.

5.10 GPIO Instructions

5.10.1 Preparation

- Use the kernel released in the SDK.

5.10.1.1 Operating Process

- All default GPIO related modules have been programmed into the kernel, and you do not need to run the loading command.
- Run the GPIO read and write command under the console or write the GPIO read and write program in kernel mode or user mode to perform input and output operations on GPIO.

5.10.2 Operation Example

5.10.2.1 Examples of GPIO Commands

Step 1: Run the echo command on the console to specify the GPIO number N.

echo N > /sys/class/gpio/export

N Indicates the number of the GPIO to be operated. GPIO number = GPIO group number + offset value.

The GPIO_num pin in the schematic diagram is used as an example. num % 32 = base ··· off. The corresponding group ID is base and the offset value is off.

For example, GPIO32, 32 % 32 = 1 ··· 0, the group ID is 1, the offset value is 0, and the group ID corresponding to group 1 is 448

Therefore, the GPIO number N is 448 + 0 = 448

The mapping between group numbers and values is as follows:

group 0 corresponds to linux group value 480
group 1 corresponds to linux group value 448
group 2 corresponds to linux group value 416
group 3 corresponds to linux group value 384
group 4 corresponds to linux group value 352
group 5 corresponds to linux group value 320
pwr_gpio corresponds to linux group value 288

After the "echo N > /sys/class/gpio/export" command, the "/sys/class/gpio/gpioN" directory is generated.

Step 2: In the console, use the echo command to set the GPIO direction.

Set to output mode: echo in > /sys/class/gpio/gpioN/direction

Set to input mode: echo out > /sys/class/gpio/gpioN/direction

Set GPIO32 (i.e. 448) direction as input:
echo in > /sys/class/gpio/gpio448/direction

Set GPIO32 (i.e. 448) direction as output:
echo out > /sys/class/gpio/gpio448/direction

Step 3: In the console, use the cat command to view the GPIO input value, or use the echo command to set the GPIO output value.

View input value: cat /sys/class/gpio/gpioN/value

Output low level: echo 0 > /sys/class/gpio/gpioN/value

Output high level: echo 1 > /sys/class/gpio/gpioN/value

Step 4: After use, use the echo command on the console to release resources.

echo N > /sys/class/gpio/unexport

Note: The sysfs debug function of gpio can be enabled by opening the CONFIG_DEBUG_FS option. Before the operation, run the following command to check the group number corresponding to the gpio pin.

cat /sys/kernel/debug/gpio

5.10.2.2 Kernel Mode GPIO Operating Example

Examples of kernel mode GPIO read and write operations:

Step 1. Register GPIO.

```
gpio_request(gpio_num, NULL);
```

gpio_num indicates the GPIO number to be operated. The number is GPIO group number + intra-group offset number.

Step 2. Set the GPIO direction.

```
For input: gpio_direction_input(gpio_num)
For output: gpio_direction_output(gpio_num, gpio_out_val)
```

Step 3. View the GPIO input value or set the GPIO output value.

```
View input value: gpio_get_value(gpio_num);
Output low level: gpio_set_value(gpio_num, 0);
Output high level: gpio_set_value(gpio_num, 1);
```

Step 4. Release the registered GPIO number.

```
gpio_free(gpio_num);
```

Kernel mode GPIO interrupt operating example:

Step 1. Register GPIO.

```
gpio_request(gpio_num, NULL);
```

gpio_num indicates the GPIO number to be operated. The number is GPIO group number + intra-group offset number.

Step 2. Set the GPIO direction.

```
gpio_direction_input(gpio_num);
```

For the GPIO pin to be used as the interrupt source, the direction must be configured as input to output.

Step 3. Map the interrupt number corresponding to the GPIO number of the operation.

```
irq_num = gpio_to_irq(gpio_num);
```

Return value with interrupt number gpio_to_irq(gpio_num).

Step 4. Registor interrupt:

```
request_irq(irq_num, gpio_dev_test_isr, irqflags, "gpio_dev_test",&gpio_irq_type))
```

Irqflags indicates the interrupt type that needs to be registered. Common types are:

IRQF_SHARED: Sharing interrupt;

IRQF_TRIGGER_RISING: Rising edge trigger;

IRQF_TRIGGER_FALLING: Falling edge trigger;

IRQF_TRIGGER_HIGH: High level trigger;

IRQF_TRIGGER_LOW: Low-level trigger;

Step 5. Release the registered interrupt and GPIO number at the end.

```
free_irq(gpio_to_irq(gpio_num), &gpio_irq_type);
gpio_free(gpio_num);
```


5.10.2.3 User Mode GPIO Operating Example

User mode GPIO read and write operating example:

Step 1. Export the GPIO number to be operated.

```
fp = fopen("/sys/class/gpio/export", "w");
fprintf(fp, "%d", gpio_num);
fclose(fp);
```

gpio_num indicates the GPIO number to be operated. The number is GPIO group number + intra-group offset number.

Step 2. Set the GPIO direction.

```
fp = fopen("/sys/class/gpio/gpio%d/direction", "rb+");
For input: fprintf(fp, "in");
For output: fprintf(fp, "out");

fclose(fp);
```

Step 3. View the GPIO input value or set the GPIO output value.

```
fp = fopen("/sys/class/gpio/gpio%d/direction", "rb+");
View input value: fread(buf, sizeof(char), sizeof(buf) - 1, fp);
Output low level:
  strcpy(buf, “0” );
  fwrite(buf, sizeof(char), sizeof(buf) - 1, fp);
Output high level:
  strcpy(buf, “1” );
  fwrite(buf, sizeof(char), sizeof(buf) - 1, fp);
```

Step 4. Unexport the GPIO number.

```
fp = fopen("/sys/class/gpio/unexport", "w");
fprintf(fp, "%d", gpio_num);
fclose(fp);
```

5.11 UART Instructions

5.11.1 Preparation

- Use the kernel released in the SDK;
- Change the device tree node status to “okay” ;

```
uart0: serial@29180000 {
    compatible = "snps,dw-apb-uart";
    reg = <0x0 0x29180000 0x0 0x1000>;
```

(continues on next page)

(continued from previous page)

```
    clock-frequency = <200000000>;
    reg-shift = <2>;
    reg-io-width = <4>;
    status = "okay";
};
```

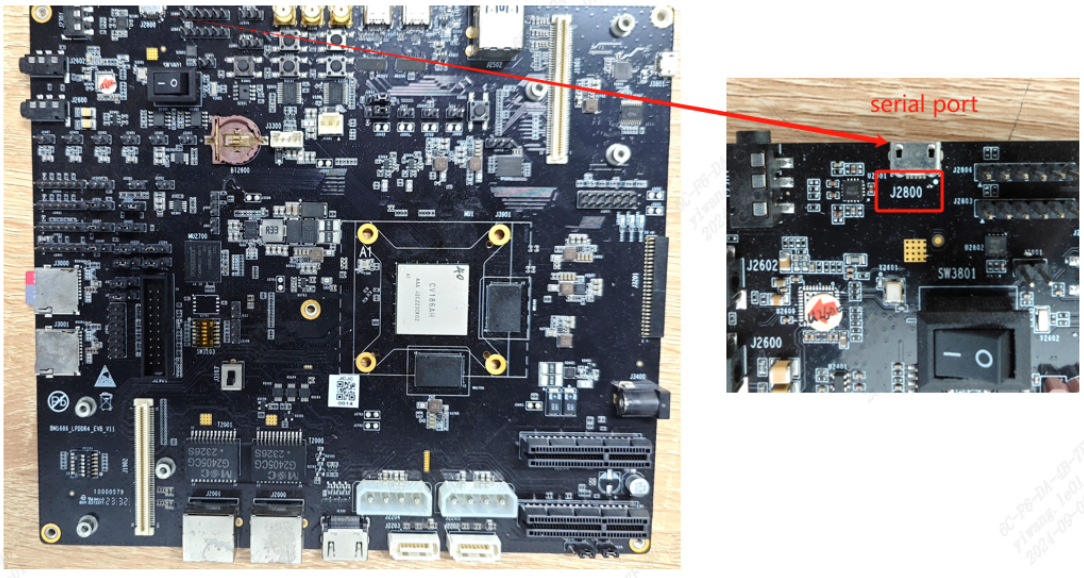


Fig. 5.1: Serial Port Diagram

5.11.2 Operating Process

5.11.2.1 Example of Command Line Operation

- Set the PINMUX corresponding to the pin;
- Execute the following command to set the baud rate to 115200, data bit 8, stop bit 1 (ttySx, x is 0~8):

  stty -F /dev/ttySx ispeed 115200 ospeed 115200 cs8 stop 1

- Send and receive data:

  echo 123 > /dev/ttySx
  cat /dev/ttySx

5.11.2.2 User Mode UART Operating Program Example

```
/* Open your specific device (e.g., /dev/mydevice): */
fd = open ("/dev/ttyS0", O_RDWR);
if (fd < 0) {
    /* Error handling. See errno. */
    return -1;
}

struct termios t;
tcgetattr(fd, &t);
cfsetispeed(&t, B115200); // Set the input baud rate to 115200
cfsetospeed(&t, B115200); // Set the output baud rate to 115200
t.c_cflag &= ~PARENB;   // Don't use parity check bit
t.c_cflag &= ~CSTOPB;   // Use 1 stop bit
t.c_cflag &= ~CSIZE;    // Clear data bit settings
t.c_cflag |= CS8;       // Set the data bit to 8 bits
tcsetattr(fd, TCSANOW, &t);
// read/write ...
```

5.12 Watchdog Instructions

5.12.1 Preparation

- Use the kernel released in the SDK.

5.12.1.1 Module Compilation

- Insert module: insmod athena2_wdt.ko.
- To operate the Watchdog, run the Watchdog read/write command on the console or write the Watchdog read/write program in kernel or user mode.

5.12.2 Operating Example

Watchdog uses the standard linux framework to provide a hardware watchdog. Users can use watchdog simply by turning it on, off, or setting timeout. The system restarts when the watchdog timeout occurs.

The Watchdog is disabled by default. Customers can decide whether to enable it. The timeout times that can be specified are 1 second, 2 seconds, 5 seconds, 10 seconds, 21 seconds, 42 seconds, and 85 seconds.

When the user enters timtout time as 8 seconds, the driver will select a timeout greater than or equal to this value, that is, 10 seconds; If timeout is not set, the driver uses the default 42 seconds.

- Open WATCHDOG

  Open the /dev/watchdog device. The watchdog is started. You should ping (feed the dog) immediately after opening, otherwise wdt will restart immediately.

```
int wdt_fd = -1;
wdt_fd = open("/dev/watchdog", O_WRONLY);
if (wdt_fd == -1)
{
  // fail to open watchdog device
}
ioctl(fd, WDIOC_KEEPALIVE, 0);
```

- Close WATCHDOG

  Driver support "Magic Close". The magic character 'V' must be written to the watchdog device before closing the watchdog.

  If the userspace daemon shuts down the device without sending a 'V', the watchdog driver keeps counting, and if the dog is not fed within a given period of time, it still causes a timeout and the system restarts.

  The reference code is as follows:

```
int option = WDIOS_DISABLECARD;
ioctl(wdt_fd, WDIOC_SETOPTIONS, &option);
if (wdt_fd != -1)
{
  write(wdt_fd, "V", 1);
  close(wdt_fd);
  wdt_fd = -1;
}
```

- Set TIMEOUT value

Set timeout in seconds by the standard IOCTL command WDIOC_SETTIMEOUT. The timeout times that can be specified are 1 second, 2 seconds, 5 seconds, 10 seconds, 21 seconds, 42 seconds, and 85 seconds.

```
#define WATCHDOG_IOCTL_BASE 'W'
#define WDIOC_SETTIMEOUT \_IOWR(WATCHDOG_IOCTL_BASE, 6, int)

int timeout = 10;
ioctl(wdt_fd, WDIOC_SETTIMEOUT, &timeout);
```

- PING watchdog

  Feed the dog with the standard IOCTL command WDIOC_KEEPALIVE.

```
while (1) {
  ioctl(fd, WDIOC_KEEPALIVE, 0);
  sleep(1);
}
```

5.13 PWM Instructions

5.13.1 Preparation

- Use the kernel released in the SDK.

5.13.1.1 Operating Process

- Insert module: insmod athena2_pwm.ko;
- Run the PMW read and write command under the console or write the PWM read and write program in the kernel or user mode, you can perform PWM input and output operation;
- PWM operation in the fixed frequency clock 100MHz, a total of 20, each can be controlled separately;
- Athena2 has 20 PWM IP (pwmchip0/ pwmchip4/ pwmchip8/ pwmchip12/ pwmchip16), each IP can control 5 signals, a total of 20 signals can be controlled. The circuit diagram is represented by pwm0 to pwm15.

  In Linux sysfs, the pwm0 to pwm3 device nodes are as follows:

  /sys/class/pwm/pwmchip0/pwm0~3

  In Linux sysfs, the pwm4 to pwm7 device nodes are as follows:

  /sys/class/pwm/pwmchip:mark:4/pwm0~3

  And so on.

5.13.2 Operating Example

5.13.2.1 Example of PWM Operation Commands

Step 1.

Run the echo command on the control panel to configure the PWM number to be operated.

echo 1 > /sys/class/pwm/pwmchip0/export

Step 2.

Set the duration of a PWM cycle in ns.

echo 1000000 >/sys/class/pwm/pwmchip0/pwm1/period

Step 3.

Set the "ON" time in a period in ns. Duty cycle =duty_cycle/period=50%.

echo 500000 >/sys/class/pwm/pwmchip0/pwm1/duty_cycle

Step 4.

Enable PWM.

echo 1 >/sys/class/pwm/pwmchip0/pwm1/enable

5.13.2.2 Flile I/O Operating Program Example

User mode GPIO read and write operation example

Step 1. Configure the PWM number.

```
fd = open("/sys/class/pwm/pwmchip0/export", O_WRONLY);
if(fd < 0)
{
dbmsg("open export error\n");
return -1;
}
ret = write(fd, "1", strlen("0"));
if(ret < 0)
{
dbmsg("Export pwm1 error\n");
return -1;
}
```

Step 2. Set the duration of a PWM cycle in ns.

```
fd_period = open("/sys/class/pwm/pwmchip0/pwm1/period", O_RDWR);
ret = write(fd_period, "1000000" ,strlen("1000000" ));
if(ret < 0)
{
dbmsg("Set period error\n");
return -1;
}
```

Step 3. Set the "ON" time in a period in ns. (In this example, the duty cycle is 50%)

```
fd_duty = open("/sys/class/pwm/pwmchip0/pwm1/duty_cycle", O_RDWR);
ret = write(fd_duty, "500000", strlen("500000"));
if(ret < 0)
{
dbmsg("Set period error\n");
return -1;
}
```

Step 4. Enable PWM.

```
fd_enable = open("/sys/class/pwm/pwmchip0/pwm1/enable", O_RDWR);
ret = write(fd_enable, "1", strlen("1"));
if(ret < 0)
{
  dbmsg("enable pwm0 error\n");
  return -1;
}
```

5.14 ADC Instructions

5.14.1 Preparation

- Use the kernel released in the SDK.

5.14.1.1 Operating Process

- Insert module: insmod athena2_saradc.ko.
- Run ADC read and write command under the console or write ADC read and write program in kernel mode or user mode, you can perform ADC input and output operation.
- The user layer accesses the IIO interface to implement the 5-channel, 12-bit ADC triggering, sampling and other operations.
- 1.5v reference voltage.
- The adc pin corresponds to the sysfs file as follows:

  adc1 corresponds to in_voltage1_raw
  adc2 corresponds to in_voltage2_raw
  adc3 corresponds to in_voltage3_raw
  sar0 corresponds to in_voltage4_raw
  sar1 corresponds to in_voltage5_raw

- Voltage calculation formula: vol = val * 1500 / 4096, unit: mV.

5.14.2 Operating Example

5.14.2.1 Example of ADC Operation Commands

Step 1. | Specifies the ADC channel, in this example ADC 1.

echo 1 > /sys/bus/iio/devices/iio:device0/in_voltage1_raw

Step 2.

Read the specified ADC channel value.

cat /sys/bus/iio/devices/iio:device0/in_voltage1_raw

5.14.2.2 Example of User Mode ADC Read and Write Operations

Example of user mode ADC read and write operations:

```
fd = open("/sys/bus/iio/devices/iio:device0/in_voltage1_raw", O_RDWR|
O_NOCTTY|O_NDELAY);
if (fd < 0)
    printf("open adc err!\n");

write(fd, "1", 1);
lseek(fd, -1, SEEK_CUR);


char buffer[512] = {0};
int len = 0;
unsigned int adc_value = 0;

len = read(fd, buffer, 5);
if (len != 0) {
    printf("read buf: %s\n", buffer);
    adc_value= atoi(buffer);
    printf("adc value is %d\n", adc_value);
}
write(fd, "0", 1);
close(fd);
```

5.15 PINMUX Instructions

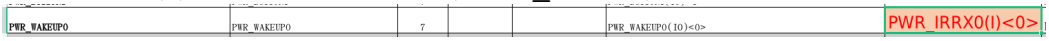
5.15.1 Set pinmux in u-boot

The pinmux header file is in directory u-boot-2021.10/board/cvitek/athena2/pinmux. To configure pinmux in uboot, you need to include athena2_pinmux.h.

- **PINMUX_CONFIG(PIN_NAME, FUNC_NAME, GROUP):**
  - PIN_NAME: Corresponds to the Signal Name in the first column of the pinlist document;
  - FUNC_NAME: Corresponding to the function to be switched in the pinlist document;
  - GROUP: The group where this function resides corresponds to the group column in the pinlist document (for details, see group_pin_t enum).

For example, switch the PWR_WAKEUP0 pin' s function to PWR_IRRX0:

- **First, go to pinlist and locate the row where the PWR_WAKEUP0 pin is located, and locate group(7) and target function(PWR_IRRX0):**



- Set pinmux in the code with PINMUX_CONFIG(PWR_WAKEUP0, PWR_IRRX0, G7);
- Recompile u-boot, burn firmware and restart;

5.15.2 Set pinmux in kernel

Pinmux header files is in directory linux_5.10/drivers/pinctrl/cvitek/pinctrl-athena2.h.

5.15.3 Set pinmux in userspace

Userspace provides the cvi_pinmux tool, which supports setting pinmux, internal drop-down, and driver capabilities. Use the following method:

```
cvi_pinmux for Athena2
./cvi_pinmux -p          <== List all pins
./cvi_pinmux -l          <== List all pins and its func
./cvi_pinmux -r pin      <== Get func from pin
./cvi_pinmux -w pin/func <== Set func to pin
./cvi_pinmux -c <pin name>,<0 or 1 or 2>  <== Set pin pull up/down (0:pull down; 1:pull up;F
↪2:pull off)
./cvi_pinmux -d <pin name>,<0 ~ 15>  <== Set pin driving
```

For example, to set the PWR_WAKEUP0 pin to the PWR_IRRX0 function and set its internal pull-up, the drive capability is set to 15 (the larger the value from 0 to 15, the stronger the drive capability is):

- cvi_pinmux -w PWR_WAKEUP0/PWR_IRRX0
- cvi_pinmux -c PWR_WAKEUP0,1
- cvi_pinmux -d PWR_WAKEUP0,15

In addition, you can list all pins and their functions on the soc with "cvi_pinmux -l" , or read the functions contained in the corresponding pin with "cvi_pinmux -r pin_name" .

5.16 BM1688 Built-in MCU Function Description

5.16.1 BM1688 Built-in MCU Load Program and Startup

Built-in 8051 MCU firmware has been placed on "FSBL/platt/athena2 / prebuilt/athena2_mcu_fw bin" . When the fsbl is compiled, the firmware is encoded in fip.bin. The MCU firmware will load during the bl2 phase. After entering the system, you can check whether the MCU is running using the following command.

```
busybox devmem 0x05025018
```

Use this command to check whether bit1 of the register is 1. If it is 1, the MCU is started. Write this bit to 0 to shut down the MCU.

5.16.2 Built-in MCU Functions

At present, the built-in MCU undertakes two parts of the function, one is responsible for some functions of ddr retrain. The second is the related needs of buttons and leds. The following is an introduction to the mcu keys and led functions.

- The following code runs only on SE9. MCU determines if it is SE9 by reading the OEM' s product parameters. If yes, enable the following functions.
- STAT indicator turns red during system loading: MCU turns pwr_gpio4 low to turn off the green light and pwr_gpio5 high to turn on the red light.
- SSD indicator turns red during system loading: MCU turns pwr_gpio3 low to turn off the green light and pwr_gpio6 high to turn on the red light.
- The power indicator turns on after the power is turned on, and turns off after the power is turned off: MCU turns pwr_gpio1 down to turn off and up to turn on.
- Press and hold power button for 2s to power off: MCU detects a low level of pin PWR_BUTTON1 for 2s, notifying the system layer to power off. Three lights off.
- Press and hold power button for 5s to power off: MCU detects the low level of PWR_BUTTON1 for 5s, and then controls the system to power off. Three lights off.
- Press power button to power on: MCU controls the system power on, the power indicator light green and other indicators are red.

- Press and hold reset button for 12s to restore factory settings: MCU detects a PWR_ON low level of 12s, notifies the system layer to restore factory settings and restart. STAT and SSD light red.
- Press reset button to restart: MCU detects a low pin level and then controls a system hot restart. STAT and SSD light red.

The MCU writes 1 to 0x0502601c to notify the system layer software to power off. Write 2 Notify the system layer to restore factory defaults. In addition, the MCU records the power-on and power-off reasons, which can be read from register 0x05026030, and the 0-3bit records the power-off reasons.

- 0x1: Press reset button.
- 0x2: Press and hold reset button for 12s.
- 0x4: Press and hold power button for 2s.
- 0x5: Press and hold power button for 5s.
- 0x6: Overheat power-off.

4~7bit record the power-on cause.

- 0x1: Press reset button.
- 0x2: Press and hold on reset button.
- 0x3: Press power button.

5.16.3 The MCU Determines whether to Enable Related Functions Based on the OEM

The MCU determines whether to execute the key and led code based on the product parameter in the OEM. The fsbl stage reads the last 256 bytes of the MCU' s sram from the hardware boot1 partition of the EMMC. See OEM instructions for details. The MCU reads it. If the value of the product parameter is SE9, related functions are enabled. You can enter the kernel and use the following command to write SE9 to the OEM product parameter:

```
echo 0 > /sys/block/mmcblk0boot1/force_ro
echo "SE9" > se9.txt
dd if=se9.txt of=/dev/mmcblk0boot1 count=4 bs=1 seek=208
echo 1 > /sys/block/mmcblk0boot1/force_ro
```

5.17 CAN Instructions

5.17.1 Preparation

Preparations for CAN are as follows:

Use the kernel released in the SDK.

5.17.2 Operating Process

Load the kernel. By default, all CAN drivers have been programmed into the kernel, and no loading command is required.

Configure the can baud rate using the ip command on the console. Use cansend to send data and candump to receive data.

You can run the "ifconfig -a" command to view the network nodes of the CAN.

```
[root@cvitek]~#
[root@cvitek]~#
[root@cvitek]~# ifconfig -a
can0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          NOARP  MTU:16  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:10
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:29

eth0      Link encap:Ethernet  HWaddr 06:CC:E6:ED:80:9B
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

5.17.3 Examples of CAN Read and Write Commands

Set CAN2.0 A/B baud rate to 250K.

```
ip link set can0 up type can bitrate 250000
```

Set CANFD baud rate to 250K.

```
ip link set can0 type can bitrate 250000 sample-point 0.8 dbitrate 250000 dsample-point 0.75 fd on
```

Open CAN device.

```
ifconfig can0 up
```

Receive CAN data.

```
candump can0
```

```
[root@cvitek]/mnt/sd/sample#
[root@cvitek]/mnt/sd/sample# candump can0
interface = can0, family = 29, type = 3, proto = 1
<0x237> [4] 12 34 34 56
<0x237> [4] 12 34 34 56
```

Send CAN data.

```
cansend can0 -i 0x55 0x12 0x13 0x14 0x13
```

-i indicates the ID of the frame sent by A. 0x55 indicates the CAN ID.

5.17.4 Operating Example

Examples of kernel-mode CAN read and write programs:

This action example shows how to receive CAN data in kernel state.

Step 1. Call alloc_can_skb to obtain the skb of the CAN network node.

Step 2. Determine the frame type and call receive_remote_frame or receive_frame to read the FIFO data and pass the received data to cf.

```
static void sdvt_can_read_fifo(struct net_device *dev)
{
        struct net_device_stats *stats = &dev->stats;
        struct sdvt_can_classdev *cdev = netdev_priv(dev);
        struct canfd_frame *cf;
        struct sk_buff *skb;
        int m_i_i   = 0;



        skb = alloc_can_skb(dev, (struct can_frame **)&cf);
```

(continues on next page)

(continued from previous page)

```
        // cdev->ops->write_reg(cdev,SDVT_CAN_FIFO_FLUSH,0x2);
        if (!skb) {
                stats->rx_dropped++;
                return;
        }

        if(cmd_o.irq_status0_8b & (1 << SDVT_CAN_IRQ_REMOTE_FRAME)){
                // Read the length from RX LENGTH FIFO
                receive_remote_frame(cdev,&cmd_o,&cfg_o);
                cf->can_id = cmd_o.ident_32b | CAN_RTR_FLAG;
                netdev_dbg(dev, "remote frame\n");
        }else{
                receive_frame(cdev,&cmd_o,&cfg_o);
                cf->len =  cmd_o.rx_len_2d_8b[1];
                cf->can_id =  cmd_o.ident_32b;
                for (m_i_i = 0; m_i_i < cf->len; m_i_i += 1) {
                        ((u_int8_t *)cf->data)[m_i_i] = cmd_o.rx_data_2d_8b[m_i_i];
                }
        }

        cmd_o.irq_status1_8b = 0x00;
        cmd_o.irq_status0_8b = 0x00;
        dev_info(cdev->dev,"\n");

        stats->rx_packets++;
        stats->rx_bytes += cf->len;
        netif_receive_skb(skb);
}
```

This action example shows how the kernel state sends CAN data:

Step 1. Get structure cf.

Step 2. Assign len obtained from the struct to cmd_o.dlc_4b, data to cmd_o.data_2d_8b, and then write these values into the can FIFO via the send_command(cdev,&cmd_o,&cfg_o) function.

```
static netdev_tx_t stvd_can_tx_handler(struct sdvt_can_classdev *cdev)
{
        struct canfd_frame *cf = (struct canfd_frame *)cdev->tx_skb->data;
        int m_i_i   = 0;
        cmd_o.data_len_code_4b  = cf->len;
        cmd_o.dlc_4b = cmd_o.data_len_code_4b;

        pr_info("send data:");
        for (m_i_i = 0; m_i_i < cmd_o.data_len_code_4b; m_i_i += 1) {
                cmd_o.data_2d_8b[m_i_i] = ((u_int8_t *)cf->data)[m_i_i];
                pr_info(" get xmit data %#x", cmd_o.data_2d_8b[m_i_i]);
        }
        pr_info("\n");
```

(continues on next page)

(continued from previous page)

```
        send_command(cdev,&cmd_o,&cfg_o);

        return NETDEV_TX_OK;
}
```

User mode CAN Read and write program Example CAN user mode data receiving uses the socket mode

CAN sends and receives data in user-mode, uses the socket() function to create a socket descriptor, but pays attention to the socket' s domain,type, and protocol Settings, and then uses bind() to assign a specific address in an address family to the socket. After the connection is established, use write/read to read and write data.

```
int main(int argc, char **argv)
{
        struct can_frame frame = {
                .can_id = 1,
        };
        struct ifreq ifr;
        struct sockaddr_can addr;
        char *interface;
        int family = PF_CAN, type = SOCK_RAW, proto = CAN_RAW;
        int loopcount = 1, infinite = 0;
        int s, opt, ret, i,nbytes, err,dlc = 0, rtr = 0, extended = 0;
        int verbose = 0;
        int count = 0;
        int read_enable =0;
        char buf[BUF_SIZ];
        unsigned char send_data[8] = {0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08};

        interface = "can0";

        printf("interface = %s, family = %d, type = %d, proto = %d\n",
                interface, family, type, proto);

        s = socket(family, type, proto);
        if (s < 0) {
                perror("socket");
                return 1;
        }

        addr.can_family = family;
        strcpy(ifr.ifr_name, interface);
        if (ioctl(s, SIOCGIFINDEX, &ifr)) {
                perror("ioctl");
                return 1;
        }
        addr.can_ifindex = ifr.ifr_ifindex;

        if (bind(s, (struct sockaddr *)&addr, sizeof(addr)) < 0) {
```

(continues on next page)

(continued from previous page)

```
        perror("bind");
        return 1;
    }

    for (i = 0; i < sizeof(send_data); i++) {
        frame.data[dlc] = send_data[i];
        dlc++;
    }
    frame.can_dlc = dlc;


    if (extended) {
        frame.can_id &= CAN_EFF_MASK;
        frame.can_id |= CAN_EFF_FLAG;
    } else {
        frame.can_id &= CAN_SFF_MASK;
    }

    if (rtr)
        frame.can_id |= CAN_RTR_FLAG;

    ret = write(s, &frame, sizeof(frame));

    read_enable = 1;
    while (read_enable) {
        if ((nbytes = read(s, &frame, sizeof(struct can_frame))) < 0) {
            perror("read");
            return 1;
        } else {
            if (frame.can_id & CAN_EFF_FLAG)
                n = snprintf(buf, BUF_SIZ, "<0x%08x> ", frame.can_id & CAN_EFF_
↪MASK);
            else
                n = snprintf(buf, BUF_SIZ, "<0x%03x> ", frame.can_id & CAN_SFF_
↪MASK);

            n += snprintf(buf + n, BUF_SIZ - n, "[%d] ", frame.can_dlc);
            for (i = 0; i < frame.can_dlc; i++) {
                n += snprintf(buf + n, BUF_SIZ - n, "%02x ", frame.data[i]);
            }
            if (frame.can_id & CAN_RTR_FLAG)
                n += snprintf(buf + n, BUF_SIZ - n, "remote request");

            fprintf(out, "%s\n", buf);

            do {
                err = fflush(out);
                if (err == -1 && errno == EPIPE) {
                    err = -EPIPE;
                    fclose(out);
                    out = fopen(optout, "a");
```

(continues on next page)

(continued from previous page)

```
                        if (!out)
                                exit (EXIT_FAILURE);
                        }
                } while (err == -EPIPE);

                n = 0;
        }
    }


    close(s);
    return 0;
}
```

5.18 4G/5G Module Operation

5.18.1 Check USB Device Enumeration

First, you need to confirm whether the system has recognized and enumerated the connected USB device. Use the lsusb command to list all connected USB devices:

```
lsusb
```

If the command output shows your device information, it means the device has been successfully recognized by the system. For example, the following output shows a Fibocom wireless module that has been enumerated by the system:

```
Bus 004 Device 002: ID 2cb7:0a06 Fibocom Wireless
```

In this example, we focus on the Fibocom FM650 module.

5.18.2 Verify Network Interface Registration

Next, you need to confirm whether the network interface has been registered in the system. Run the ifconfig -a command to list all network interfaces, including those that are not yet configured:

```
ifconfig -a
```

Check the output to ensure that your network interface is listed, indicating that the interface has been registered in the system, even if it may not yet have an IP address configured.

5.18.3 Check USB Serial Ports

```
# List all connected USB serial devices
ls /dev/ttyUSB*

# In this environment, executing the above command should display the following device list:
# Note: These device details indicate the bus location and device ID of each USB device

Bus 004 Device 002: ID 2cb7:0a06 Fibocom Wireless Inc. FM650 Module
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub

# If your device does not appear in the list, you need to troubleshoot:
# Confirm whether the USB device is properly connected to the computer.
# Check if the module is correctly inserted into the adapter and whether the adapter has any physical damage.
```

5.18.4 Start Dial-up Service (Supported Models Only)

```
systemctl start lteModemManager
```

After waiting for a while, if successfully started, you can use the ifconfig -a command to check the network card' s IP address.

5.18.5 Verify Internet Connection

```
# The lteModemManager service will automatically configure DNS to 8.8.8.8 within about 60 seconds after starting
systemd-resolved --status

# The output should be similar to the following:
Link 8 (usb0)
    Current Scopes: DNS
DefaultRoute setting: yes
    LLMNR setting: yes
MulticastDNS setting: no
DNSOverTLS setting: no
    DNSSEC setting: no
DNSSEC supported: no
Current DNS Server: 8.8.8.8
    DNS Servers: 8.8.8.8

# You can also manually set the DNS as follows:
# Edit DNS settings
vim /etc/systemd/resolved.conf
```

(continues on next page)

(continued from previous page)

```
# In the file, find the [Resolve] section and set the DNS to Google's public DNS
[Resolve]
DNS=8.8.8.8

# Save and exit, then restart the systemd-resolved service to apply the changes
systemctl restart systemd-resolved

# Check if the configuration is effective by trying to ping a website
ping google.com
```

Note: The SIM card does not support hot-swapping and must be inserted before powering on.

5.19 WiFi and Bluetooth Operation Guide

5.19.1 WiFi/BT Operation Procedure

5.19.1.1 Confirm PCIe Enumeration

```
# Use the lspci command to list all PCI devices
lspci

# If the system is functioning correctly, you will see an output similar to the following:
# This output includes a list of PCI devices and related information

00:00.0 PCI bridge: Device 1f1c:186a (rev 01)
01:00.0 Network controller: Realtek Semiconductor Co., Ltd. Device b852
```

5.19.1.2 Load WiFi Driver

```
insmod /mnt/system/ko/3rd/8852be.ko
```

5.19.1.3 Load Bluetooth Driver

```
insmod /mnt/system/ko/3rd/hci_uart.ko
```

Note: The above kernel modules already exist and only need to be loaded.

5.19.1.4 Install the corresponding wpa and bluez tools

```
dpkg -i wpa_xxx.deb bluez_xxx.deb
```

These two packages can be installed using apt install based on the module' s official website information.

5.19.1.5 Configure wpa_supplicant

To enable your system to automatically connect to a wireless network, you need to create a configuration file named wpa_supplicant.conf in the /etc/wpa_supplicant/ directory. The following is a configuration example that will configure your system to automatically connect to a wireless network named "123" with the password "12345678" :

```
# wpa_supplicant global control interface settings
ctrl_interface=/var/run/wpa_supplicant

# Network configuration block
network={
    ssid="123"          # Wireless network name (SSID)
    psk="12345678"      # Wireless network pre-shared key (password)
}
```

5.19.1.6 Start WPA Service

```
systemctl start wpa_supplicant
```

5.19.1.7 Start dhclient Service to Automatically Obtain IP

```
# Start dhclient to automatically obtain an IP address
systemctl start dhclient

# After successfully obtaining an IP address, you can use the ifconfig -a command to view
↪network interface information
# The following is an example of the output after executing this command in this environment:

wlp1s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.137.175  netmask 255.255.255.0  broadcast 192.168.137.255
    inet6 fe80::7a8a:86ff:fe51:59d6  prefixlen 64  scopeid 0x20<link>
    ether 78:8a:86:51:59:d6  txqueuelen 1000  (Ethernet)
    RX packets 450  bytes 85032 (85.0 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 193  bytes 15220 (15.2 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```


5.19.1.8 Test Network Connectivity

```
# Manually configure DNS
vim /etc/systemd/resolved.conf

# In the file, find the [Resolve] section and set the DNS to Google's public DNS
[Resolve]
DNS=8.8.8.8

# Save and exit, then restart the systemd-resolved service to apply the changes
systemctl restart systemd-resolved

# Check if the configuration is effective by trying to ping a website
ping google.com
```

5.19.1.9 Configure Bluetooth (Confirm UART Pins and Perform Pinmux Switching)

```
# Configure UART pins
cvi_pinmux -w UART4_RX/UART4_RX
cvi_pinmux -w UART4_TX/UART4_TX
cvi_pinmux -w UART4_CTS/UART4_CTS
cvi_pinmux -w UART4_RTS/UART4_RTS

# Start Bluetooth HCI
rtk_hciattach -n -s 115200 ttyS4 rtk_h5 &

# If Bluetooth initialization fails, you need to power cycle
# Reset Bluetooth
cvi_pinmux -w PAD_MIPI1_TX0P/GPIO185
echo 345 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio345/direction
echo 1 > /sys/class/gpio/gpio345/value
echo 0 > /sys/class/gpio/gpio345/value
```

5.19.1.10 Check if Bluetooth Node is Generated

```
# Check Bluetooth nodes
hciconfig -a

# Expected output example
hci0:   Type: Primary  Bus: UART
    BD Address: 78:8A:86:51:59:D7  ACL MTU: 1021:5  SCO MTU: 255:11
    UP RUNNING PSCAN
    RX bytes:912735 acl:38 sco:0 events:4708 errors:0
    TX bytes:44343 acl:37 sco:0 commands:218 errors:0
    Features: 0xff 0xff 0xff 0xfa 0xdb 0xbf 0x7b 0x87
    Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
```

(continues on next page)

(continued from previous page)

```
    Link policy: RSWITCH HOLD SNIFF PARK
    Link mode: SLAVE ACCEPT
```

5.19.1.11 Start Bluetooth

```
# Start Bluetooth service
systemctl start bluetooth
```

5.19.1.12 Use BlueZ Tools for Communication

```
# Enter Bluetooth command line interface
bluetoothctl

# Command line display
Agent registered
[CHG] Controller 78:8A:86:51:59:D7 Pairable: yes
[bluetooth]#

# Turn on Bluetooth power
power on

# Expected output
[bluetooth]# power on
Changing power on succeeded
[bluetooth]#

# Scan for nearby Bluetooth devices
scan on

# Expected output
[bluetooth]# scan on
Discovery started
[CHG] Controller 78:8A:86:51:59:D7 Discovering: yes
[NEW] Device 7B:4F:A0:69:9F:CB 7B-4F-A0-69-9F-CB
[NEW] Device 73:D7:AE:07:F8:79 73-D7-AE-07-F8-79
[NEW] Device 68:F3:5C:FD:59:25 68-F3-5C-FD-59-25
[NEW] Device 1C:3A:C6:A6:AC:B2 1C-3A-C6-A6-AC-B2
[NEW] Device 5A:27:AF:9D:F0:BE 5A-27-AF-9D:F0-BE

# Pair with a Bluetooth device
pair <XX:XX:XX:XX:XX:XX>

# Expected output
pair F4:1A:9C:BA:30:E3
Attempting to pair with F4:1A:9C:BA:30:E3
[CHG] Device F4:1A:9C:BA:30:E3 Connected: yes
Request confirmation
```

(continues on next page)

(continued from previous page)

```
[agent] Confirm passkey 442308 (yes/no): yes
[CHG] Device F4:1A:9C:BA:30:E3 Modalias: bluetooth:v038Fp1200d1436
[CHG] Device F4:1A:9C:BA:30:E3 UUIDs: 00001105-0000-1000-8000-00805f9b34fb
[CHG] Device F4:1A:9C:BA:30:E3 UUIDs: 0000110a-0000-1000-8000-00805f9b34fb
[CHG] Device F4:1A:9C:BA:30:E3 UUIDs: 0000110c-0000-1000-8000-00805f9b34fb
[CHG] Device F4:1A:9C:BA:30:E3 UUIDs: 00001112-0000-1000-8000-00805f9b34fb
[CHG] Device F4:1A:9C:BA:30:E3 UUIDs: 00001115-0000-1000-8000-00805f9b34fb
[CHG] Device F4:1A:9C:BA:30:E3 UUIDs: 00001116-0000-1000-8000-00805f9b34fb
[CHG] Device F4:1A:9C:BA:30:E3 UUIDs: 0000111f-0000-1000-8000-00805f9b34fb
[CHG] Device F4:1A:9C:BA:30:E3 UUIDs: 0000112f-0000-1000-8000-00805f9b34fb
[CHG] Device F4:1A:9C:BA:30:E3 UUIDs: 00001132-0000-1000-8000-00805f9b34fb
[CHG] Device F4:1A:9C:BA:30:E3 UUIDs: 00001200-0000-1000-8000-00805f9b34fb
[CHG] Device F4:1A:9C:BA:30:E3 UUIDs: 00001800-0000-1000-8000-00805f9b34fb
[CHG] Device F4:1A:9C:BA:30:E3 UUIDs: 00001801-0000-1000-8000-00805f9b34fb
[CHG] Device F4:1A:9C:BA:30:E3 UUIDs: 00001855-0000-1000-8000-00805f9b34fb
[CHG] Device F4:1A:9C:BA:30:E3 UUIDs: 0000fdaa-0000-1000-8000-00805f9b34fb
[CHG] Device F4:1A:9C:BA:30:E3 UUIDs: 98b97136-36a2-11ea-8467-484d7e99a198
[CHG] Device F4:1A:9C:BA:30:E3 UUIDs: ada499be-27d6-11ec-9427-0a80ff2603de
[CHG] Device F4:1A:9C:BA:30:E3 ServicesResolved: yes
[CHG] Device F4:1A:9C:BA:30:E3 Paired: yes
Pairing successful
```

5.19.2 WiFi AP Operation Procedure

5.19.2.1 Set Up a 5G Wireless Access Point (AP)

To set up a 5G AP, you need to use the US country code:

```
insmod /mnt/system/ko/3rd/8852be.ko rtw_country_code=US
sudo apt install iw
iw reg set US
ifconfig wlp1s0 192.168.1.10
```

5.19.2.2 Install DHCP Server

For most Linux distributions, you can use the package manager to install isc-dhcp-server. For example, on Debian/Ubuntu, you can use the following commands:

```
sudo apt update
sudo apt install isc-dhcp-server
```


5.19.2.3 Configure DHCP Server

You need to edit the DHCP server' s configuration file, typically located at /etc/dhcp/dhcpd.conf, to define the IP address range and other network settings. The following is a configuration example:

```
subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.11 192.168.1.254;
    option domain-name-servers 8.8.8.8, 8.8.4.4;
    option routers 192.168.1.1;
}
```

5.19.2.4 Specify Listening Interface

In the /etc/default/isc-dhcp-server file, specify the network interface that the DHCP server should listen on:

```
INTERFACESv4="wlp1s0"
INTERFACESv6=""
```

Start the service:

```
systemctl start isc-dhcp-server
```

5.19.2.5 Set Up Wireless Access Point

You can use hostapd to set up a wireless access point. First, install hostapd:

```
sudo apt install hostapd
```

It is best to disable auto-start after installation. Before configuring hostapd, it is best to disable its automatic startup to avoid starting it automatically when the configuration file is not fully set up:

```
sudo systemctl unmask hostapd
sudo systemctl disable hostapd
```

Then, create a configuration file named /etc/hostapd/hostapd.conf with the following content:

```
interface=wlp1s0
driver=nl80211

# Control interface directory
ctrl_interface=/var/run/hostapd
# Control interface group
ctrl_interface_group=0
```

(continues on next page)

(continued from previous page)

```
ssid=SE9
wpa=2
wpa_passphrase=12345678
wpa_key_mgmt=WPA-PSK
#ieee80211d=1
#ieee80211n=1
ieee80211ac=1

hw_mode=a
channel=40

# Enable channel bonding as above
ht_capab=[MAX-AMSDU-3839][HT40-][SHORT-GI-20][SHORT-GI-40][DSSS_CCK-40]
require_ht=1

# Explanation from the official documentation:
# vht_capab: VHT capabilities (list of flags)
# vht_max_mpdu_len: [MAX-MPDU-7991] [MAX-MPDU-11454]
# Indicates maximum MPDU length
#
# Short GI for 80 MHz: [SHORT-GI-80]
# Indicates short GI support for reception of packets transmitted with TXVECTOR
# params format equal to VHT and CBW = 80Mhz
#
# SU Beamformee Capable: [SU-BEAMFORMEE]
# Indicates support for operation as a single user beamformee
vht_capab=[MAX-MPDU-3895][SHORT-GI-80][SU-BEAMFORMEE]

# Require stations to support VHT PHY (reject association if they do not)
require_vht=1

# 0 = 20 or 40 MHz operating Channel width
# 1 = 80 MHz channel width
# 2 = 160 MHz channel width
# 3 = 80+80 MHz channel width
vht_oper_chwidth=0

#For more details, refer to the official documentation.
beacon_int=100
dtim_period=2
max_num_sta=255
preamble=1
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wmm_enabled=1
eapol_key_index_workaround=0
eap_server=0
rsn_pairwise=CCMP
```

CHAPTER 6

System Interfaces

6.1 OEM Instructions

The location of the OEM parameter is the boot1 partition of the eMMC hardware. After entering the kernel, the parameter is mounted to /dev/mmcblk0boot1. When the MCU starts to run in the fsbl stage, OEM parameters are required for setting different boards. A53 and MCU cannot access eMMC at the same time, so before the FSBL loads the MCU, the OEM parameters are read from the boot1 partition of eMMC to RTC_SRAM, and then the MCU retrieves the parameters.

The start address of the RTC_SRAM is 0x5200000 and its length is 0x8000(32K). The last 256 bytes of the RTC_SRAM are reserved for storing OEM parameters. The OEM structure is as follows:

offset	name	description
0x00	SN0	chip sn
0x20	SN1	reserve for product or customer
0x40	MAC0	
0x50	MAC1	
0x60	PRODUCT_TYPE	
0x70	MODULE_TYPE	
0x80	INTERFACE_FLAG	
0x81	AGING_FLAG	
0x90	VENDER	
0xa0	DTS_TYPE	
0xc0	HW_VERSION	
0xd0	PRODUCT	product family, for example: SE9
0xe0	CHIP	BM1688/CV186AH

6.2 Read and Write SN and MAC Addresses

The SN and MAC address of Athena2 are stored in the EEPROM of the MCU. You can modify and read the SN and MAC address as follows:

Unlock the mmcblk0boot1 device node:

```
sudo -i
echo 0 > /sys/block/mmcblk0boot1/force_ro
```

Write SN:

```
echo "HQATEVBAIAIAI0001" > sn.txt
dd if=sn.txt of=/dev/mmcblk0boot1 count=17 bs=1
echo "HQATEVBAIAIAI0002" > sn.txt
dd if=sn.txt of=/dev/mmcblk0boot1 count=17 bs=1 seek=32
```

Read SN:

```
dd if=/dev/mmcblk0boot1 of=dump.bin count=17 bs=1
hexdump -C dump.bin | head
```

Write MAC:

```
echo "E0A509261417" > mac0.txt
xxd -p -u -r mac0.txt > mac0.bin
dd if=mac0.bin of=/dev/mmcblk0boot1 count=6 bs=1 seek=64
echo "E0A509261418" > mac1.txt
xxd -p -u -r mac1.txt > mac1.bin
dd if=mac1.bin of=/dev/mmcblk0boot1 count=6 bs=1 seek=80
```

Read MAC:

```
dd if=/dev/mmcblk0boot1 of=mac_dump.bin count=6 bs=1 skip=64
hexdump mac_dump.bin
```

Lock the mmcblk0boot1 device node again to avoid accidental rewriting:

```
echo 1 > /sys/block/mmcblk0boot1/force_ro
```

The new MAC address takes effect after the system is restarted.

6.3 Read Athena2 Chip Temperature

Command:

```
cat /sys/class/thermal/thermal_zone0/temp
```

Return value:

```
38745
```

Linux thermal framework will use this temperature for management (need to load soph_clock_cooling driver):

- When the temperature rises to 110 degrees, the TPU frequency will be reduced to 450MHz, and the CPU frequency will be reduced to 1GHz;
- When the temperature rises to 120 degrees, the TPU frequency drops to 100 MHZ, and the CPU frequency drops to 1GHz.
- When the temperature falls back to 100 degrees, the TPU and CPU frequency return to the rated value;
- When the temperature rises to 125 degrees, the system shuts down.

Support pwm control fan for active cooling, the default strategy is:

- When the temperature is below 40 degrees, the fan does not rotate;
- When the temperature rises to 40 degrees, the fan runs at the maximum speed of 100/255;
- When the temperature rises to 60 degrees, the fan runs at the maximum speed of 170/255;
- When the temperature rises to 80 degrees, the fan runs at the maximum speed.

Note: The above policy can be modified through the device tree in the soph_base.dtsi file:

```
athena2_cooling:athena2-cooling {
      clocks = <&clk ATHENA2_AP_CPU_CLK>, <&clk ATHENA2_TPU_CLK_
↪TPU>;
```

(continues on next page)

(continued from previous page)

```
        clock-names = "clk_cpu", "clk_tpu_axi";
        dev-freqs = <1650000000 900000000>,
                <1000000000 450000000>,
                <1000000000 100000000>;
        compatible = "cvitek,athena2-cooling";
        #cooling-cells = <2>;
};

fan0: pwm-fan {
        compatible = "pwm-fan";
        #cooling-cells = <2>;
        pwms = <&pwm0 0 1000000 0>; // default pwm0, 1ms period, normal polarity
        cooling-levels = <1 100 170 255>; // max 255
};


thermal-zones {
        soc_thermal_0: soc_thermal_0 {
                polling-delay-passive = <1000>; /* milliseconds */
                polling-delay = <1000>; /* milliseconds */
                thermal-sensors = <&thermal 0>;

                trips {
                        soc_thermal_trip_0: soc_thermal_trip_0 {
                                temperature = <110000>; /* millicelsius */
                                hysteresis = <10000>; /* millicelsius */
                                type = "passive";
                        };

                        soc_thermal_trip_1: soc_thermal_trip_1 {
                                temperature = <120000>; /* millicelsius */
                                hysteresis = <20000>; /* millicelsius */
                                type = "passive";
                        };

                        soc_thermal_crtical_0: soc_thermal_crtical_0 {
                                temperature = <125000>; /* millicelsius */
                                hysteresis = <0>; /* millicelsius */
                                type = "critical";
                        };

                        soc_thermal_active_0: soc_thermal_active_0 {
                                temperature = <40000>; /* millicelsius */
                                hysteresis = <0>; /* millicelsius */
                                type = "active";
                        };

                        soc_thermal_active_1: soc_thermal_active_1 {
                                temperature = <60000>; /* millicelsius */
                                hysteresis = <0>; /* millicelsius */
                                type = "active";
```

(continues on next page)

(continued from previous page)

```
                };

                soc_thermal_active_2: soc_thermal_active_2 {
                        temperature = <80000>; /* millicelsius */
                        hysteresis = <0>; /* millicelsius */
                        type = "active";
                };
        };

        cooling-maps {
                map0 {
                        trip = <&soc_thermal_trip_0>;
                        cooling-device = <&athena2_cooling 1 1>;
                };

                map1 {
                        trip = <&soc_thermal_trip_1>;
                        cooling-device = <&athena2_cooling 2 2>;
                };

                map2 {
                        trip = <&soc_thermal_active_0>;
                        cooling-device = <&fan0 1 1>;
                };

                map3 {
                        trip = <&soc_thermal_active_1>;
                        cooling-device = <&fan0 2 2>;
                };

                map4 {
                        trip = <&soc_thermal_active_2>;
                        cooling-device = <&fan0 3 3>;
                };
        };
};
```

Modify the above device tree nodes to customize the temperature control policy (TPU frequency division can only be integer multiples), see linux Documention related device tree description.

6.4 Read Power Information

To be updated···

6.5 Query the Memory Usage

Athena2 is equipped with 16GB DDR, which can be divided into three categories:

1. The part of kernel management can be allocated for use by malloc, kmalloc and other conventional apis.
2. The ION management part is reserved for TPU, VPU, and VPP. Use the ionctl interface of ION or the interface provided by bmlib in sophonSDK to allocate the ION management part.
3. The parts reserved for firmware cannot be used by users.

You can check the memory usage of each part in the following ways:

1. View system memory

```
linaro@bm1684:~$ free -h
               total        used        free      shared  buff/cache   available
Mem:           6.6Gi       230Mi       6.2Gi       1.0Mi       230Mi       6.3Gi
Swap:            0B          0B          0B
```

2. View ION memory

```
sudo -i
root@bm1684:~# cat /sys/kernel/debug/ion/cvi_npu_heap_dump/summary  | head -2
Summary:
[0] npu heap size:4141875200 bytes, used:0 bytes        usage rate:0%, memory usage peak
0 bytes

root@bm1684:~# cat /sys/kernel/debug/ion/cvi_vpp_heap_dump/summary  | head -2
Summary:
[2] vpu heap size:2147483648 bytes, used:0 bytes        usage rate:0%, memory usage peak
0 bytes
```

   As above, there are usually 2 ions. heap (i.e., two reserved memory areas), as the name suggests, are used by the TPU and VPP respectively. In the example above, only the beginning of each heap usage information is printed. If you complete the cat summary file, you can see the address and size information for each allocated buffer.

CHAPTER 7

System Customization

7.1 File Structure

To be updated···

7.2 Cross Compilation

To be updated···

7.3 Modify Kernel

To be updated···

7.4 Modify Ubuntu 20.04

To be updated···

7.5 Ubuntu-desktop Installation

1. Change the download source to Alibaba Cloud to speed up the installation.

```
sudo cp /etc/apt/sources.list /etc/apt/sources.list.bak
sudo sed -i 's/ports.ubuntu.com/mirrors.aliyun.com/g' /etc/apt/sources.list
sudo apt update
```

2. ubuntu desktop may conflict with SophonUI, so you need to close SophonUI first, the command is as follows:

```
sudo systemctl disable SophonHDMI.service
```

3. Install ubuntu-desktop.

```
sudo apt install -y ubuntu-desktop
```

4. Restart.

```
sudo reboot
```

5. After installing the desktop, if you want to temporarily shut down the ubuntu desktop, run the following command to restart it.

```
sudo systemctl set-default multi-user
```

6. To restart the desktop, run the following command for the restart to take effect.

```
sudo systemctl set-default graphical
```

7.6 Customized Software Package

You can obtain the specific packages you need by:

1. Obtain the sdcard.tgz basic software package from the official website.
2. You need to copy the sdcard.tgz software package to the $OUTPUT_DIR directory by referring to the file structure section. The directory indicated by $OUTPUT_DIR is related to the configuration specified by the defconfig command, for example:

```
source build/cvisetup.sh

defconfig bm1688_wevb_emmc
```

The terminal will print the following content:

```
Run defconfig function
Loaded configuration '/data/sdk_release_1.3/build/boards/athena2/
```

(continues on next page)

(continued from previous page)

```
↪bm1688_wevb_emmc/bm1688_wevb_emmc_defconfig'
No change to configuration in '.config'
Loaded configuration '.config'
No change to minimal configuration in '/data/sdk_release_1.3/build/.
↪defconfig'
/data/sdk_release_1.3/build /data/sdk_release_1.3
/data/sdk_release_1.3

====== Environment Variables =======

PROJECT: bm1688_wevb_emmc, DDR_CFG=ddr4_3200_x16_2s
CHIP_ARCH: ATHENA2, DEBUG=0
SDK VERSION: 64bit, RPC=0
ATF options: ATF_KEY_SEL=default, BL32=1
Linux source folder:linux_5.10, Uboot source folder: u-boot-2021.10
CROSS_COMPILE_PREFIX: aarch64-linux-gnu-
ENABLE_BOOTLOGO: 0
Flash layout xml: /data/sdk_release_1.3/build/boards/athena2/bm1688_
↪wevb_emmc/partition/partition_emmc.xml
Sensor tuning bin: gcore_gc4653
Output path: /data/sdk_release_1.3/install/soc_bm1688_wevb_emmc
```

In the last line, the output path is the directory corresponding to $OUTPUT_DIR. If the directory corresponding to $OUTPUT_DIR does not exist, run the following command to create it:

```
mkdir -p $OUTPUT_DIR
```

After creating a directory, copy the sdcard.tgz basic software package to the directory by running the following command:

```
cp -rf {your_path}/sdcard.tgz $OUTPUT_DIR/    // {your_path}
↪是您获取的sdcard.tgz基础软件包的本地路径
```

3. Run the revert_package command:

```
revert_package
```

After this command is executed, boot.tgz, data.tgz, recovery.tgz, rootfs.tgz, rootfs_w.tgz are generated in the $OUTPUT_DIR/package_edge directory. Two folders sdcard and update are generated in the $OUTPUT_DIR/package_update/ directory.

The sdcard folder is the decompressed sdcard.tgz software package. The update folder contains the five packages initially packaged after revert_package command is executed.

boot.tgz software package is mainly used for kernel.

data.tgz software package is mainly used for data partition.

recovery.tgz software package is used to restore factory settings.

The rootfs.tgz package can be used to create the file system you need. Refer to the Modify Ubuntu 20.04 section to update the rootfs.tgz package. Note that the original rootfs.tgz package used is rootfs.tgz under install/soc_bm1688/.

rootfs_rw.tgz software package file system overlay area, which includes all the settings in the installed app, lib, scripts, services, configurations, is cleared after the update.

4. If you want to modify the partition information, you need to modify the partition32G.xml file in build/scripts/.
5. Replace the modified *.tgz package with the same name *.tgz package in $OUTPUT_DIR/package_edge, and then recompile the refresh package if necessary.

```
build_update sdcard  // recompile sdcard refresh package
build_update tftp    // recompile tftp refresh package
```

After the compile and refresh package command is executed, the sdcard directory and tftp directory are generated in the $OUTPUT_DIR/package_edge directory, and the files in the directories are required for refresh.

7.7 Build the Package from Github Code

To be updated···

7.8 Compile the Kernel Modules at Athena2

You can choose to compile kernel module directly on Athena2, which can save the trouble of building a cross-compilation environment. The steps are as follows:

1. Run the "uname -r" command to obtain the kernel version number and compare it with the file name in /home/linar/bsp-debs and /lib/modules.
2. Because the kernel has defects in making bindeb-pkg in the cross-compilation environment, additional processing needs to be done as follows:
   a. Run the date command to check the current system time. If the difference between the current system time and the actual system time is too large, reset the current system time.

   ```
   sudo date -s "01:01:01 2021-03-01"
   ```

   b. Check whether /home/linar/bsp-dbs/linux-headers-install.sh exists. If so, run it.
   c. If not, you need to do it manually.

```
sudo dpkg -i /home/linaro/bsp-debs/linux-headers-*.deb
sudo mkdir -p /usr/src/linux-headers-$(uname -r)/tools/include/tools
sudo cp /home/linaro/bsp-debs/*.h /usr/src/linux-headers-$(uname-r)/tools/
include/tools
cd /usr/src/linux-headers-$(uname -r)
sudo apt update
sudo apt-get install -y build-essential bc bison flex libssl-dev
sudo make scripts
```

7.9 Modify Partition Table

Athena2 uses the GPT partition table. The configuration file of the partition table is in

```
build/boards/athena2/bm1688_wevb_emmc/partition/partition_emmc.xml
```

The partition table describes the size information for each partition in turn. It is not recommended that you change the order and number of partitions, as well as the readonly and format attributes, to avoid conflicts with writing in some other preinstalled scripts. You can modify the size of each partition. The size of the last partition does not need to be full of the actual eMMC capacity, you can set it to a relatively small value, as long as it is enough to hold the files you are going to preload (that is, the contents of the data.tgz is unwrapped). At the first boot after the refresh, a script will automatically expand this partition to fill all the remaining free space in eMMC.

7.10 Modify u-boot

To be updated···

7.11 Modify the Preset Memory Layout of the Board

To be updated···

7.12 Select the Preset Memory Layout of the Board

To be updated···

7.13 Athena2 kdump-crash Instructions

To be updated···

7.14 Auto Startup Service

If you need to automatically start certain services after startup, refer to this section. The Athena2 series of chips uses systemd to enable auto startup of the core service bmrt_setup, which consists of the following three key files:

```
/etc/systemd/system/bmrt_setup.service (service description file)
/etc/systemd/system/multi-user.target.wants/bmrt_setup.service (soft link)
/usr/sbin/bmrt_setup.sh (execute scripts)
```

The content of bmrt_setup.service is as follows, and you can construct your own service in the /etc/systemd/system directory by referring to its syntax.

```
[Unit]
Description=setup bitmain runtime env.
After=docker.service # Indicates that the service starts after the docker service.

[Service]
User=root
ExecStart=/usr/sbin/bmrt_setup.sh # Specifies the command that the service starts
↪to execute.
Type=oneshot

[Install]
WantedBy=multi-user.target
```

In the bmrt_setup.sh script, we load key drivers such as VPU, JPU, and TPU, so please execute your custom service after this service, or wait for the driver to load in your program logic. You can use "systemd-analyze plot > boot.svg" to generates a vector diagram of the startup details, which is then opened with an image browser or web browser to see the startup order and time taken for all services.

