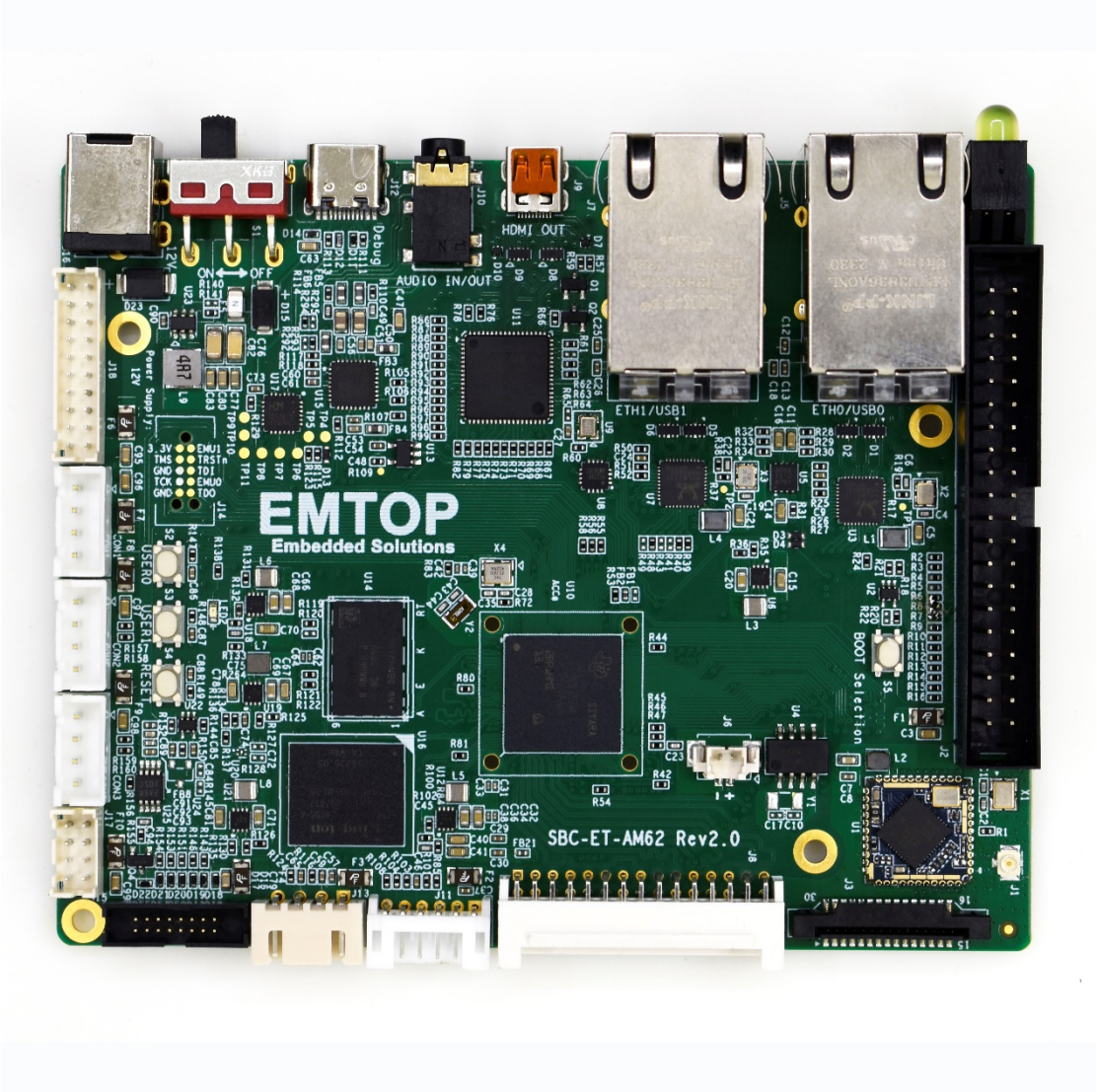
SBC-ET-AM62



User Manual

Version: 1.0

2024-07-18

Revision History：

Version	Date	Description
1.0	2024-07-18	Initial Release

Table of Contents

1. Product Overview

1.1 Introduction

1.2 Resource Download

1.3 Hardware Features

1.4 Mechanical Dimension

2. Linux Operation System

This chapter will give you a general map of the Linux software resources contained in the DVD-ROM provided along with the product, as well as detailed introduction to the process of Linux system development, drivers development, system update, functionality tests and application development examples.

Note：

🕮 It is recommended to learn Ubuntu Linux installation and embedded Linux development technology in advance.

2.1 Software Resources

The DVR-ROM provided along with the board contains demos, application examples, Linux source code and tools, helping you to develop Linux applications and systems easily and quickly.

2.1.1 Location of Resources

You can find software resources such as programs and codes contained in the DVD-ROM according to the information showed in the table below;

Categories	Location
Applications	
Source Code	CD\Source\u-boot-ti-2023.04
	CD\Source\linux-ti-6.1.33
	CD\Source\App
Tools	CD\Tools\
Precompiled Images	CD\Image

2.1.2 BSP

The following table lists types and formats of the files contained in BSP;

Names		Note	Formats
BOOTLOADER	U-BOOT	MMC/SD	Source Code
		FAT	Source Code
		NET	Source Code
KERNEL	LINUX-6.1.33	Support JFFS2/EXT4/FAT/NFS various of file system	Source Code
DEVICE DRIVER	SERIAL	Serials driver	Source Code
	RTC	Hardware RTC driver	Source Code
	NET	10/100M/1Gbps Ethernet driver	Source Code
	CAN	CAN bus driver	Source Code
	SPI	SPI driver	Source Code
	HDMI	SII9022ACNU HDMI driver	Source Code
	I2C	I2C driver	Source Code
	LVDS	LCD driver	Source Code
	TOUCH SCREEN	I2C touch panel driver	Source Code
	MMC/SD	MMC/SD controller driver	Source Code
	USB HOST	USB HOST driver	Source Code
	AUDIO	NAU88C22YG Audio driver(supports recording & playback)	Source Code
	BUTTON	GPIO button driver	Source Code
	LED	LED driver	Source Code
	CAMERA	CSI Camera driver	Source Code
	WIFI	RTL8723DS driver	Source Code
	BLUETOOTH	RTL8723DS driver	Source Code
ROOTFS	YOCTO	Wayland with Qt 5.15	Image

2.2 Structure of Embedded Linux System

SBC-ET-AM62 is shipped with Linux-6.1.33 system in eMMC by default. This system consists of bootloader, kernel and rootfs. The following table shows the structure of embedded Linux system.

Partition	FAT	EXT4
Image	Bootloader, DTB, Kernel	Yocto Rootfs

1) Bootloader is a program generated by u-boot compiling; its file name is **flash.bin**.
2) The kernel used in this document is Linux-6.1.33 and has been customized according to the hardware design.
3) Rootfs stores open-source system Yocto with EXT4 format.

2.3 Building Development Environment

Before developing software, user has to establish a Linux cross development environment on PC. This section will take **Ubuntu 22.04** operating system as an example to describe how to establish a cross development environment.

It is strongly recommended to install necessary software packages for a newly installed Ubuntu through the following commands.

- **sudo apt-get update; sudo apt-get install -y build-essential git xz-utils ncurses-dev autoconf libtool automake texinfo bison flex libc6:i386 libncurses5:i386 libstdc++6:i386**

Note:

- Each instruction has been put a bullets "•" before it to prevent confusion caused by the long instructions that occupy more than one line in the context.
- Please note the SPACES within each instruction; Missing of any SPACE will cause failure when executing instructions.

2.3.1 Installing Cross Compilation Tools

We provide the cross-compiler under **Tools** directory: **arm-gnu-toolchain-11.3.rel1-x86_64-aarch64-none-linux-gnu.tar.xz**, **arm-gnu-toolchain-11.3.rel1-x86_64-arm-none-linux-gnueabihf.tar.xz** and **gcc-linaro-7.5.0-2019.12-x86_64_aarch64-linux-gnu.tar.xz**.

The compiler is mainly used to compile u-boot and kernel.

- **sudo mkdir -p /opt/bin/arm**
- **sudo tar -xvf <YOUR_PATH>/arm-gnu-toolchain-11.3.rel1-x86_64-aarch64-none-linux-gnu.tar.xz -C /opt/bin/arm**
- **sudo tar -xvf <YOUR_PATH>/arm-gnu-toolchain-11.3.rel1-x86_64-arm-none-linux-gnueabihf.tar.xz -C /opt/bin/arm**
- **sudo tar -xvf <YOUR_PATH>/gcc-linaro-7.5.0-2019.12-x86_64_aarch64-linux-gnu.tar.xz -C /opt/bin/arm**

It will extract and install under **/opt/bin/arm** directory, keep the default settings.

2.3.2 Set Cross Compile Environment

Run the following commands to set the source code building environment:

- **export PATH=/opt/bin/arm/gcc-linaro-7.5.0-2019.12-x86_64_aarch64-linux-gnu/bin:$PATH**
- **export ARCH=arm64**
- **export CROSS_COMPILE=arm-linux-**

Note：

🕮 The instructions can be added in the .bashrc file located at the user directory, so that the addition of environment variables will be loaded automatically when the system is booting up;

🕮 If you want to check the path, please use the instruction **printenv PATH**

2.4 Preparing the Source Code

Please refer to chapter <**1.2 Resource Download**> to get the development materials. You can get source code under **Source** directory.

- **tar -xvf u-boot-ti-2023.04-git-xxxxxx.tar.xz**
- **tar -xvf linux-ti-6.1.33-git-xxxxxx.tar.xz**

Then we can get the source code directory **u-boot-ti-2023.04** and **linux-ti-6.1.33**.

2.5 Compilation

1） Compiling Bootloader

Run the following commands to compile bootloader:

- **cd u-boot-ti-2023.04**
- **vi make.sh**

```
export PATH=/opt/bin/arm/arm-gnu-toolchain-11.3.rel1-x86_64-aarch64-none-linux-gnu/bin:$PATH
export PATH=/opt/bin/arm/arm-gnu-toolchain-11.3.rel1-x86_64-arm-none-linux-gnueabihf/bin:$PATH

DESTDIR="/dev/shm"
```

PATH: Replace the compiler path according to your local environment if it is installed under other directory.

DESTDIR: point to a directory to store the target image.

Change **DESTDIR** value to make it point to your target directory according to your local environment.

- **./make.sh setup**

This step will install several necessary tools to compile u-boot source code.

- **./make.sh**

After all the instructions are executed, you can find the booting images under **DESTDIR** directory.

```
DDR1G
├── tiboot3.bin
├── tispl.bin
└── u-boot.img
DDR2G
├── tiboot3.bin
├── tispl.bin
└── u-boot.img
DDR4G
├── tiboot3.bin
├── tispl.bin
└── u-boot.img
```

If you only need to compile bootloader for DDR1G, DDR2G or DDR4G, please run the below command to compile one of them:

- **./make.sh 1g**
- **./make.sh 2g**
- **./make.sh 4g**

2） Compiling Kernel

Execute the following instructions to compile kernel:

- **cd linux-ti-6.1.33**
- **git checkout .**
- **vi make.sh**

```
export PATH=/opt/bin/arm/gcc-linaro-7.5.0-2019.12-x86_64_aarch64-linux-gnu/bin:$PATH
export ARCH=arm64
export CROSS_COMPILE=arm-linux-

DESTDIR="/dev/shm"
```

PATH: Replace the compiler path according to your local environment if it is installed under other directory.

DESTDIR: point to a directory to store the target image.

Please modify **DESTDIR** according to your local environment.

- **make ARCH=arm64 distclean**
- **./make.sh modules**

If it's successfully built, you can find kernel images named **.dtb** files, **Image** and **lib/modules/6.1.33** under **DESTDIR** directory.

Note:

The command **./make.sh**, without parameter, only build dtbs and Image; but **./make.sh modules** will build dtbs, Image and driver modules.

2.6 Yocto SDK

There are 2 ways to compile user application program [Linux C/C++ project]:

① Compile on Board;

② Compile with Yocto SDK under PC Ubuntu system.

If the application code is simple, and doesn't include too much third party libraries, you can try to compile it on board. For example, let us try to compile the UART program [**Source/App/com.tar.xz**]:

- root@arm:~# **vi Makefile**

```
# CROSS_COMPILE ?= arm-linux-

all:
	$(CROSS_COMPILE)gcc -s -o com com_example.c

clean:
	rm -fr com *~
```

- root@arm:~# **make**

```
gcc -s -o com com_example.c
```

Now, we already get the target ELF file **com**. Try to run and check:

- root@arm:~# **./com -h**

```
Usage: ./com option [ dev... ]
        -h  --help      Display this usage information.
        -d  --device    The device ttyS[0-3] or ttyEXT[0-3]
        -s  --string    Write the device data
        -f  --flow      Flow control switch
        -b  --speed     Set speed bit/s
        -m  --mode      Set mode, rs232 or rs485
        -n  --count     Repeat times, default non-stop
```

If the application code is complex, and we can't compile it on board successfully, please try to compile it under Ubuntu with Yocto SDK.

2.6.1 Build Yocto SDK

Note：

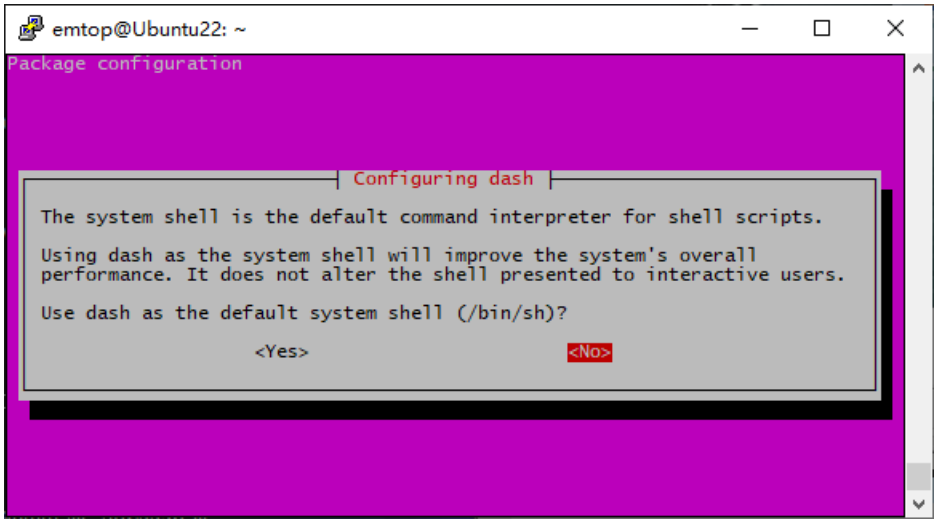
🕮 This is an **OPTIONAL** operation step. Only useful when you want to build your own SDK.

Please refer to TI document: **https://software-dl.ti.com/processor-sdk-linux-rt/esd/AM62X/09_00_00_03/exports/docs/linux/Overview_Building_the_SDK.html**

[HOST] Ubuntu 22.04

- **sudo apt-get update**
- **sudo apt-get -f -y install git build-essential diffstat texinfo gawk chrpath socat doxygen dos2unix python3 bison flex libssl-dev u-boot-tools mono-devel mono-complete curl python3-distutils repo pseudo python3-sphinx g++-multilib libc6-dev-i386 jq git-lfs pigz zstd liblz4-tool cpio file zstd lz4**
- **sudo dpkg-reconfigure dash**

Be sure to select “No” when you are asked to use dash as the default system shell:



- **git clone https://git.ti.com/git/arago-project/oe-layersetup.git tisdk**
- **cd tisdk**
- **./oe-layertool-setup.sh -f configs/processor-sdk/processor-sdk-09.00.00-config.txt**
- **cd build**
- **. conf/setenv**
- **export TOOLCHAIN_PATH_ARMV7=/opt/bin/arm/arm-gnu-toolchain-11.3.rel1-x86_64-arm-none-linux-gnueabihf**
- **export TOOLCHAIN_PATH_ARMV8=/opt/bin/arm/arm-gnu-toolchain-11.3.rel1-x86_64-aarch64-none-linux-gnu**
- **vi ../sources/meta-processor-sdk/recipes-graphics/powervr-graphics/powervr-graphics_5.10.bb**

```
DESCRIPTION = "Imagination PowerVR SDK binaries/examples"
LICENSE = "MIT"

SRC_URI = " \
    gitsm://github.com/powervr-graphics/Native_SDK.git;protocol=https;branch=${BRANCH} \
    file://0001-PATCH-use-library-so-names-for-linking.patch \
"
```

- **vi /etc/hosts**

```
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters

185.199.108.133  raw.githubusercontent.com
```

Build system image:

- **MACHINE=am62xx-evm ARAGO_RT_ENABLE=1 bitbake tisdk-default-image**

It will generate **arago-tmp-default-glibc/deploy/images/am62xx-evm/tisdk-default-image-am62xx-evm.wic.xz**.

Note：

The wic.xz is a **soft link**, please check to find the real file it points to.

2.6.2 Install Yocto SDK

Install SDK **Tools/arago-2023.04.tar.xz** with the below command:

- **sudo tar -xvf <YOUR_PATH>/arago-2023.04.tar.xz -C /opt**

2.6.3 Compile User Application Program Project

Let us try to compile a Qt example **easing**, which is from **qt-everywhere-src-5.15.0/qtbase/examples/widgets/animation/easing**. Get it from **Source/easing.tar.xz**.

- **source /opt/arago-2023.04/environment-setup**
- **cd easing**
- **qmake**

```
Info: creating stash file easing/.qmake.stash
```

- **make**

```
.....
aarch64-oe-linux-g++ --sysroot=/opt/arago-2023.04/sysroots/aarch64-oe-linux --sysroot=/opt/arago-2023.04/sysroots/aarch64-oe-linux --sysroot=/opt/arago-2023.04/sysroots/aarch64-oe-linux -Wl,-O1 -o easing main.o window.o qrc_easing.o moc_window.o /opt/arago-2023.04/sysroots/aarch64-oe-linux/usr/lib/libQt5Widgets.so /opt/arago-2023.04/sysroots/aarch64-oe-linux/usr/lib/libQt5Gui.so /opt/arago-2023.04/sysroots/aarch64-oe-linux/usr/lib/libQt5Core.so -lGLESv2 -lpthread
```

- **file easing**

```
easing: ELF 64-bit LSB shared object, ARM aarch64, version 1 (GNU/Linux), ...
```

Copy the target file **easing** to ARM board, run it and you can see the animation widget.

2.7 Linux System Customization

In order to satisfy different requirements of customers, designers commonly need to make some custom modification based on the default configuration of Linux kernel. This chapter will introduce the process of system customization with some examples.

2.7.1 Replace Kernel LOGO

- Prepare a picture suitable for your display screen size, named **my_logo.png** for example.
- Install some necessary programs under Ubuntu.
  - **sudo apt-get install netpbm gimp**
- Run command under Ubuntu desktop terminal:
  - **pngtopnm my_logo.png > linuxlogo.pnm**
  - **pnmquant 224 linuxlogo.pnm > linuxlogo224.pnm**
  - **pnmtoplainpnm linuxlogo224.pnm > logo_linux_clut224.ppm**

- Update Linux source code.
  - **cp -f logo_linux_clut224.ppm <YOUR_PATH>/linux-ti-6.1.33/drivers/video/logo/logo_linux_clut224.ppm**
- Re-build the kernel.
  - **make ARCH=arm64 distclean**
  - **./make.sh**

Update the target file **Image** to the board, reboot and check the boot logo on the display screen.

2.7.2 Setting Configuration Menu

A default configuration file is provided under kernel source codes:

linux-ti-6.1.33/kernel/configs/emtop-sbc-et-am62.config

Please execute the following commands to enter the configuration menu:

- **cd linux-ti-6.1.33**
- **export PATH=/opt/bin/arm/gcc-linaro-7.5.0-2019.12-x86_64_aarch64-linux-gnu/bin:$PATH**
- **export ARCH=arm64**
- **export CROSS_COMPILE=aarch64-linux-gnu-**
- **make defconfig ti_arm64_prune.config ti_rt.config emtop-sbc-et-am62.config**
- **make menuconfig**

Note：

🕮 If an error occurs when command 'make ARCH=arm64 menuconfig' is executed, you might need to install 'ncurse' in the Ubuntu system, ‘ncurses’ is a character graphic library required to generate configuration menu. Please enter the following instruction to install the library:
sudo apt-get install libncurses5-dev

2.7.3 Menu Options

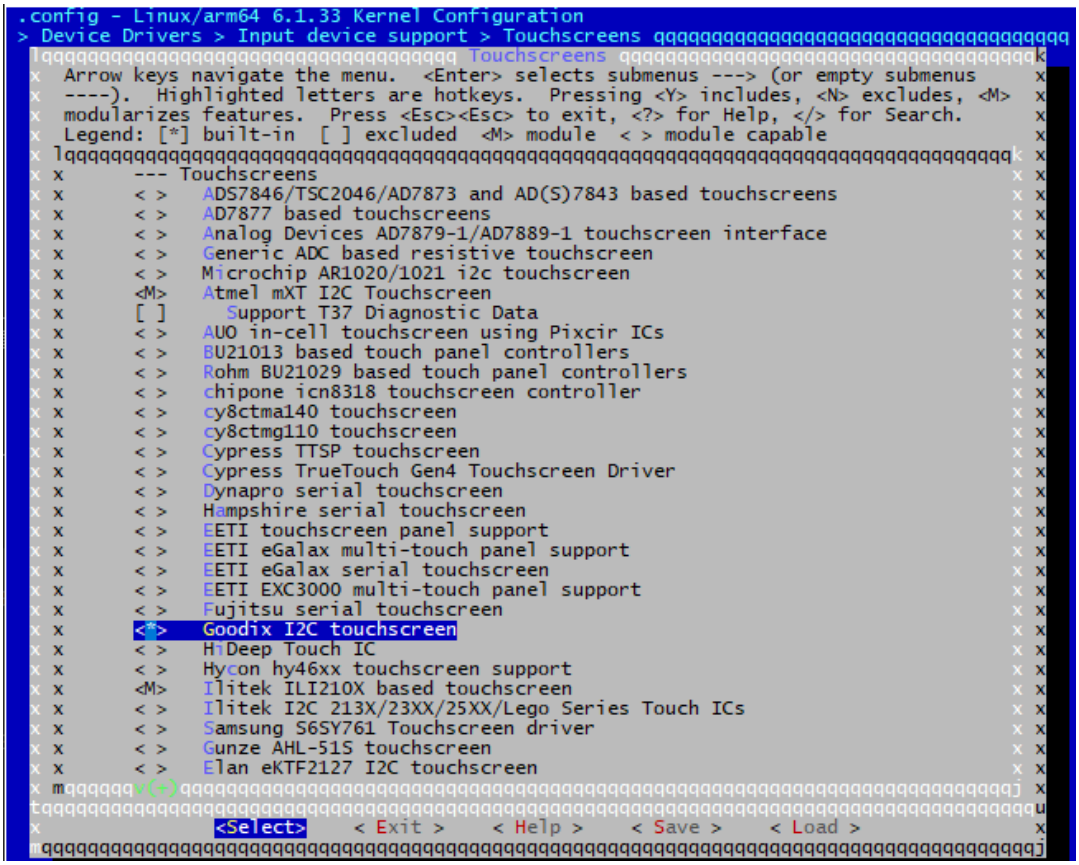
Configure options according to customization requirements after entering configu ration menu, for example, access **Device Drivers** > **Input device support** > **Touc hscreens** > **Goodix I2C touchscreen** as shown below:

-> Device Drivers

-> Input device support

-> Touchscreens

-> Goodix I2C touchscreen



Set Goodix I2C touchscreen to **<*>**, exit and save changes.

2.7.4 Compile Kernel

Please execute the following instructions to recompile kernel:

- **./make.sh**

The script will **NOT** overwrite the configuration modified by menuconfig. It mea ns that the current setting you modified is effective in your target kernel image.

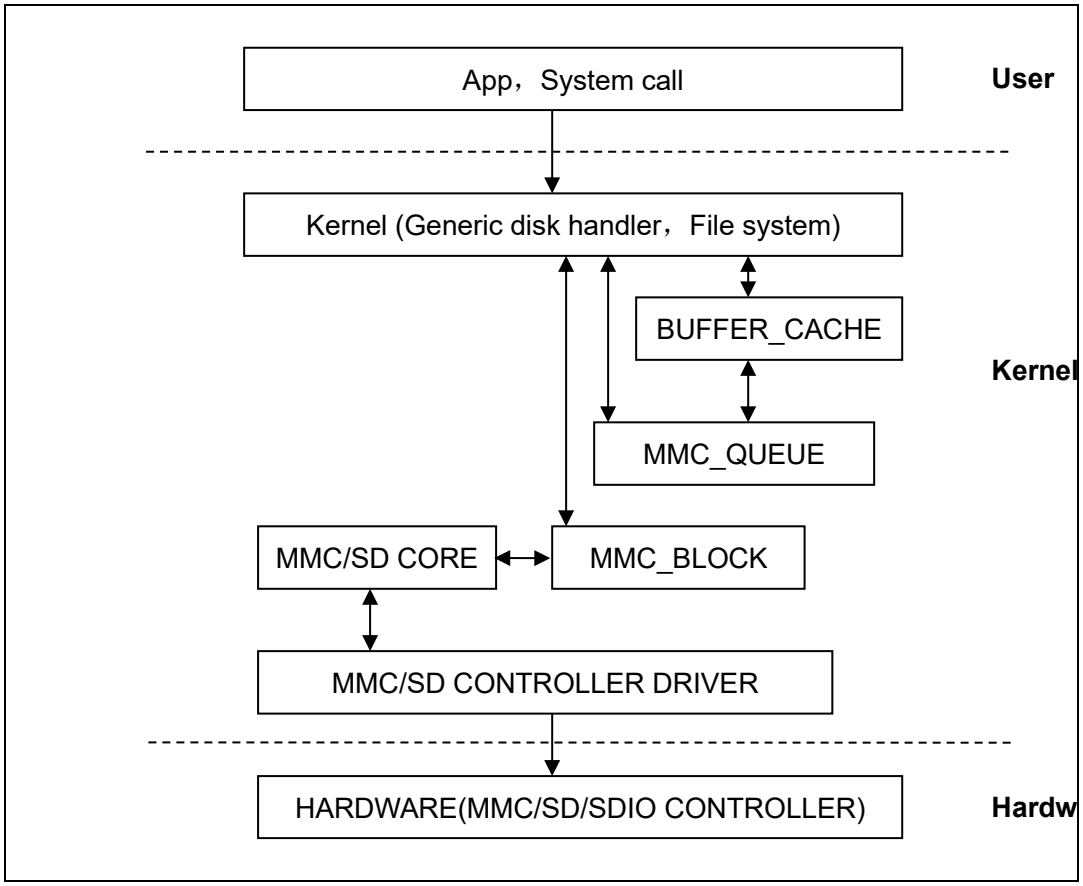
If you want to restore to the default configuration, please delete the file **.config** and run **./make.sh**.

2.8 Introduction to Drivers

The table below shows the access path to find all the drivers:

Category	Name	Description	Location
Bootloader	U-BOOT	MMC/SD	drivers/mmc/am654_sdhci.c
		FAT	fs/
		NET	drivers/net/ti/am65-cpsw-nuss.c
Kernel	Linux-6.1.33	Support JFFS2/EXT4/FAT/NFS etc.	fs/
Devices	SERIAL	Serial driver	drivers/tty/serial/8250/8250_omap.c
	RTC	Hardware RTC driver	drivers/rtc/rtc-rx8010.c
	NET	10/100M/1000M Ethernet driver	drivers/net/ethernet/ti/am65-cpsw-nuss.c
	CAN	CAN bus driver	drivers/net/can/m_can/m_can_platform.c
	SPI	SPI driver	drivers/spi/spi-omap2-mcspi.c
	DSS	DSS driver	drivers/gpu/drm/tidss/tidss_drv.c
	HDMI	HDMI driver	drivers/gpu/drm/bridge/sii902x.c
	TOUCH SCREEN	I2C touch panel driver	drivers/input/touchscreen/goodix.c
	MMC/SD	MMC/SD controller driver	drivers/mmc/host/sdhci_am654.c
	USB	USB controller driver	drivers/usb/dwc3/dwc3-am62.c
	AUDIO	NAU88C22 Audio driver(supports recording & playback)	sound/soc/codecs/nau8822.c
	BUTTON	GPIO button driver	drivers/input/keyboard/gpio_keys.c
	LED	LED driver	drivers/leds/leds-gpio.c
	WIFI/BT	RTL8723DS driver	3rdparty/rtl8723ds
	CAMERA	CSI Camera driver	drivers/media/i2c/ov5640.c

2.8.1 SD/MMC



SD/MMC drivers in Linux are mainly consisted of SD/MMC core, mmc_block, mmc_queue and SD/MMC driver：

1） SD/MMC core realizes the codes unrelated to structure in the SD/MMC card operation;

2） mmc_block realizes driver structure when SD/MMC card is used as a block device;

3） mmc_queue realizes management of request queue;

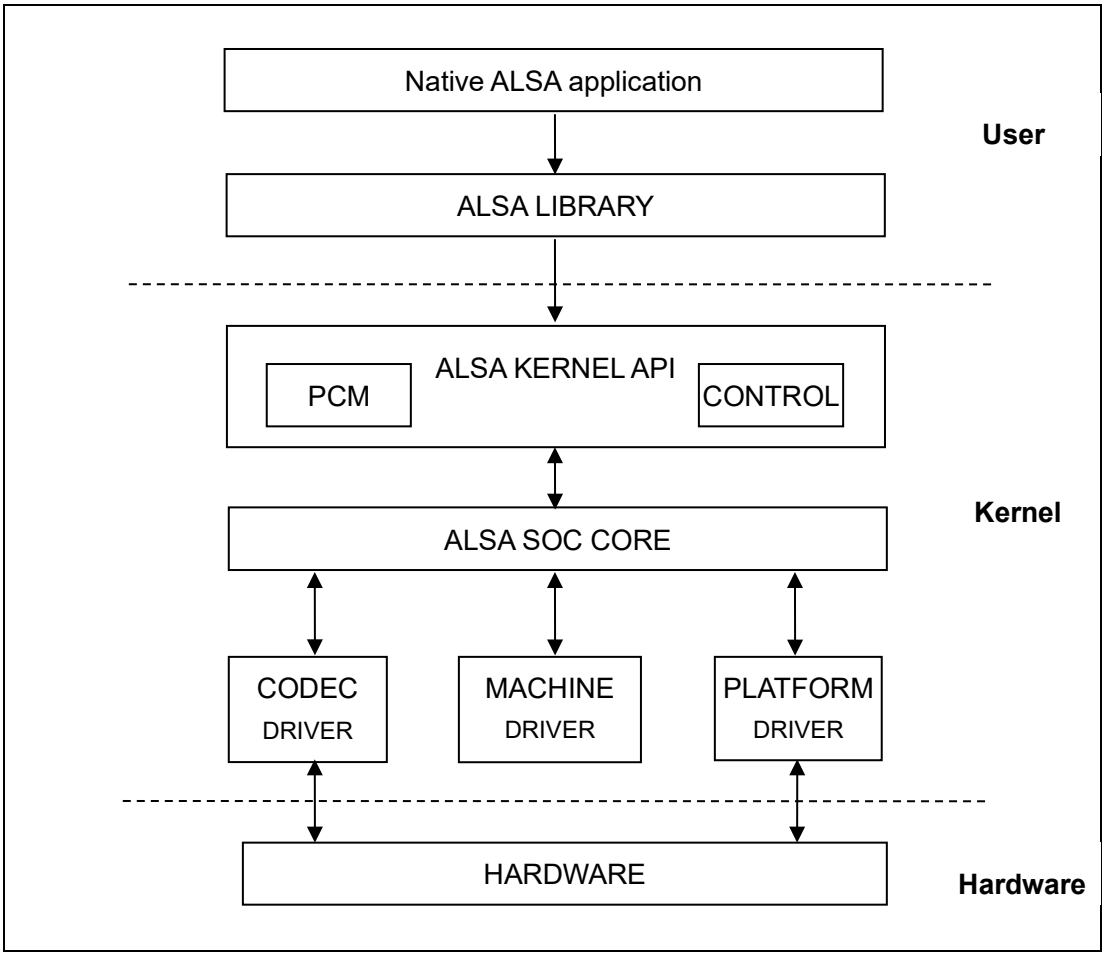
4） SD/MMC driver realizes specific controller driver.

Drivers and relevant documents:

linux-ti-6.1.33/drivers/mmc/

linux-ti-6.1.33/drivers/mmc/host/sdhci_am654.c

2.8.2 Audio In/Out



ASoC embedded audio system basically consists of three components:

1) Codec driver: The codec driver is platform independent and contains audio controls, audio interface capabilities, codec dapm definition and codec IO functions.

2) Platform driver: It contains the audio dma engine and audio interface drivers (e.g. I2S, AC97, PCM) of that platform.

3) Machine driver: The machine driver handles any machine specific controls and audio events i.e. turning on an amp at start of playback.

Drivers and relevant documents:

linux-ti-6.1.33/sound/soc/ti

linux-ti-6.1.33/sound/soc/codecs/nau8822.c

2.9 Driver development

2.9.1 GPIO_LEDs Driver

1）Device Definition

linux-ti-6.1.33/arch/arm64/boot/dts/ti/emtop-sbc-et-am62.dts

Configure MCU_GPIO0_22 as system running status indicator, blinking as heartbeat.

```
leds {
    compatible = "gpio-leds";
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_gpio_led>;

    sys {
        label = "sys";
        gpios = <&mcu_gpio0 22 GPIO_ACTIVE_HIGH>;
        linux,default-trigger = "heartbeat";
    };
```

2）GPIO pinmux Configuration

linux-ti-6.1.33/arch/arm64/boot/dts/ti/emtop-sbc-et-am62.dts

Configure PMIC_LPM_EN0 as MCU_GPIO0_22 function:

```
&mcu_pmx0 {
    ...
    pinctrl_gpio_led: gpioledgrp {
        pinctrl-single,pins = <
            AM62X_MCU_IOPAD(0x0080, PIN_OUTPUT, 7) /* (B7) PMIC_LPM_EN
0.MCU_GPIO0_22 */
        >;
    };
};
```

3）Driver Design

linux-ti-6.1.33/drivers/leds/leds-gpio.c

a) Call platform_driver_register to register gpio_leds driver

```
static struct platform_driver gpio_led_driver = {
	.probe		= gpio_led_probe,
	.shutdown	= gpio_led_shutdown,
	.driver		= {
		.name	= "leds-gpio",
		.of_match_table = of_gpio_leds_match,
	 },
};

module_platform_driver(gpio_led_driver);

MODULE_AUTHOR("Raphael Assenat <raph@8d.com>, Trent Piepho <tpiepho@freescale.com>");
MODULE_DESCRIPTION("GPIO LED driver");
MODULE_LICENSE("GPL");
MODULE_ALIAS("platform:leds-gpio");
```

b) Apply for gpio and call led_classdev_register to led_classdev drivr.

```
static int gpio_led_probe(struct platform_device *pdev)
{
…
	priv->num_leds = pdata->num_leds;
		for (i = 0; i < priv->num_leds; i++) {
			const struct gpio_led *template = &pdata->leds[i];
			struct gpio_led_data *led_dat = &priv->leds[i];

			if (template->gpiod)
				led_dat->gpiod = template->gpiod;
			else
				led_dat->gpiod =
					gpio_led_get_gpiod(&pdev->dev,
							i, template);
			if (IS_ERR(led_dat->gpiod)) {
				dev_info(&pdev->dev, "Skipping unavailable LED gpio %d (%s)\n",
					template->gpio, template->name);
				continue;
			}
```

```
                ret = create_gpio_led(template, led_dat,
                                      &pdev->dev, NULL,
                                      pdata->gpio_blink_set);
                if (ret < 0)
                    return ret;
            }
        } else {
            priv = gpio_leds_create(pdev);
            if (IS_ERR(priv))
                return PTR_ERR(priv);
        }

        platform_set_drvdata(pdev, priv);

        return 0;
}

static int create_gpio_led(const struct gpio_led *template,
        struct gpio_led_data *led_dat, struct device *parent,
        struct fwnode_handle *fwnode, gpio_blink_set_t blink_set)
{
        struct led_init_data init_data = {};
        int ret, state;

        led_dat->cdev.default_trigger = template->default_trigger;
        led_dat->can_sleep = gpiod_cansleep(led_dat->gpiod);
        if (!led_dat->can_sleep)
            led_dat->cdev.brightness_set = gpio_led_set;
        else
            led_dat->cdev.brightness_set_blocking = gpio_led_set_blocking;
        led_dat->blinking = 0;
        if (blink_set) {
            led_dat->platform_gpio_blink_set = blink_set;
            led_dat->cdev.blink_set = gpio_blink_set;
        }
        if (template->default_state == LEDS_GPIO_DEFSTATE_KEEP) {
            state = gpiod_get_value_cansleep(led_dat->gpiod);
            if (state < 0)
                return state;
        } else {
```

```
            state = (template->default_state == LEDS_GPIO_DEFSTATE_ON);
        }
        led_dat->cdev.brightness = state ? LED_FULL : LED_OFF;
        if (!template->retain_state_suspended)
            led_dat->cdev.flags |= LED_CORE_SUSPENDRESUME;
        if (template->panic_indicator)
            led_dat->cdev.flags |= LED_PANIC_INDICATOR;
        if (template->retain_state_shutdown)
            led_dat->cdev.flags |= LED_RETAIN_AT_SHUTDOWN;

        ret = gpiod_direction_output(led_dat->gpiod, state);
        if (ret < 0)
            return ret;

        if (template->name) {
            led_dat->cdev.name = template->name;
            ret = devm_led_classdev_register(parent, &led_dat->cdev);
        } else {
            init_data.fwnode = fwnode;
            ret = devm_led_classdev_register_ext(parent, &led_dat->cdev,
                                                 &init_data);
        }
        return ret;
}
```

c) Users may access the file named brightness under **/sys/class/leds/sys/brightness**, and call gpio_led_set to configure LED status

```
static void gpio_led_set(struct led_classdev *led_cdev,
        enum led_brightness value)
{
...
                    gpiod_set_value(led_dat->gpiod, level);
}
```

2.9.2 PINMUX Configuration Guide

AM625 has two types of GPIO: one controlled by A53, the other controlled by MCU.

A53 GPIOs are named as: GPIO0_12, GPIO1_31

MCU GPIOs are named as: MCU_GPIO0_22

Configure the A53 GPIO pin attribute:

```
AM62X_IOPAD(0x01b0, PIN_OUTPUT, 7)  /* MCASP0_ACLKR.GPIO1_14 */
```

Configure the MCU GPIO pin attribute:

```
AM62X_MCU_IOPAD(0x0080, PIN_OUTPUT, 7) /* PMIC_LPM_EN0.MCU_GPIO0_22*/
```

Function syntax:

AM62X_IOPAD(pa, val, muxmode) or AM62X_MCU_IOPAD(pa, val, muxmode)
pa: Physical Address **val**: value to write **muxmode**: MUXMODE[3:0]

They are defined in **linux-ti-6.1.33/arch/arm64/boot/dts/ti/k3-pinctrl.h**:

```
#define AM62X_IOPAD(pa, val, muxmode)          (((pa) & 0x1fff)) ((val) | (muxmode))
#define AM62X_MCU_IOPAD(pa, val, muxmode)    (((pa) & 0x1fff)) ((val) | (muxmod
e))
```

Now, let's explain how to calculate the parameter 'pa' of PMIC_LPM_EN0.

Open document <AM62x Sitara™ Processors>, find out the physical address of pin PMIC_LPM_EN0 is 0x04084080:

B7	C7	PMIC_LPM_EN0 PADCONFIG: MCU_PADCONFIG32 0x04084080	PMIC_LPM_EN0 MCU_GPIO0_22

- **vi linux-ti-6.1.33/arch/arm64/boot/dts/ti/k3-am62-mcu.dtsi**

```
mcu_pmx0: pinctrl@4084000 {
    compatible = "pinctrl-single";
    reg = <0x00 0x04084000 0x00 0x88>;
    #pinctrl-cells = <1>;
    pinctrl-single,register-width = <32>;
    pinctrl-single,function-mask = <0xffffffff>;
};
```

It says the base physical address of mcu_pmx0 is **0x04084000**, and then we only need to pass the offset address of pin PMIC_LPM_EN0 through macro AM62X_MCU_IOPAD:

$$offset = 0x04084080 - 0x04084000 = 0x80$$

About the parameter 'val', you can choose from below items:

- **vi linux-ti-6.1.33/arch/arm64/boot/dts/ti/k3-pinctrl.h**

```
/* Only these macros are expected be used directly in device tree files */
#define PIN_OUTPUT          (INPUT_DISABLE | PULL_DISABLE)
#define PIN_OUTPUT_PULLUP   (INPUT_DISABLE | PULL_UP)
#define PIN_OUTPUT_PULLDOWN (INPUT_DISABLE | PULL_DOWN)
#define PIN_INPUT           (INPUT_EN | PULL_DISABLE)
#define PIN_INPUT_PULLUP    (INPUT_EN | PULL_UP)
#define PIN_INPUT_PULLDOWN  (INPUT_EN | PULL_DOWN)
```

About the parameter 'muxmode', find 'Pin Attributes' table of document <AM62x Sitara™ Processors>:

ALW BALL NUMBER [1]	AMC BALL NUMBER [1]	BALL NAME [2] PADCONFIG Register [15] PADCONFIG Address [16]	SIGNAL NAME [3]	MUX MODE [4]	TYPE [5]
J22	J21	OSPI0_D7 PADCONFIG: PADCONFIG10 0x000F4028	OSPI0_D7	0	IO
			SPI1_D1	1	IO
			MCASP1_AFSX	2	IO
			UART6_CTSn	3	I
			GPIO0_10	7	IO
B7	C7	PMIC_LPM_EN0 PADCONFIG: MCU_PADCONFIG32 0x04084080	PMIC_LPM_EN0	0	O
			MCU_GPIO0_22	7	IO

You can see the MUXMODE of MCU_GPIO0_22 is **7**.

If your target pin is controlled by A53, please append it under **main_pmx0** node in

dts file; otherwise put it under **mcu_pmx0** node:

```
&mcu_pmx0 {
    usr_led_pins_default: usr-led-pins-default {
        pinctrl-single,pins = <
            AM62X_MCU_IOPAD(0x0080, PIN_OUTPUT, 7) /* (B7) PMIC_LPM_EN0.
MCU_GPIO0_22 */
        >;
    };
    ......
}
```

2.10 System Update

SBC-ET-AM62 can boot up from TF card and eMMC. It's decided by the BOOT button:

Press Down [Not Release]: Boot from TF card

Otherwise: Boot from eMMC

2.10.1 Update TF Card System Image

1） Make A Bootable TF Card

- Get the system image from **Image** directory, named as **SBC-ET-AM62-SD-REVXX.img.xz**, unxz it and get the raw image **SBC-ET-AM62-SD-REVXX.img**.
- If you work under Windows system, please run **Tools/win32diskimager** to write the **SBC-ET-AM62-SD-REVXX.img** into TF Card. If you work under Linux system, please use **dd** command to write it into TF Card.

After win32diskimage writing completes, the TF card space is not 100% available for

the system.

Install the TF card under Ubuntu system, and run commands below:

- **fdisk -l /dev/sdx**

```
Disk /dev/sdb: 14.84 GiB, 15931539456 bytes, 31116288 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x076c4a2a

Device          Boot  Start      End  Sectors  Size Id Type
/dev/sdb1       *      2048   526335   524288  256M  c W95 FAT32 (LBA)
/dev/sdb2       526336 12582911 12056576  5.7G 83 Linux
```

In order to use the whole space of TF card, run the tool program **Tools/mmc-resize-full.sh**:

- **./mmc-resize-full.sh /dev/sdx**

```
Warning: Dangerouse operation! Please confirm /dev/sdb is the target device[Y/N]:Y
Info: Capacity: 15931539456

Welcome to fdisk (util-linux 2.37.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help): Partition number (1,2, default 2):
Partition 2 has been deleted.

Command (m for help): Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): Partition number (2-4, default 2): First sector (2048-31116287, defa
ult 2048): Last sector, +/-sectors or +/-size{K,M,G,T,P} (196608-31116287, default 311
16287):
Created a new partition 2 of type 'Linux' and of size 14.8 GiB.
Partition #2 contains a ext4 signature.
```

```
Command (m for help): Partition number (1,2, default 2): Hex code (type L to list all codes):
Changed type of partition 'Linux' to 'Linux'.

Command (m for help): The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

Info: fsck /dev/sdb2
e2fsck 1.45.5 (07-Jan-2020)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/sdb2: 55280/190464 files (0.1% non-contiguous), 742369/761856 blocks
Info: resize2fs /dev/sdb2
resize2fs 1.45.5 (07-Jan-2020)
Resizing the filesystem on /dev/sdb2 to 3864960 (4k) blocks.
The filesystem on /dev/sdb2 is now 3864960 (4k) blocks long.
```

Now, check its capacity:

- **fdisk -l /dev/sdx**

```
Disk /dev/sdb: 14.84 GiB, 15931539456 bytes, 31116288 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x076c4a2a

Device     Boot  Start      End  Sectors  Size Id Type
/dev/sdb1  *      2048   526335   524288  256M  c W95 FAT32 (LBA)
/dev/sdb2       526336 31116287 30589952 14.6G 83 Linux
```

2） Update U-Boot

If you've made some changes to the u-boot source code, and want to update it into TF card, please copy the target image into the root directory of the TF card FAT partition:

```
├── tiboot3.bin
├── tispl.bin
└── u-boot.img
```

3） Update Kernel

If you have modified the kernel source code, please update the dtb and Image under Partition 1 [FAT32] of the TF Card. That partition can be recognized by Windows or Linux.

4） Update Rootfs

Because EXT4 isn't accessible Under Windows, please mount the Partition 2 of TF Card under Ubuntu, change the target file and umount the card.

Note:

🕮 If eMMC is already written with system image, please erase eMMC and then reboot the board, because the board will first try to boot from eMMC by default.

🕮 Enter u-boot command and erase eMMC:

u-boot=> **mmc dev 0 && mmc erase 0 20000**

2.10.2 Update eMMC with TFCard

Option 1: Write Complete Image into eMMC

- Make a bootable TF card and boot up the system;
- Choose the target image [under directory **Image/**] and copy it into the USB disk [Formatted as NTFS or exFAT]. If it is **.xz** file, please unxz it to generate **.img** file.
- Install the USB disk on the ARM board, for example, the USB disk is recognized as **sda**;
  - root@arm:~# **mount /dev/sda /mnt**
- Run command to start writing eMMC:

- root@arm:~# **umount /dev/mmcblk0***

```
umount: /dev/mmcblk0: not mounted.
umount: /dev/mmcblk0boot0: not mounted.
umount: /dev/mmcblk0boot1: not mounted.
umount: /dev/mmcblk0p1: not mounted.
umount: /dev/mmcblk0p2: not mounted.
umount: /dev/mmcblk0rpmb: not mounted.
```

- root@arm:~# **dd if=/mnt/SBC-ET-AM62-SD-REVXX.img of=/dev/mmcblk0 status=progress bs=4M**

- Run command to write bootloader [MUST]:

  - root@arm:~# **bootloader-update.sh 1g**

bootloader-update.sh 1g For 1GB DDR Device

bootloader-update.sh 2g For 2GB DDR Device

bootloader-update.sh 4g For 4GB DDR Device

  - root@arm:~# **mmc-resize-full.sh /dev/mmcblk0**

```
Warning: Dangerouse operation! Please confirm /dev/mmcblk0 is the target device[Y/N]:Y
......
Syncing disks.

[  167.199381]  mmcblk0: p1 p2
Info: fsck /dev/mmcblk0p2
e2fsck 1.47.0 (5-Feb-2023)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/mmcblk0p2: 91790/928512 files (0.1% non-contiguous), 1020852/3803136 blocks
Info: resize2fs /dev/mmcblk0p2
resize2fs 1.47.0 (5-Feb-2023)
The filesystem is already 3803136 (4k) blocks long.  Nothing to do!
```

After it's done, power off the board, remove the TF card, then reboot the board, it

should boot from eMMC and enter into Linux prompt.

Option 2: Synchronize eMMC with TF card

- Make a bootable TF card and boot up the system;
- Run command to start writing eMMC:
  - root@arm:~# **system-update.sh**

```
running system update...
==============eMMC UPDATE=================
Warning: disk /dev/mmcblk0 will be formatted !
3000+0 records in
3000+0 records out
1536000 bytes (1.5 MB, 1.5 MiB) copied, 0.189324 s, 8.1 MB/s

Welcome to fdisk (util-linux 2.37.4).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.
......
Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

[    82.174125] EXT4-fs (mmcblk0p2): mounted filesystem with ordered data mode. O
pts: (null). Quota mode: none.
sending incremental file list
./
bin/
bin/arping
bin/ash -> /bin/busybox.nosuid
bin/base64 -> /usr/bin/base64.coreutils
bin/bash -> /bin/bash.bash
bin/bash.bash
bin/busybox -> busybox.nosuid
......
sent 13,977,149 bytes  received 141 bytes  2,541,325.45 bytes/sec
```

```
total size is 31,423,849  speedup is 2.25
rsync error: some files/attrs were not transferred (see previous errors) (code 23) at main.c(1336) [sender=3.2.7]
[  825.639924]  mmcblk0: p1 p2
5120+0 records in
5120+0 records out
5242880 bytes (5.2 MB, 5.0 MiB) copied, 0.203386 s, 25.8 MB/s
UPDATE  : COMPLETED
Catch a signal
[  826.153152] EXT4-fs (mmcblk0p2): mounted filesystem with ordered data mode. Opts: (null). Quota mode: none.
```

- Power down the board and remove the TF card.

2.11 Test and Demonstration

This section will run some tests on the peripheral devices.

POWER: **12V DC**

Debug Port [Type-C Slot]: **UART0**, **115200 1N8**.

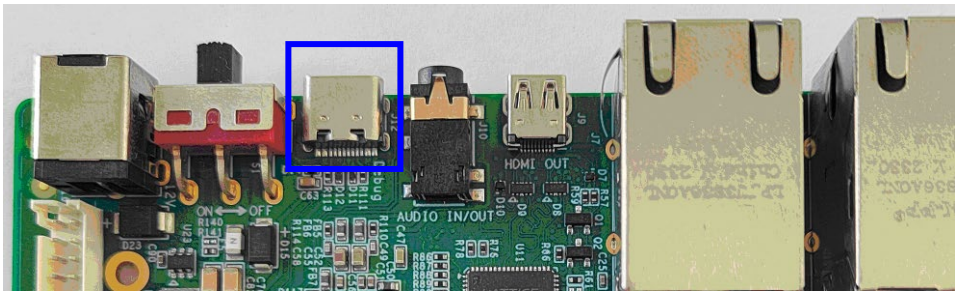


Figure 2-1 Debug Port

2.11.1 SSH LOGIN

The SSH server is already enabled by default. Please get the local IP of the wired-network or wireless-network on ARM board and then login from PC side with SSH client such as PuTTY, **root** account with empty password.

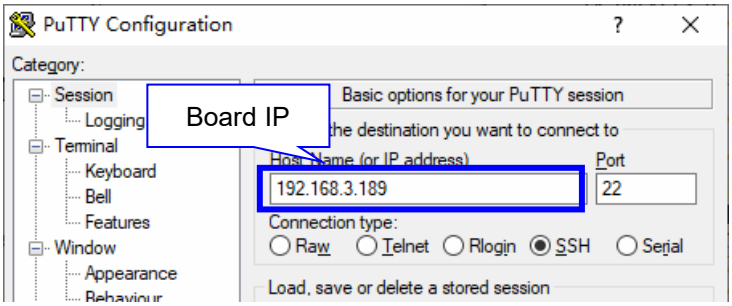


Figure 2-2 PuTTY Login

Note:

The SSH server is dropbear, not openssh-server.

2.11.2 RTC

There is a RTC chip RX8010SJ on the board, but the integrated RTC is still enabled by default. So there are 2 RTC devices accessible under system.

- root@arm:~# **cat /sys/class/rtc/rtc0/name**

```
rtc-rx8010 0-0032
```

- root@arm:~# **cat /sys/class/rtc/rtc1/name**

```
rtc-ti-k3 2b1f0000.rtc
```

That means the **rtc0** is RX8010SJ, and **rtc1** is the integrated RTC. The command **hwclock** accesses /dev/rtc0 as default. If you want to access /dev/rtc1, please append parameter: **-f /dev/rtc1**.

Let's set the current time to 2023-02-05 10:12,

- root@arm:~# **date -s "2023-02-05 10:12"; hwclock -w**

Reboot the board, and check the hardware RTC time with below command:

- root@arm:~# **hwclock**

```
2023-02-05 10:13:03.435901+00:00
```

2.11.3 TIMEZONE SETTING

Set Beijing Time for example:

- root@arm:~# **echo "Asia/Shanghai" > /etc/timezone**
- root@arm:~# **ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime**
- root@arm:~# **sync**

Note:

🕮 If TI Yocto image doesn't contain zoneinfo, copy **/usr/share/zoneinfo** under Ubuntu system to the board, and retry the above commands.

2.11.4 USB HOST

There are 2 USB host channels [USB type-A slot] extended on the base board.

Install an USB disk on these slots, check message below:

```
[ 272.082860] usb-storage 2-1.1:1.0: USB Mass Storage device detected
[ 272.098248] scsi host0: usb-storage 2-1.1:1.0
[ 273.104255] scsi 0:0:0:0: Direct-Access SanDisk Flash Memory 0.1 PQ: 0
ANSI: 2
[ 273.130158] sd 0:0:0:0: [sda] 2001888 512-byte logical blocks: (1.02 GB/977 MiB)
[ 273.143825] sd 0:0:0:0: [sda] Write Protect is off
[ 273.147410] sd 0:0:0:0: [sda] Mode Sense: 03 00 00 00
[ 273.148611] sd 0:0:0:0: [sda] No Caching mode page found
[ 273.155755] sd 0:0:0:0: [sda] Assuming drive cache: write through
[ 273.176207] sda: sda1
[ 273.199625] sd 0:0:0:0: [sda] Attached SCSI removable disk
[ 273.783449] FAT-fs (sda1): Volume was not properly unmounted. Some data may be
corrupt. Please run fsck.
```

- root@arm:~# **mount**

```
.....
/dev/sda1 on /run/media/sda1 type vfat (rw,relatime,gid=6,fmask=0007,dmask=0007,all
ow_utime=0020,codepage=437,iocharset=iso8859-1,shortname=mixed,errors=remount-ro)
```

The USB disk is automatically mounted under **/run/media/sda1** by udev.

2.11.5 NETWORK

There are two 1Gbps network chips RTL8211F on board.

HARDWARE	LINUX SYSTEM	INTERFACE	PHY	PHY ADDR
J5	eth0	RGMII1	RTL8211F	4
J7	eth1	RGMII2	RTL8211F	6

- root@arm:~# **ifconfig eth0**

```
eth0      Link encap:Ethernet  HWaddr 3a:f7:82:bc:fa:0a
          inet addr:192.168.1.81  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::38f7:82ff:febc:fa0a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11 errors:0 dropped:4 overruns:0 frame:0
          TX packets:42 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1555 (1.5 KiB)  TX bytes:7192 (7.0 KiB)
```

DHCP feature is enabled as default; the board can request a valid IP address from DHCP server in local network. Also, you can try the below command to force to request IP address:

- root@arm:~# **udhcpc -i eth0**

```
udhcpc: started, v1.35.0
udhcpc: broadcasting discover
udhcpc: broadcasting select for 192.168.1.81, server 192.168.1.1
udhcpc: lease of 192.168.1.81 obtained from 192.168.1.1, lease time 86400
/etc/udhcpc.d/50default: Adding DNS 192.168.1.1
```

Because there are several network interfaces: eth1, ppp, wlan, we need to configure

the default gateway:

- root@arm:~# **route del default; route add default eth0**

- root@arm:~# **ping -I eth0 www.baidu.com**

```
PING www.a.shifen.com (14.215.177.38) from 192.168.1.81 eth0: 56(84) bytes of data.
64 bytes from www.baidu.com (183.232.231.174): icmp_seq=1 ttl=56 time=12.1 ms
64 bytes from www.baidu.com (183.232.231.174): icmp_seq=2 ttl=56 time=12.2 ms
64 bytes from www.baidu.com (183.232.231.174): icmp_seq=3 ttl=56 time=12.1 ms
64 bytes from www.baidu.com (183.232.231.174): icmp_seq=4 ttl=56 time=12.5 ms
^C
--- www.a.shifen.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 7.058/7.447/7.771/0.319 ms
```

Do the same operations to eth1.

2.11.6 HDMI

Connect HDMI displayer with mini-HDMI cable, power up the ARM board. It can display Wayland desktop.

2.11.7 HDMI AUDIO

MODEL	DESCRIPTION	DTB
HDMI	1920 * 1080	emtop-sbc-et-am62-hdmi.dtb

Edit **uEnv.txt**: let **name_fdt** point to the DTB in the above table.

- root@arm:~# **aplay -l**

```
**** List of PLAYBACK Hardware Devices ****
card 0: AM62xSil9022HDM [AM62x-Sil9022-HDMI], device 0: davinci-mcasp.0-i2s-hif]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

- root@arm:~# **aplay /test/app/music.wav**

```
Playing WAVE '/test/app/music.wav' : Signed 16 bit Little Endian, Rate 48000 Hz, Stereo
```

Now you can hear music sound from HDMI player.

2.11.8 LVDS

Devices already tested:

MODEL	DESCRIPTION	DTB
VISLCD-101HYS145ACT02	1920 * 1200	emtop-sbc-et-am62.dtb
BOE-BA104S01-100	800 * 600	emtop-sbc-et-am62-lvds-boe.dtb

Edit **uEnv.txt**: let **name_fdt** point to the DTB in the above table.

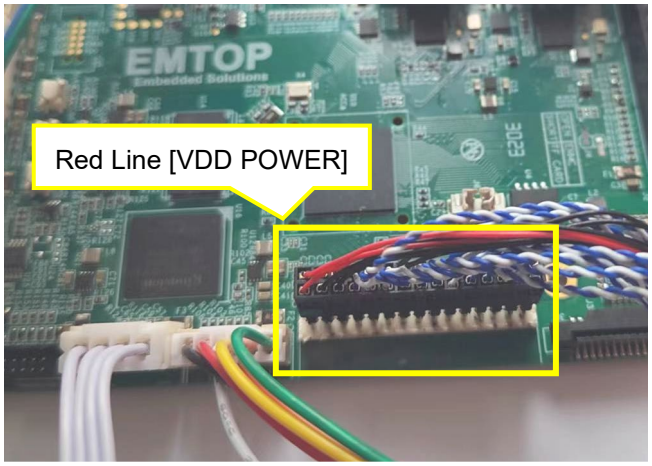


Figure 2-3 VISLCD-101HYS145ACT02

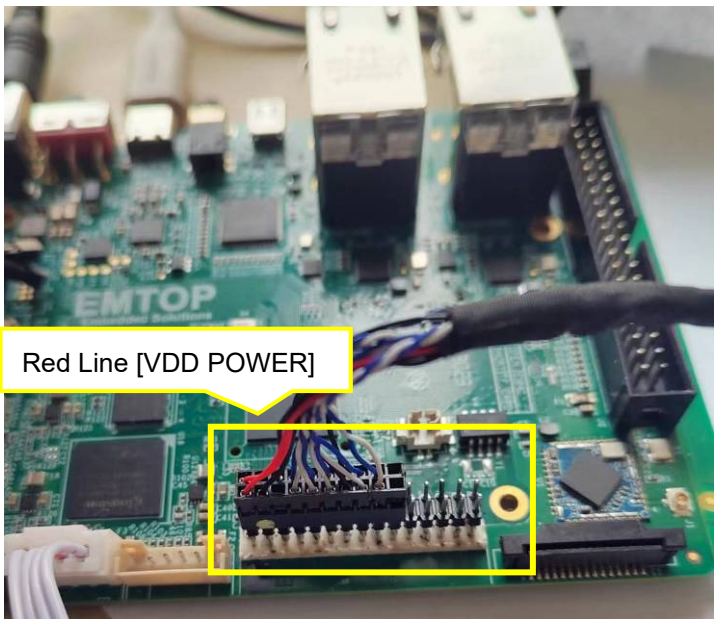


Figure 2-4 BOE-BA104S01-100

2.11.9 LVDS BACKLIGHT

- root@arm:~# **echo 5 > /sys/class/backlight/backlight/brightness**

Note:

🕮 The value of backlight level should be: **0** ~ **8**

2.11.10 TOUCH PANEL

MODEL	TYPE	I2C BUS	REMARK
GT9271	I2C CTP	I2C3	VISLCD-101HYS145ACT02 integrated

- root@arm:~# **evtest**

```
No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:          Goodix Capacitive TouchScreen
/dev/input/event1:          keys
Select the device event number [0-2]: 0
Input driver version is 1.0.1
```

```
Input device ID: bus 0x18 vendor 0x416 product 0x38f version 0x1060
Input device name: "Goodix Capacitive TouchScreen"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 59 (KEY_F1)
    Event code 60 (KEY_F2)
    Event code 61 (KEY_F3)
    Event code 62 (KEY_F4)
    Event code 63 (KEY_F5)
    Event code 64 (KEY_F6)
    Event code 125 (KEY_LEFTMETA)
    Event code 330 (BTN_TOUCH)
  Event type 3 (EV_ABS)
    Event code 0 (ABS_X)
      Value          0
      Min            0
      Max         1919
    Event code 1 (ABS_Y)
      Value          0
      Min            0
      Max         1199
    Event code 47 (ABS_MT_SLOT)
      Value          0
      Min            0
      Max            4
    Event code 48 (ABS_MT_TOUCH_MAJOR)
      Value          0
      Min            0
      Max          255
    Event code 50 (ABS_MT_WIDTH_MAJOR)
      Value          0
      Min            0
      Max          255
    Event code 53 (ABS_MT_POSITION_X)
      Value          0
      Min            0
      Max         1919
    Event code 54 (ABS_MT_POSITION_Y)
      Value          0
```

```
        Min          0
        Max       1199
    Event code 57 (ABS_MT_TRACKING_ID)
        Value        0
        Min          0
        Max      65535
Properties:
  Property type 1 (INPUT_PROP_DIRECT)
Testing ... (interrupt to exit)
[Touch the panel ...]
Event: time 1647024852.722824, type 3 (EV_ABS), code 57 (ABS_MT_TRACKING_ID), value 0
Event: time 1647024852.722824, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value 878
Event: time 1647024852.722824, type 3 (EV_ABS), code 54 (ABS_MT_POSITION_Y), value 255
Event: time 1647024852.722824, type 3 (EV_ABS), code 48 (ABS_MT_TOUCH_MAJOR), value 10
Event: time 1647024852.722824, type 3 (EV_ABS), code 50 (ABS_MT_WIDTH_MAJOR), value 10
Event: time 1647024852.722824, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 1
Event: time 1647024852.722824, type 3 (EV_ABS), code 0 (ABS_X), value 878
Event: time 1647024852.722824, type 3 (EV_ABS), code 1 (ABS_Y), value 255
Event: time 1647024852.722824, ---------- SYN_REPORT ------------
Event: time 1647024852.756503, type 3 (EV_ABS), code 57 (ABS_MT_TRACKING_ID), value -1
Event: time 1647024852.756503, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 0
Event: time 1647024852.756503, ---------- SYN_REPORT ------------
```

2.11.11 NAU88C22YG AUDIO

Choose an earphone with MIC inside.

Figure 2-5 Earphone with MIC

- root@arm:~# **aplay -l**

```
**** List of PLAYBACK Hardware Devices ****
card 0: AM62xNAU8822 [AM62x-NAU8822], device 0: davinci-mcasp.0-nau8822-hifi nau8822-hifi-0 [davinci-mcasp.0-nau8822-hifi nau8822-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

Playback:

- root@arm:~# **aplay /test/app/music.wav**

```
Playing WAVE 'music.wav' : Signed 16 bit Little Endian, Rate 48000 Hz, Stereo
```

Note:

Only support **48kHz** sample rate audio file. If you play 44.1 kHz, it sounds a bit faster than normal speed.

Record:

- root@arm:~# **amixer set 'ALC Enable' Both**
- root@arm:~# **arecord -d 20 -t wav -f cd --period-size=64 record.wav**

Note:

🕮 The parameter '--period-size=64' is very crucial. If it's not set, fatal error may be reported: "**davinci-mcasp 2b00000.audio-controller: Receive buffer overflow**"

Wait several seconds, press Ctrl+C to terminate arecord program. Now, let's play it to check:

- root@arm:~# **aplay record.wav**

2.11.12 UART

DEVICE NODE	HARDWARE	USAGE
/dev/ttyS2	UART0	DEBUG
/dev/ttyS7	UART5	5-WIRED
/dev/ttyS8	UART6	BLUETOOTH
/dev/ttyS9	MCU_UART0	3-WIRED
/dev/ttyS10	WKUP_UART0	3-WIRED

Test UART5, it has hardware flow control function. Connect TXD and RXD, RTS and CTS, run command:

- root@arm:~# **/test/app/com -f -d /dev/ttyS7**

```
SEND: 1234567890
RECV: 1234567890
SEND: 1234567890
RECV: 1234567890
```

The default baud rate is **115200**. If you want to assign another specific baud rate:

- root@arm:~# **/test/app/com -f -b 9600 -d /dev/ttyS7**

Please refer to the source code **com.tar.xz** for all supported baud rates.

Note:

🕮 Parameter '**-f**': enable hardware flow control.

Test MCU_UART0 or WKUP_UART0, connect TXD and RXD, run command:

- root@arm:~# **/test/app/com -d /dev/ttyS9**

```
SEND: 1234567890
RECV: 1234567890
SEND: 1234567890
RECV: 1234567890
```

They don't have hardware flow control signal, so the hardware flow control function should be disabled.

2.11.13 CAN BUS

INTERFACE	HARDWARE	USAGE
can0	MCU_MCAN0	
can1	MCU_MCAN1	
can2	MCAN0	

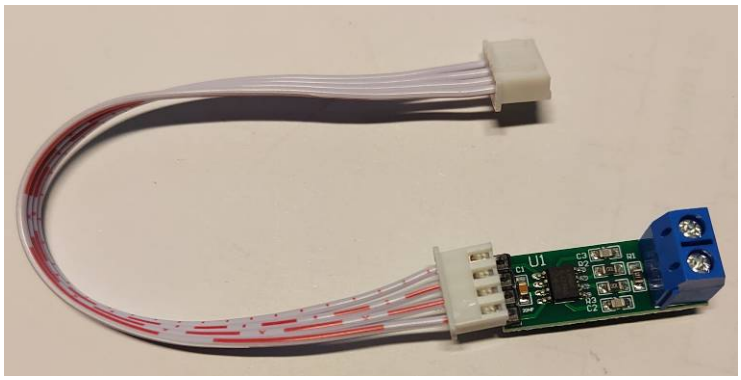


Figure 2-6 CAN Transceiver TJA1050

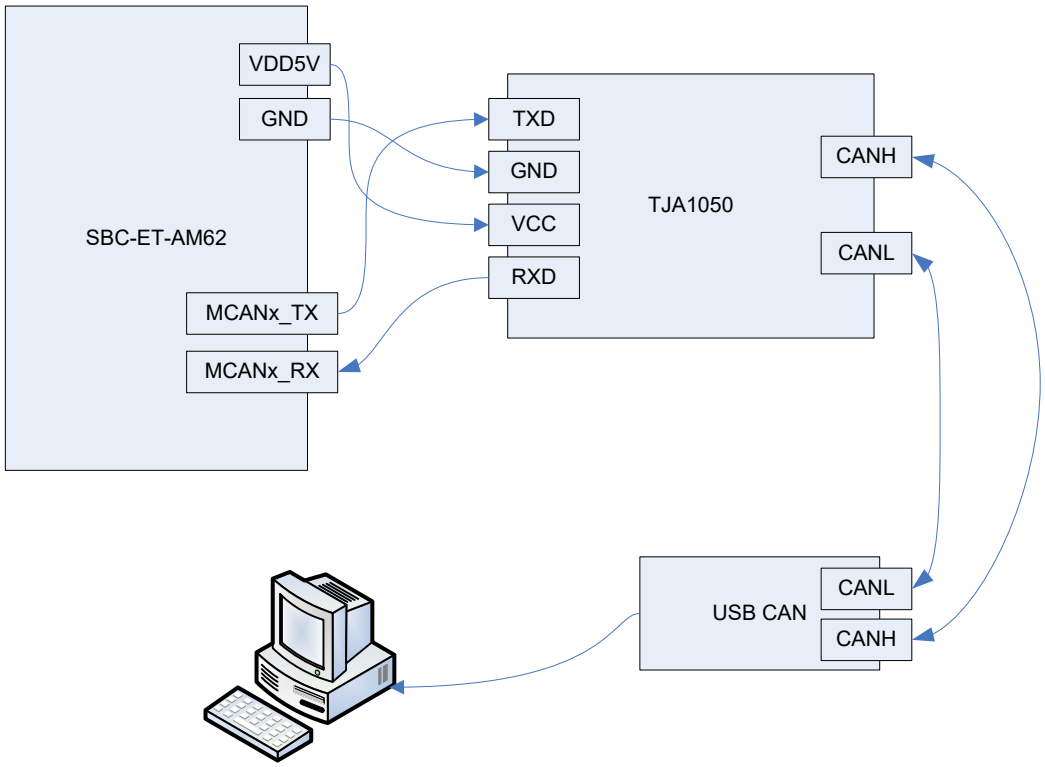


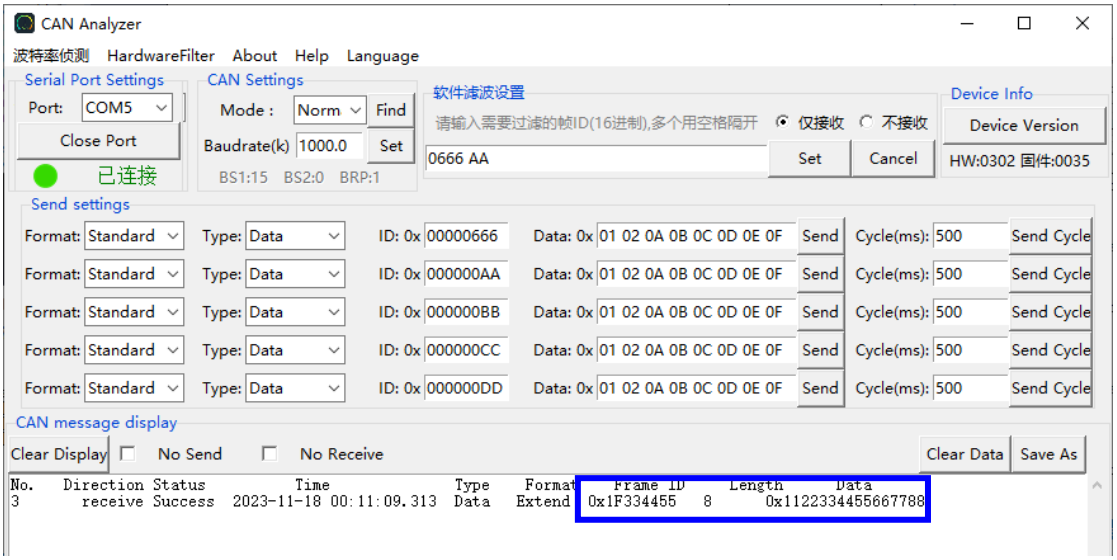
Figure 2-7 CAN BUS Connection

Let's test can0 as an example, set the bitrate 1Mbps:

- root@arm:~# **ifconfig can0 down; ip link set can0 type can bitrate 1000000; if config can0 up**

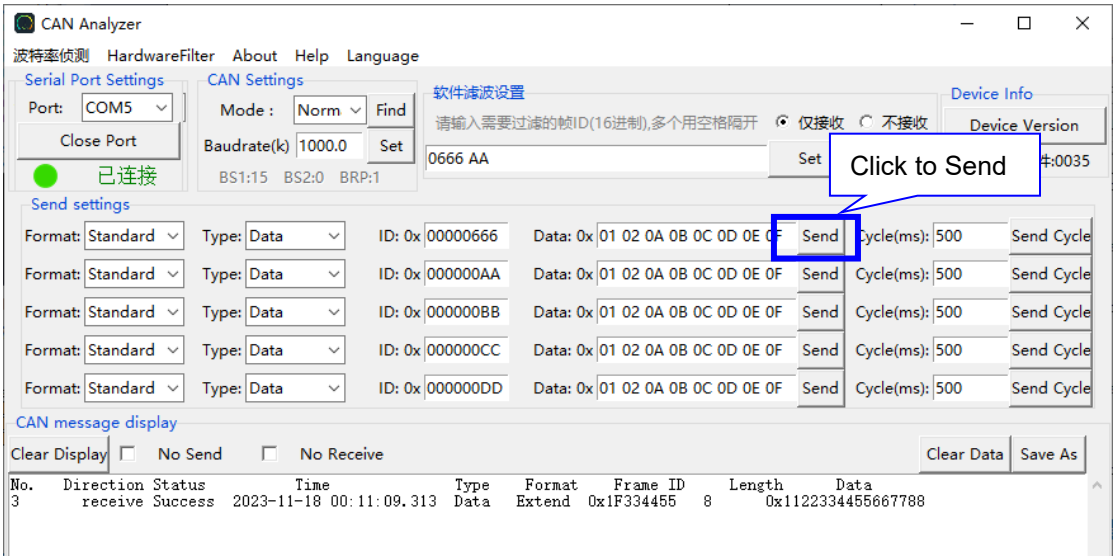
Send test:

- root@arm:~# **cansend can0 1F334455#1122334455667788**



Receive test:

- root@arm:~# **candump can0**




```
can0  666  [8]  01 02 0A 0B 0C 0D 0E 0F
```

2.11.14 BUTTON

There are 2 user buttons on board: USR0 and USR1:

- root@arm:~# **evtest**

```
No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:          Goodix Capacitive TouchScreen
/dev/input/event1:          keys
Select the device event number [0-1]: 1
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "keys"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 102 (KEY_HOME)
    Event code 105 (KEY_LEFT)
Properties:
Testing ... (interrupt to exit)
[Press button S2 and S3]
Event: time 1700142827.619520, type 1 (EV_KEY), code 105 (KEY_LEFT), value 1
Event: time 1700142827.619520, ---------- SYN_REPORT ------------
Event: time 1700142827.757994, type 1 (EV_KEY), code 105 (KEY_LEFT), value 0
Event: time 1700142827.757994, ---------- SYN_REPORT ------------
Event: time 1700142828.283934, type 1 (EV_KEY), code 102 (KEY_HOME), value 1
Event: time 1700142828.283934, ---------- SYN_REPORT ------------
Event: time 1700142828.448805, type 1 (EV_KEY), code 102 (KEY_HOME), value 0
Event: time 1700142828.448805, ---------- SYN_REPORT ------------
```

2.11.15 LED

There is only one LED for user control:

- root@arm:~# **echo none > /sys/class/leds/sys/trigger**
- root@arm:~# **while test 1; do echo 1 > /sys/class/leds/sys/brightness;sleep 1;echo 0 > /sys/class/leds/sys/brightness;sleep 1;done**

Your can see the corresponding LED blinking with 2Hz frequency.

2.11.16 PWM

There are 2 PWM channels in CON1, ECAP_IN_PWMO and EXP_REFCLK1.

EXP_REFCLK1 works under ECAP0_IN_APWM_OUT mode.

- root@arm:~# **echo 0 > /sys/class/pwm/pwmchip2/export**

- root@arm:~# **echo 0 > /sys/class/pwm/pwmchip2/pwm0/enable 2>/dev/null**
- root@arm:~# **echo 100000 > /sys/class/pwm/pwmchip2/pwm0/period**
- root@arm:~# **echo 50000 > /sys/class/pwm/pwmchip2/pwm0/duty_cycle**
- root@arm:~# **echo normal > /sys/class/pwm/pwmchip2/pwm0/polarity**
- root@arm:~# **echo 1 > /sys/class/pwm/pwmchip2/pwm0/enable**

period = 100000(ns), frequency = 1 / period = 10KHz

duty_cycle = 50000(ns), duty ratio = duty_cycle / period = 50%

polarity = normal, it can be set to **normal** or **inversed**

ECAP_IN_PWMO works under ECAP1_IN_APWM_OUT mode

- root@arm:~# **echo 0 > /sys/class/pwm/pwmchip3/export**

- root@arm:~# **echo 0 > /sys/class/pwm/pwmchip3/pwm0/enable**
- root@arm:~# **echo 200000 > /sys/class/pwm/pwmchip3/pwm0/period**
- root@arm:~# **echo 50000 > /sys/class/pwm/pwmchip3/pwm0/duty_cycle**
- root@arm:~# **echo normal > /sys/class/pwm/pwmchip3/pwm0/polarity**
- root@arm:~# **echo 1 > /sys/class/pwm/pwmchip3/pwm0/enable**

period = 200000(ns), frequency = 1 / period = 5KHz

duty_cycle = 50000(ns), duty ratio = duty_cycle / period = 25%

polarity = normal, it can be set to **normal** or **inversed**

2.11.17 EEPROM

There is an EEPROM 24LC32A on board.

- root@arm:~# **hexdump -Cv /sys/bus/i2c/devices/0-0050/eeprom**

```
00000000  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
00000010  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
00000020  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
00000030  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
00000040  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
00000050  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
00000060  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
00000070  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
00000080  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
00000090  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
```

Usually it's designed to store board information and not writable for user.

2.11.18 TFCard

When booting from eMMC, the TFCard will be recognized as a removable disk device.

- root@arm:~# **fdisk -l**

```
Disk /dev/mmcblk1: 14.98 GiB, 16088301568 bytes, 31422464 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x9bb82186

Device             Boot  Start      End  Sectors  Size Id Type
/dev/mmcblk1p1 *         2048   526335   524288  256M  c W95 FAT32 (LBA)
/dev/mmcblk1p2         526336 31422463 30896128 14.7G 83 Linux
```

2.11.19 eMMC

eMMC is mainly used for keeping system image, needless to test it manually.

- root@arm:~# **fdisk -l**

```
Disk /dev/mmcblk0: 7.13 GiB, 7650410496 bytes, 14942208 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/mmcblk0boot0: 4 MiB, 4194304 bytes, 8192 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/mmcblk0boot1: 4 MiB, 4194304 bytes, 8192 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

2.11.20 RTL8723DS WIFI

- root@arm:~# **ifconfig wlan0 up**
- root@arm:~# **iw dev wlan0 scan**

```
BSS da:32:5d:02:29:9f(on wlan0)
        TSF: 1709603810 usec (0d, 00:28:29)
        freq: 2462
        beacon interval: 100 TUs
        capability: ESS Privacy ShortPreamble ShortSlotTime RadioMeasure (0x1431)
        signal: -67.00 dBm
        last seen: 4 ms ago
        Information elements from Probe Response frame:
        SSID: EMTOP
        Supported rates: 1.0* 2.0* 5.5* 11.0*
        DS Parameter set: channel 11
        Country: CN     Environment: bogus
                Channels [1 - 13] @ 20 dBm
```

```
Supported operating classes:
        * current operating class: 81
        * operating class: 83
        * operating class: 84
        * operating class: 125
        * operating class: 128
        * operating class: 129
ERP: Use_Protection
Extended supported rates: 6.0 9.0 12.0 18.0 24.0 36.0 48.0 54.0
RSN:    * Version: 1
        * Group cipher: CCMP
        * Pairwise ciphers: CCMP
        * Authentication suites: PSK
        * Capabilities: 16-PTKSA-RC 1-GTKSA-RC (0x000c)
HT capabilities:
        Capabilities: 0x9ad
                RX LDPC
                HT20
                SM Power Save disabled
                RX HT20 SGI
                TX STBC
                RX STBC 1-stream
                Max AMSDU length: 7935 bytes
                No DSSS/CCK HT40
        Maximum RX AMPDU length 65535 bytes (exponent: 0x003)
        Minimum RX AMPDU time spacing: 2 usec (0x04)
        HT RX MCS rate indexes supported: 0-15
        HT TX MCS rate indexes are undefined
HT operation:
        * primary channel: 11
        * secondary channel offset: no secondary
        * STA channel width: 20 MHz
        * RIFS: 0
        * HT protection: nonmember
        * non-GF present: 0
        * OBSS non-GF present: 1
        * dual beacon: 0
        * dual CTS protection: 0
        * STBC beacon: 0
        * L-SIG TXOP Prot: 0
```

```
                * PCO active: 0
                * PCO phase: 0
```

If you only care about the available WiFi AP, please try:

- root@arm:~# **iw dev wlan0 scan | grep SSID**

- root@arm:~# **wpa_passphrase EMTOP 12345678 >> /etc/wpa_supplicant.conf**

File: /etc/wpa_supplicant.conf

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1

network={
        key_mgmt=NONE
}
network={
        ssid="EMTOP"
        #psk="12345678"

psk=c238e09ef54285daf31c8f6833efab9fb8ff55632f7b9a7d94c117711de27822
}
```

- root@arm:~# **wpa_supplicant -B -iwlan0 -c/etc/wpa_supplicant.conf**

```
Successfully initialized wpa_supplicant
rfkill: Cannot open RFKILL control device
[ 2594.006812] wlan: wlan0 START SCAN
[ 2598.357520] wlan: SCAN COMPLETED: scanned AP count=0
[ 2603.369086] wlan: wlan0 START SCAN
[ 2607.717417] wlan: SCAN COMPLETED: scanned AP count=1
[ 2607.735508] wlan: Connected to bssid 94:XX:XX:XX:0a:bc successfully
[ 2608.381534] wlan0:
[ 2608.381550] wlan: Send EAPOL pkt to 94:XX:XX:XX:0a:bc
[ 2608.398971] wlan0:
[ 2608.398985] wlan: Send EAPOL pkt to 94:XX:XX:XX:0a:bc
[ 2608.400137] woal_cfg80211_set_rekey_data return: gtk_rekey_offload is DISABLE
```

- root@arm:~# **udhcpc -i wlan0**

```
udhcpc: started, v1.35.0
```

```
udhcpc: broadcasting discover
udhcpc: broadcasting discover
udhcpc: broadcasting select for 192.168.1.100, server 192.168.1.1
udhcpc: lease of 192.168.1.100 obtained from 192.168.1.1, lease time 7200
RTNETLINK answers: File exists
/etc/udhcpc.d/50default: Adding DNS 192.168.1.1
```

2.11.21 RTL8723DS BLUETOOTH

- root@arm:~# **rtk_hciattach -n -s 115200 /dev/ttyS8 rtk_h5 &**

```
Realtek Bluetooth :Realtek Bluetooth init uart with init speed:115200, type:HCI UART H5
Realtek Bluetooth :Realtek hciattach version 3.1

Realtek Bluetooth :Use epoll
Realtek Bluetooth :[SYNC] Get SYNC Resp Pkt
Realtek Bluetooth :[CONFIG] Get SYNC pkt
Realtek Bluetooth :[CONFIG] Get CONFG pkt
Realtek Bluetooth :[CONFIG] Get CONFG resp pkt
Realtek Bluetooth :dic is 1, cfg field 0x14
Realtek Bluetooth :H5 init finished

Realtek Bluetooth :Realtek H5 IC
Realtek Bluetooth :Receive cmd complete event of command: fc61
Realtek Bluetooth :LMP Subversion 0xbeef
Realtek Bluetooth :Receive cmd complete event of command: 1001
Realtek Bluetooth :HCI Version 0x08
Realtek Bluetooth :HCI Revision 0x000d
Realtek Bluetooth :LMP Subversion 0x8723
Realtek Bluetooth :Receive cmd complete event of command: fc6d
Realtek Bluetooth :Read ROM version 02
Realtek Bluetooth :LMP Subversion 0x8723
Realtek Bluetooth :EVersion 2
Realtek Bluetooth :IC: RTL8723DS
Realtek Bluetooth :Firmware/config: rtl8723d_fw, rtl8723d_config
Realtek Bluetooth :Couldnt open extra config /opt/rtk_btconfig.txt, No such file or directory
Realtek Bluetooth :Couldnt access customer BT MAC file /opt/bdaddr
Realtek Bluetooth :Origin cfg len 47
.....
```

```
Realtek Bluetooth :Realtek Bluetooth post process
[ 1959.885935] Bluetooth: HCI UART protocol LL registered
Realtek Bluetooth :Device setup complete
[ 1959.885987] Bluetooth: HCI UART protocol Three-wire (H5) registered
[ 1959.889250] Bluetooth: HCI UART protocol Broadcom registered
[ 1959.889321] Bluetooth: HCI UART protocol QCA registered
[ 1959.889662] Bluetooth: HCI UART protocol Marvell registered
[ 1960.209802] Bluetooth: : Non-link packet received in non-active state
[ 1960.746695] Bluetooth: MGMT ver 1.22
[ 1960.764557] NET: Registered PF_ALG protocol family
```

- root@arm:~# **bluetoothctl**

```
Agent registered
[bluetooth]# power on
Changing power on succeeded
[bluetooth]# scan on
Discovery started
[CHG] Controller D0:C5:D3:F9:60:06 Discovering: yes
[NEW] Device 78:C5:28:67:88:03 78-C5-28-67-88-03
[NEW] Device 7B:A2:1E:1D:15:60 7B-A2-1E-1D-15-60
.. ..
[bluetooth]# scan off
```

Please search **bluetoothctl** usage on web for more information.

2.11.22 MIPI-CSI CAMERA

Devices already tested:

MODEL	CORE	RESOLUTION
ALINX AN5641	OV5640	QSXGA (2592x1944), 1080p, 1280x960, VGA (640x480)

Figure 2-8 AN5641

Camera Test:

- root@arm:~# **systemctl stop weston**
- root@arm:~# **gst-launch-1.0 v4l2src device="/dev/video0" ! video/x-raw, width=640, height=480, format=YUY2 ! queue ! kmssink driver-name=tidss**

```
Setting pipeline to PAUSED ...
Pipeline is live and does not need PREROLL ...
Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
Redistribute latency...
0:00:01.9 / 99:99:99.
```

Now we can see the real-time image stream captured by the camera is displaying on screen.

2.11.23 PWM-LED [EMTOP Sensor Board]

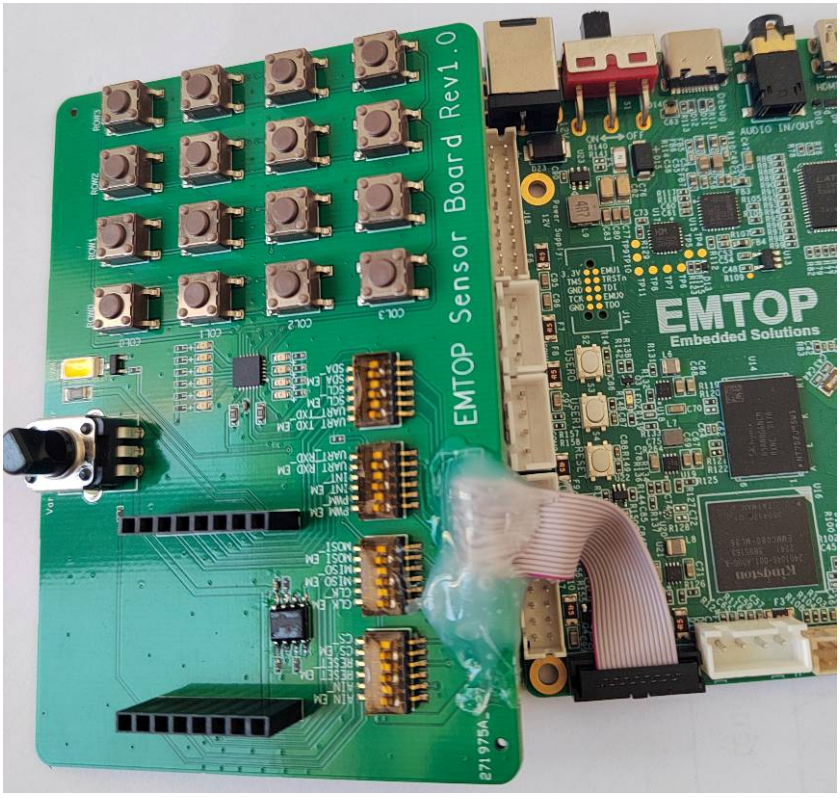


Figure 2-9 EMTOP Sensor Board

There is a LED on the Sensor Board, its brightness can be controlled by PWM signal.

- root@arm:~# **for i in `seq 0 20`; do /test/app/mcu_timer1 200 $i; sleep 0.2;done**

2.11.24 ROTARY-SWITCH [EMTOP Sensor Board]

It reads the analog signal input from **AIN1**, value range is **0 ~ 4095**.

- root@arm:~# **while test 1; do cat /sys/bus/iio/devices/iio:device0/in_voltage0_raw; sleep 0.2; done**

```
4092
4044
3964
```

```
3960
3928
3824
3820
3820
3788
3740
3740
3744
```

When the switch is rotated, the read value will change.

Let us control PWM-LED with the rotary-switch:

- root@arm:~# **/test/app/rotary-pwmled**

Now, when rotating the switch, the PWM-LED brightness will change.

2.11.25 QSPIFLASH [EMTOP Sensor Board]

- root@arm:~# **cat /sys/class/mtd/mtd0/device/spi-nor/partname**

```
xt25f64
```

- root@arm:~# **cat /proc/mtd**

```
dev:    size   erasesize  name
mtd0: 00800000 00001000 "spi1.0"
```

Erase and format:

- root@arm:~# **flash_erase /dev/mtd0 0 0**

```
Erasing 8192 Kibyte @ 0 -- 100 % complete
```

- root@arm:~# **mount -t jffs2 /dev/mtdblock0 /mnt**

Write and read under directory **/mnt**, the content will keep in the QSPIFlash memory.

- root@arm:~# **umount /mnt**

Next boot, mount the flash and you can see the contents written before.

2.11.26 MATRIX KEYBOARD [EMTOP Sensor Board]

There is a 4 * 4 matrix keyboard on the Sensor Board.

- root@arm:~# **evtest**

```
Available devices:
/dev/input/event0:          tca8418
/dev/input/event1:          Goodix Capacitive TouchScreen
/dev/input/event2:          keys
Select the device event number [0-2]: 0
Input driver version is 1.0.1
Input device ID: bus 0x18 vendor 0x1 product 0x1 version 0x1
Input device name: "tca8418"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 256 (BTN_0)
    Event code 257 (BTN_1)
    Event code 258 (BTN_2)
    Event code 259 (BTN_3)
    Event code 260 (BTN_4)
    Event code 261 (BTN_5)
    Event code 262 (BTN_6)
    Event code 263 (BTN_7)
    Event code 264 (BTN_8)
    Event code 265 (BTN_9)
    Event code 304 (BTN_SOUTH)
    Event code 305 (BTN_EAST)
    Event code 306 (BTN_C)
    Event code 307 (BTN_NORTH)
    Event code 308 (BTN_WEST)
    Event code 309 (BTN_Z)
  Event type 4 (EV_MSC)
    Event code 4 (MSC_SCAN)
Properties:
Testing ... (interrupt to exit)
Event: time 1700202528.742025, type 4 (EV_MSC), code 4 (MSC_SCAN), value 0c
Event: time 1700202528.742025, type 1 (EV_KEY), code 306 (BTN_C), value 1
Event: time 1700202528.742025, ---------- SYN_REPORT ------------
```

```
Event: time 1700202528.902470, type 4 (EV_MSC), code 4 (MSC_SCAN), value 0c
Event: time 1700202528.902470, type 1 (EV_KEY), code 306 (BTN_C), value 0
Event: time 1700202528.902470, ---------- SYN_REPORT ------------
Event: time 1700202529.274165, type 4 (EV_MSC), code 4 (MSC_SCAN), value 08
Event: time 1700202529.274165, type 1 (EV_KEY), code 264 (BTN_8), value 1
Event: time 1700202529.274165, ---------- SYN_REPORT ------------
Event: time 1700202529.457756, type 4 (EV_MSC), code 4 (MSC_SCAN), value 08
Event: time 1700202529.457756, type 1 (EV_KEY), code 264 (BTN_8), value 0
Event: time 1700202529.457756, ---------- SYN_REPORT ------------
```