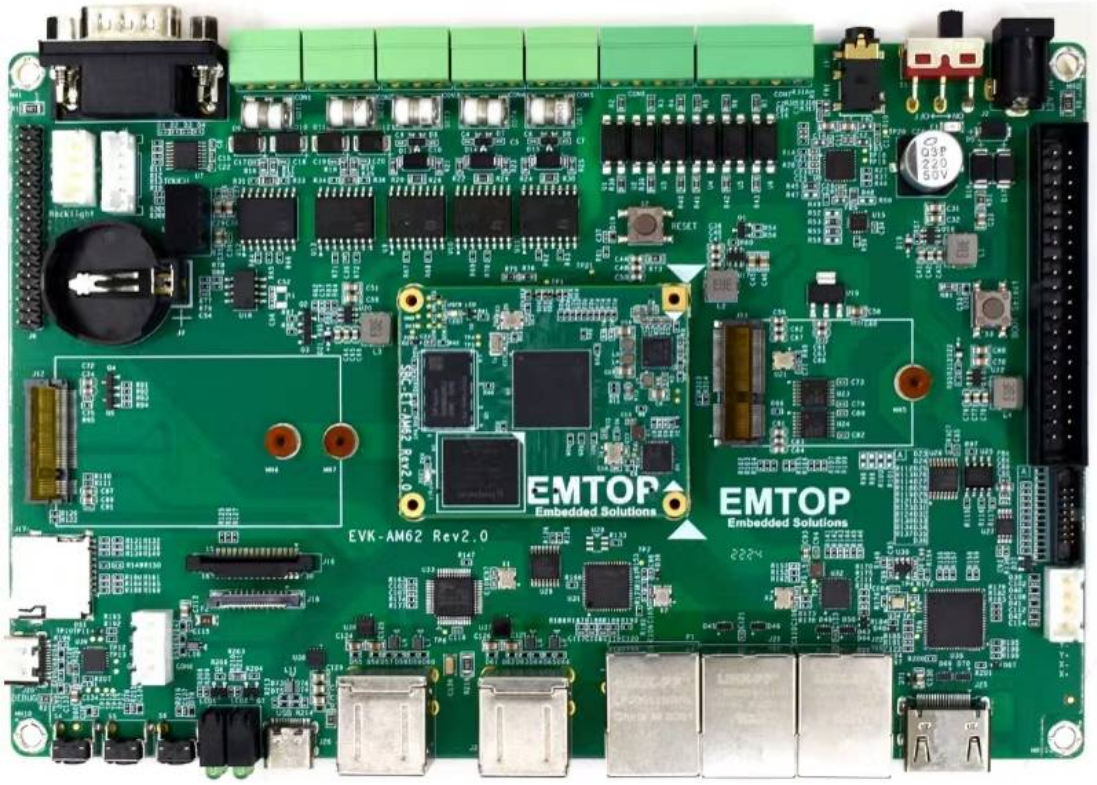
EVK-ET-AM62



SOM-ET-AM62 + EVK-ET-AM62

DEBIAN 13 [TRIXIE]
用户手册

Version: 0.1
2024-10-08

修订记录：

Version	Date	Description
0.1	2024-10-08	Initial Release

Table of Contents

1. DEBIAN 操作系统 6

1.1 软件资源 6

1.1.1 资源目录 6

1.1.2 BSP 6

1.2 嵌入式 LINUX 系统结构 7

1.3 搭建开发环境 8

1.3.1 安装交叉编译工具 8

1.3.2 设置交叉编译环境 9

1.4 准备源代码 9

1.5 编译 10

1.5.1 编译用户应用程序项目 12

1.6 LINUX 系统定制 13

1.6.1 替换 U-BOOT LOGO 13

1.6.2 设置配置菜单 14

1.6.3 菜单选项 14

1.6.4 编译 Kernel 15

1.7 内核驱动介绍 15

1.7.1 SD/MMC 17

1.7.2 Audio In/Out 18

1.8 驱动开发 19

1.8.1 GPIO_LEDs 驱动 19

1.8.2 PINMUX 配置指导 22

1.9 系统更新 25

1.9.1 更新 TF 卡系统镜像 25

1.9.2 从 TF 卡启动更新 eMMC 26

1.10 测试与演示 28

1.10.1 SSH 登录 29
1.10.2 RTC 29
1.10.3 TIMEZONE 设置 29
1.10.4 USB HOST 30
1.10.5 NETWORK 31
1.10.6 TFT-LCD 33
1.10.7 LVDS 34
1.10.8 LVDS BACKLIGHT 34
1.10.9 TOUCH PANEL 34
1.10.10 NAU88C22 AUDIO 36
1.10.11 HDMI 37
1.10.12 HDMI AUDIO 37
1.10.13 UART 38
1.10.14 RS485 39
1.10.15 EEPROM 40
1.10.16 CAN BUS 41
1.10.17 BUTTON 42
1.10.18 LED 43
1.10.19 DI/DO 44
1.10.20 SPI ADC 46
1.10.21 PWM 47
1.10.22 eMMC 47
1.10.23 SPIFLASH 47
1.10.24 M.2/KEY B [WIFI and BLUETOOTH] 48
1.10.25 M.2 WIFI 49
1.10.26 M.2 BLUETOOTH 51
1.10.27 M.2 4G/5G MODULE 52
1.10.28 MIPI-CSI CAMERA 55

1.10.29 WAYLAND GPU 57

1.10.30 QT6 GPU 58

1. Debian 操作系统

Debian 是完全由自由软件组成的类 UNIX 操作系统，其包含的多数软体使用 GNU 通用公共许可协议授权，并由 Debian 计划的参与者组成团队对其进行打包、开发与维护。

Note：

🕮 建议提前学习 Ubuntu Linux 安装、嵌入式 Linux 开发技术。

1.1 软件资源

随板提供的 DVR-ROM 包含演示、应用示例、Linux 源代码和工具，帮助您轻松快速地开发 Linux 应用程序和系统。

1.1.1 资源目录

您可以按照下表所示的信息找到 DVD-ROM 中包含的程序、代码等软件资源；

类别	目录
应用程序	
源码	CD\Source\u-boot-ti-2023.04
	CD\Debian\Source\linux-ti-6.6.32
	CD\Source\App
工具	CD\Tools\
映像	CD\Image

1.1.2 BSP

下表列出了 BSP 中包含的文件类型和格式：

Names		Note	Formats
BOOTLOADER	U-BOOT	MMC/SD	Source Code
		FAT	Source Code
		NET	Source Code
KERNEL	LINUX-6.6.32	Support JFFS2/EXT4/FAT/NFS various of file system	Source Code
DEVICE DRIVER	PMIC	PCA9450CHN driver	Source Code
	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
	MIPI-DSI	MIPI-DSI 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/BT	NXP 88W8987 driver	Source Code
ROOTFS	DEBIAN	Weston Desktop[Qt 6.7.2]	Image

1.2 嵌入式 Linux 系统结构

EVK-ET-AM62 默认搭载的是 Linux-6.6.32 eMMC 系统，该系统由 bootloader、kernel、rootfs 组成，下表是嵌入式 Linux 系统结构。

eMMC/SD		
Partition	FAT	EXT4
Image	Bootloader, DTB, Kernel	Debian Rootfs

1) Bootloader 是 u-boot 编译生成的程序，文件名为**tiboot3.bin**，**tispl.bin** 和**u-boot.img**。
2) 本文档所采用的内核是Linux-6.6.32，并根据硬件设计进行了定制。
3) Rootfs存储开源系统Debian，格式为EXT4。

1.3 搭建开发环境

在开发软件之前，用户需要在 PC 上建立 Linux 交叉开发环境。本节将以 **Ubuntu22.04** 操作系统为例介绍如何建立交叉开发环境。

强烈建议通过以下命令为新安装的 Ubuntu 安装必要的软件包。

- **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：

🕮 每条指令前均有“•”标记，以避免因长指令占用多行而造成混淆。

🕮 请注意每条指令内的空格；缺少任何空格都会导致执行指令失败。

1.3.1 安装交叉编译工具

我们在**Tools**目录下提供了交叉编译器：**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**和**gcc-linaro-7.5.0-2019.12-x86_64_aarch64-linux-gnu.tar.xz**。。

该编译器主要用于编译u-boot和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**

它将解压并安装在**/opt/bin/arm**目录下，保留默认设置。

1.3.2 设置交叉编译环境

运行以下命令设置源代码构建环境：

- **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：

- 可以在用户目录下的**.bashrc** 文件中添加指令，这样系统启动时就会自动加载添加的环境变量；
- 如果你想检查路径，请使用指令 **printenv PATH**

1.4 准备源代码

请从 **Source** 目录下获取源代码。

- **tar -xvf u-boot-ti-2023.04-git-xxxxxx.tar.xz**
- **tar -xvf linux-ti-6.6.32-git-xxxxxx.tar.xz**

然后我们就可以得到源代码目录 **u-boot-ti-2023.04** 和 **linux-ti-6.6.32**。

1.5 编译

1） 编译 Bootloader

运行以下命令编译 bootloader：

- **cd u-boot-ti-2023.04**
- **git checkout .**
- **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: 若安装在其它目录下，请根据您本地环境替换编译器路径。

DESTDIR: 指向存储目标映像的目录。

根据您的本地环境更改 **DESTDIR** 值以使其指向您的目标目录。

- **./make.sh setup**

该命令将安装编译过程需要调用的若干组件。

- **./make.sh**

所有命令成功完成后，您可以在 **DESTDIR** 目录下找到如下启动映像：

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

如果您只需要为 DDR1G、DDR2G 或 DDR4G 编译引导程序，请运行以下命令编译其中一个：

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

2） 编译 Kernel

执行如下指令编译内核：

- **cd linux-ti-6.6.32**
- **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=aarch64-linux-gnu-

DESTDIR="/dev/shm"
```

PATH: 若安装在其它目录下，请根据您本地环境替换编译器路径。

DESTDIR: 指向存储目标映像的目录。

根据您的本地环境更改 **DESTDIR** 值以使其指向您的目标目录。

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

如果编译成功，您可以在 **DESTDIR** 目录下找到名为**.dtb** 文件、**Image** 和 **lib/modules/6.6.32**。

Note：

The command **./make.sh**:

```
./make.sh                 # build dtbs and Image
./make.sh modules         # build dtbs, Image and driver modules
```

1.5.1 编译用户应用程序项目

让我们尝试编译一个 Qt 示例 **easing**，它来自 **qt-everywhere-src-6.7.2/qtbase/examples/widgets/animation/easing**。从 **Source/App/easing.tar.xz** 获取。下载到 ARM 板系统下，并按照如下步骤编译：

- root@arm:~# **cd easing**
- root@arm:~# **qmake**

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

- root@arm:~# **make -j4**

```
.....
g++ -c -pipe -O2 -Wall -Wextra -D_REENTRANT -DQT_NO_DEBUG -DQT_WIDGETS_LIB -DQT_GUI_LIB -DQT_CORE_LIB -I. -I/usr/include/aarch64-linux-gnu/qt6 -I/usr/include/aarch64-linux-gnu/qt6/QtWidgets -I/usr/include/aarch64-linux-gnu/qt6/QtGui -I/usr/include/aarch64-linux-gnu/qt6/QtCore -I. -I. -I/usr/lib/aarch64-linux-gnu/qt6/mkspecs/linux-g++ -o moc_window.o moc_window.cpp
g++ -Wl,-O1 -Wl,-rpath-link,/usr/lib/aarch64-linux-gnu -o easing  main.o window.o qrc_easing.o moc_window.o  /usr/lib/aarch64-linux-gnu/libQt6Widgets.so /usr/lib/aarch64-linux-gnu/libQt6Gui.so /usr/lib/aarch64-linux-gnu/libGLX.so /usr/lib/aarch64-linux-gnu/libOpenGL.so /usr/lib/aarch64-linux-gnu/libQt6Core.so -lpthread -lGLX -lOpenGL
```

- root@arm:~# **file easing**

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

运行它，您就可以看到动画界面：

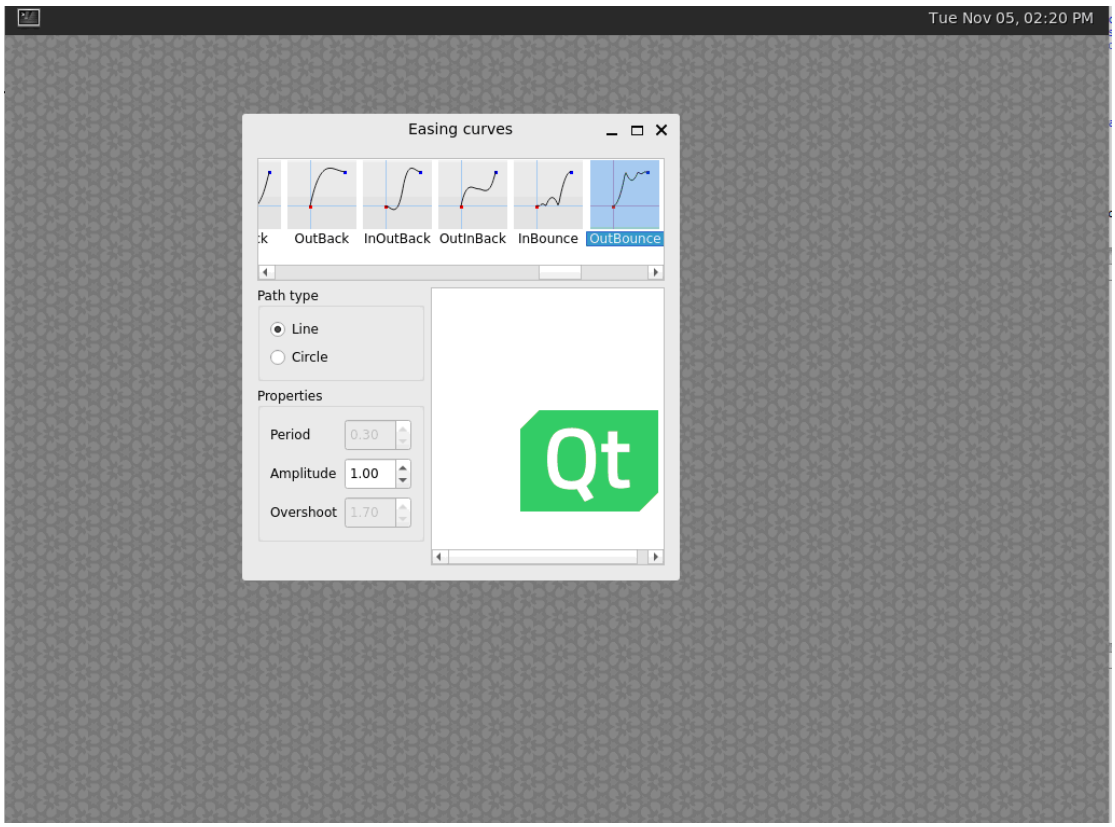


Figure 1-1 Qt easing Example

1.6 Linux 系统定制

为了满足客户的不同需求，开发人员常常需要在Linux内核默认配置的基础上进行一些定制化的修改，本章将通过实例介绍系统定制的过程。

1.6.1 替换 U-BOOT LOGO

[Not supported]

Note：

1.6.2 设置配置菜单

内核源代码下提供了一个默认的配置文件：

linux-ti-6.6.32/kernel/configs/emtop-sbc-et-am62.config

请执行以下命令进入配置菜单：

- **cd linux-ti-6.6.32**
- **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：

🕮 如果执行 **make ARCH=arm64 menuconfig** 命令时出现错误，您可能需要在 Ubuntu 系统中安装 ncurse，ncurses 是生成配置菜单所需的字符图形库，请输入以下指令安装该库：
sudo apt-get install libncurses5-dev

1.6.3 菜单选项

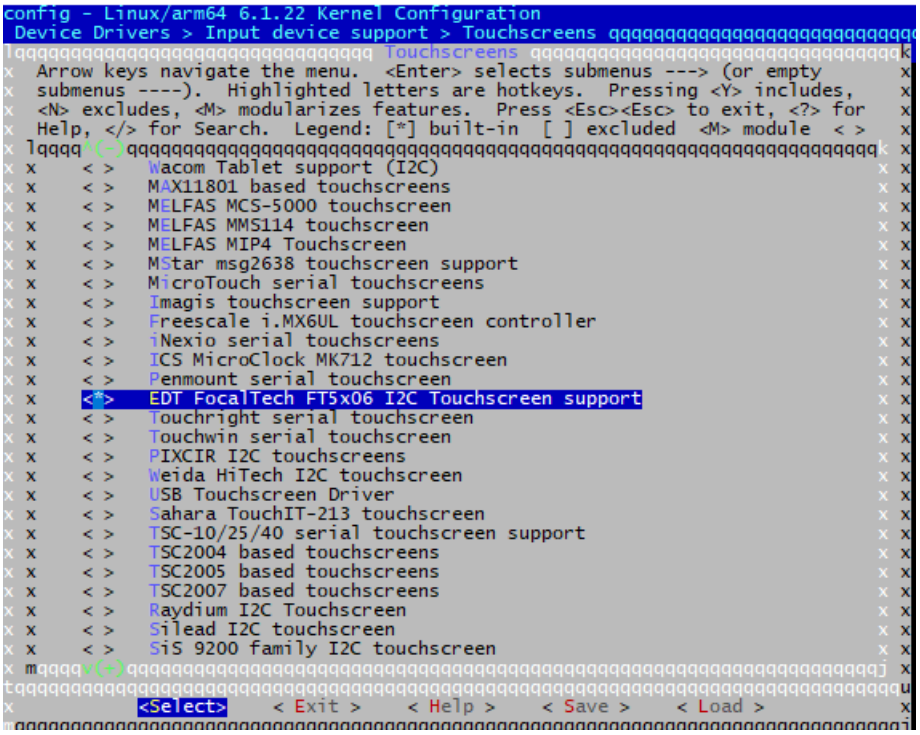
进入配置菜单后根据定制需求配置选项，例如，访问 **Device Drivers > Input device support > Touchscreens > EDT FocalTech FT5x06 I2C Touchscreen support** 如下图所示:

-> Device Drivers

-> Input device support

-> Touchscreens

-> EDT FocalTech FT5x06 I2C Touchscreen support



开启 Goodix I2C touchscreen to **<*>**，退出并保存。

1.6.4 编译 Kernel

请执行如下指令重新编译内核：

- **./make.sh**

该脚本不会覆盖 menuconfig 修改的配置。这意味着您修改的当前设置在目标内核映像中有效。

如果您想恢复默认配置，请删除文件**.config** 并运行**./make.sh**。

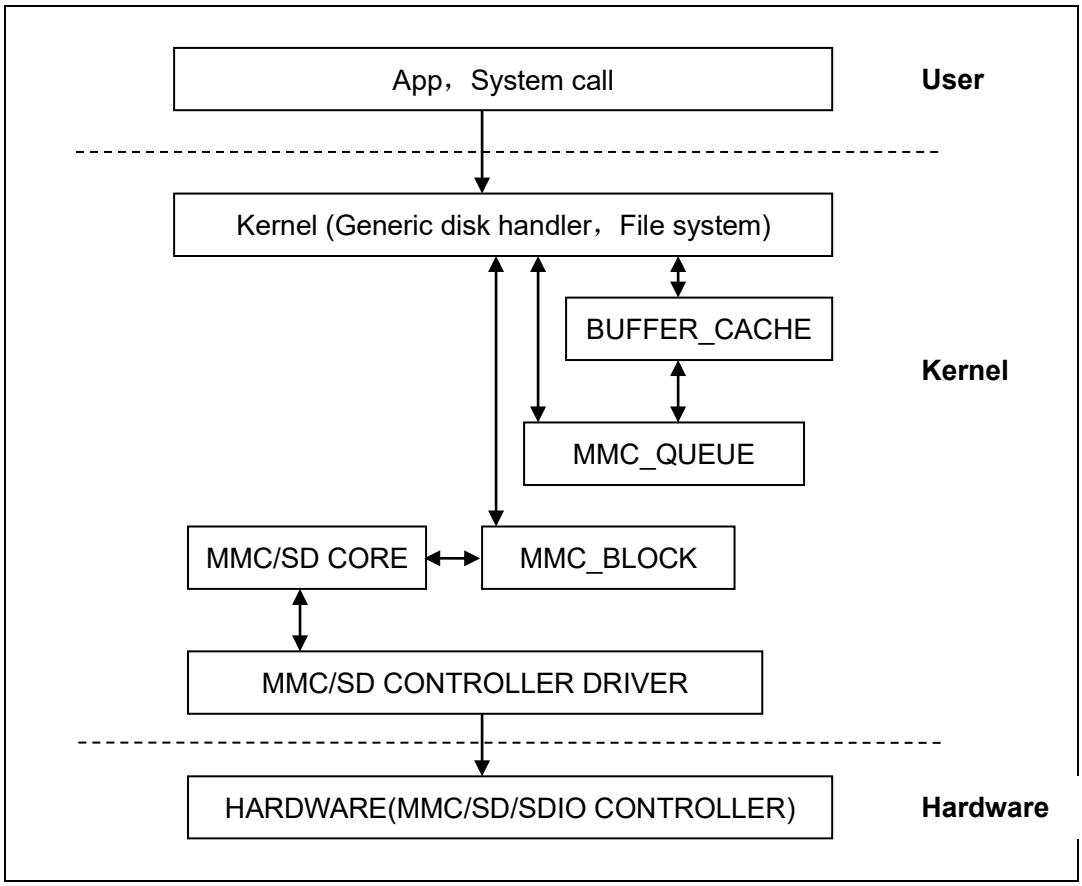
1.7 内核驱动介绍

下表显示了查找所有驱动程序的访问路径：

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-6.6.32	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
	MIPI-DSI	MIPI-DSI driver	drivers/gpu/drm/panel/panel-simple.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	NXP 88W8987 driver	3rdparty/mwifiex-lf-6.6.36_2.1.0
	CAMERA	CSI Camera driver	drivers/media/i2c/ov5640.c

1.7.1 SD/MMC



Linux 下的 SD/MMC 驱动主要由 SD/MMC core、mmc_block、mmc_queue 以及 SD/MMC 驱动组成：

1） SD/MMC core 实现 SD/MMC 卡操作中与结构体无关的代码；

2） mmc_block 实现 SD/MMC 卡作为块设备时的驱动结构；

3） mmc_queue 实现请求队列的管理；

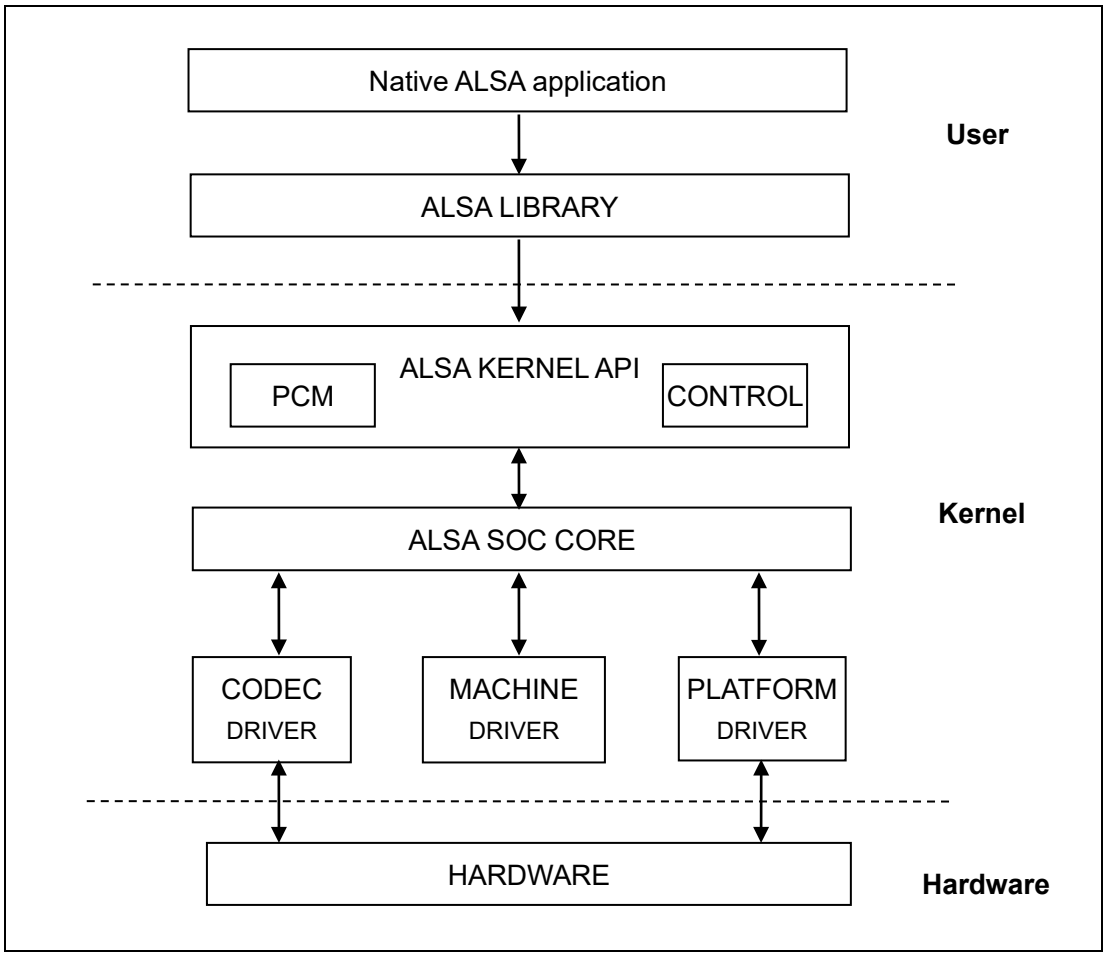
4） SD/MMC driver 实现具体的控制器驱动。

驱动及相关文件：

linux-ti-6.6.32/drivers/mmc/

linux-ti-6.6.32/drivers/mmc/host/sdhci_am654.c

1.7.2 Audio In/Out



ASoC 嵌入式音频系统基本上由三个组件组成：

1） 编解码器驱动程序：编解码器驱动程序独立于平台，包含音频控制、音频接口功能、编解码器 dapm 定义和编解码器 IO 功能；

2） 平台驱动程序：它包含该平台的音频 dma 引擎和音频接口驱动程序（例如 I2S、AC97、PCM）；

3） 板级驱动程序：板级驱动程序处理任何特定于机器的控件和音频事件，即在开始播放时打开放大器。

驱动及相关文件：

linux-ti-6.6.32/sound/soc/ti

linux-ti-6.6.32/sound/soc/codecs/nau8822.c

1.8 驱动开发

1.8.1 GPIO_LEDs 驱动

1）设备定义

linux-ti-6.6.32/arch/arm64/boot/dts/ti/emtop-evk-et-am62.dts

配置 GPIO0_12 作为系统运行状态指示灯，心跳闪烁。

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

    sys {
        label = "sys";
        gpios = <&main_gpio0 12 GPIO_ACTIVE_HIGH>;
        linux,default-trigger = "heartbeat";
    };
```

2）GPIO pinmux 配置

linux-ti-6.6.32/arch/arm64/boot/dts/ti/emtop-evk-et-am62.dts

将 OSPI0_CSN1 配置为 GPIO0_12 功能：

```
&main_pmx0 {
    ...
    usr_led_pins_default: usr-led-pins-default {
        pinctrl-single,pins = <
            AM62X_IOPAD(0x030, PIN_OUTPUT, 7) /* (G21) OSPI0_CSN1.GPIO0
_12 */
        >;
    };
};
```

3）Driver 设计

linux-ti-6.6.32/drivers/leds/leds-gpio.c

a) 调用 platform_driver_register 注册 gpio_leds 驱动

```
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) 申请 gpio，调用 led_classdev_register 注册 led_classdev 驱动。

```
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) 用户可以访问文件**/sys/class/leds/sys/brightness**，它将调用 gpio_led_set 设置 LED 的状态。

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

1.8.2 PINMUX 配置指导

AM625 有两种 GPIO：一种由 A53 控制，另一种由 MCU 控制：

A53 GPIO 命名为：GPIO0_12、GPIO1_31

MCU GPIO 命名为：MCU_GPIO0_22。

配置 A53 GPIO 引脚属性：

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

配置 MCU GPIO 引脚属性：

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

函数语法：

```
AM62X_IOPAD(pa, val, muxmode)      or   AM62X_MCU_IOPAD(pa, val, muxmode)
```

pa: Physical Address
val: value to write
muxmode: MUXMODE[3:0]

它们在 **linux-ti-6.6.32/arch/arm64/boot/dts/ti/k3-pinctrl.h** 中定义：

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

现在我们来说明一下如何计算 PMIC_LPM_EN0 的参数'pa'。

打开文档《AM62x SitaraTM Processors》，找到 PMIC_LPM_EN0 引脚的物理地址为 0x04084080：

B7	C7	PMIC_LPM_EN0 PADCONFIG: MCU_PADCONFIG32 0x04084080	PMIC_LPM_EN0 MCU_GPIO0_22

- **vi linux-ti-6.6.32/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>;
};
```

它指出 mcu_pmx0 的基物理地址是 0x04084000，那么我们只需要通过宏 AM62X_MCU_IOPAD 传递引脚 PMIC_LPM_EN0 的偏移地址就可以了：

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

关于参数‘val’，您可以从以下项目中选择：

- **vi linux-ti-6.6.32/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)
```

关于参数‘muxmode’，找到文档《AM62x SitaraTM Processors》的‘Pin Attributes’表：

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

可以看到 MCU_GPIO0_22 的 MUXMODE 是 7。

如果你的目标引脚由 A53 控制，请将其附加到 dts 文件中的 main_pmx0 节点下；否则将其放在 mcu_pmx0 节点下：

```
&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 */
```

```
            >;
        };
        ......
}
```

1.9 系统更新

EVK-ET-AM62 可以从 TF 卡和 eMMC 启动，由 **BOOT Select** 按钮决定：

按下 [不松开]：从 TF 卡启动

否则：从 eMMC 启动。

1.9.1 更新 TF 卡系统镜像

1） 制作 TF 启动卡

- 从 **Image** 目录获取系统映像，命名为 **EVK-ET-AM62-SD-REVXX.img.xz**，解压并获取原始映像 **EVK-ET-AM62-SD-REVXX.img**。
- 如果您在 Windows 系统下，请运行 **Tools/win32diskimager** 将 **EVK-ET-AM62-SD-REVXX.img** 写入 TF 卡；如果您在 Linux 系统下，请使用 dd 命令将 **EVK-ET-AM62-SD-REVXX.img** 写入 TF 卡。

2） 更新 U-Boot

如果您对 u-boot 源代码进行了修改，需要将其更新到 TF 卡中，请将目标映像复制到 TF 卡 FAT 分区的根目录中：

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

3） 更新 Kernel

如果您修改了内核源代码，请更新 TF 卡分区 1 [FAT32]下的 dtb 和 Image。该分区可以被 Windows 或 Linux 识别。

4） 更新 Rootfs

由于 Windows 下无法访问 EXT4，请在 Ubuntu 下挂载 TF 卡的第 2 分区，更改目标文件后卸载该卡。

Note:

🕮 如果 eMMC 中已经写入系统映像，请擦除 eMMC 然后重新启动主板，因为主板将默认首先尝试从 eMMC 启动。

🕮 输入 u-boot 命令，擦除 eMMC：

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

1.9.2 从 TF 卡启动更新 eMMC

Option 1: 将完整映像写入 eMMC

- 制作可启动的 TF 卡并启动系统；
- 选择目标镜像[在 **Image/**目录下]并将其复制到 U 盘[格式化为 NTFS 或 exFAT]。如果是**.xz** 文件，请将其解压以生成**.img** 文件；
- 在 ARM 板上安装 USB 盘，例如 USB 盘被识别为 sda；
  - root@arm:**~# mount /dev/sda /mnt**
- 运行命令开始写入 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/EVK-ET-AM62-SD-REVXX.img of=/dev/mmcblk0 status=progress bs=4M**

- 运行命令写入引导程序[必须步骤]：
  - 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

完成后，关闭 ARM 板电源，取出 TF 卡，然后重新启动 ARM 板，它应该从 eMMC 启动并进入 Linux 提示符。

Option 2: 将 TF 卡系统同步到 eMMC

- 制作可启动的 TF 卡并启动系统；
- 运行命令开始写入 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. O
pts: (null). Quota mode: none.
```

- 关闭开发板电源并取出 TF 卡。

1.10 测试与演示

本节将对外围设备运行一些测试。

电源: **12V DC**

调试串口: **UART0**, **115200 1N8**, **USB TypeC slot [J20]**

1.10.1 SSH 登录

SSH 服务器默认已经开启，请获取 ARM 板上有线或无线网络的本地 IP，然后从 PC 端使用 SSH 客户端（如 PuTTY）登录，**root** 账户，无密码。

1.10.2 RTC

板载有一颗 RTC 芯片 RX8010SJ，并且 CPU 内置 RTC 也是默认开启的，所以系统下有两个 RTC 设备可以访问。

- 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
```

即 rtc0 为 RX8010SJ，rtc1 为内置 RTC，**hwclock** 命令默认访问**/dev/rtc0**，若要访问**/dev/rtc1**，请附加参数：**-f /dev/rtc1**。

我们将当前时间设置为 2024-02-05 10:12：

- root@arm:~# **date -s "2024-02-05 10:12"; hwclock -f /dev/rtc0 -w**

重新启动开发板，并使用以下命令检查硬件 RTC 时间：

- root@arm:~# **hwclock -f /dev/rtc0**

```
2024-02-05 10:12:07.365014+00:00
```

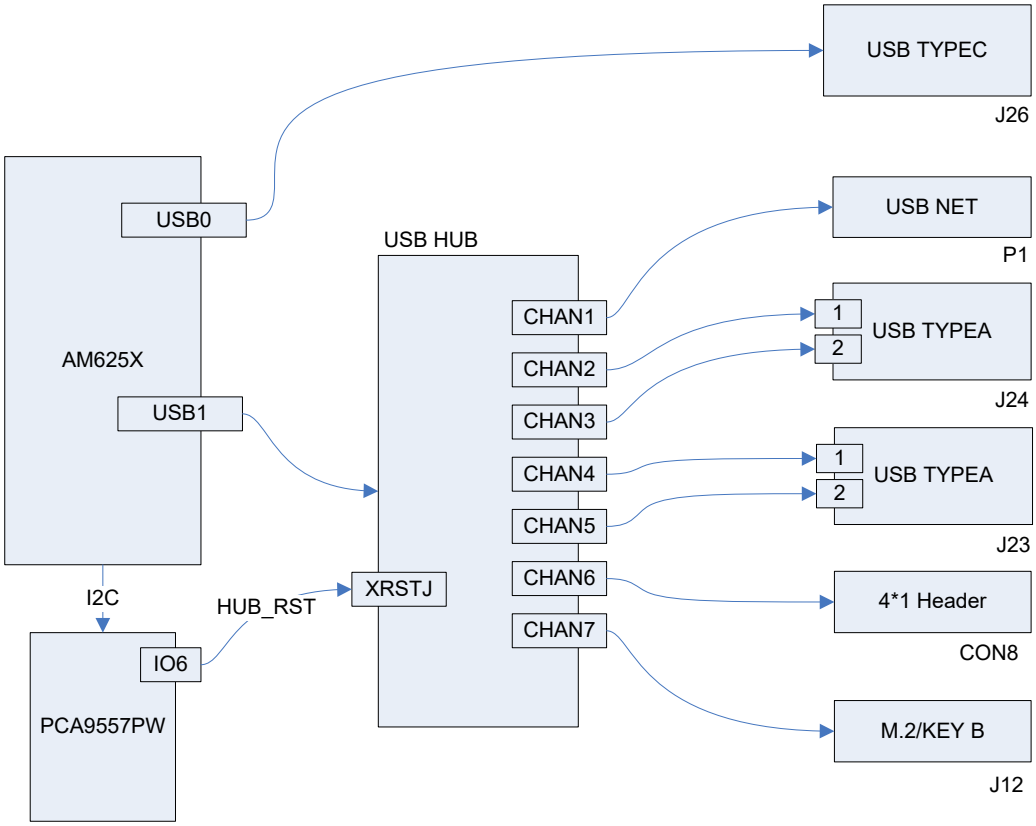
1.10.3 TIMEZONE 设置

以设置北京时间为例：

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

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

1.10.4 USB HOST



我们可以用 U 盘测试插槽 **J26**、**J24** 和 **J23**。在这些插槽上安装 U 盘，检查以下消息：

```
[  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
```

- root@arm:~# **mount /dev/sda1 /mnt**

```
[  567.749215] FAT-fs (sda1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
```

Debian 系统不会自动挂载 USB 磁盘，请手动挂载。

1.10.5 NETWORK

板上有两块 1Gbps 网络芯片 RTL8211F，一块 USB 转网卡 LAN9500A。有线网络由 NetworkManager 控制。

- root@arm:~# **nmcli dev**

```
DEVICE   TYPE       STATE                      CONNECTION
lo       loopback   connected (externally)   lo
enu1u1   ethernet   unavailable                --
eth0     ethernet   unavailable                --
eth1     ethernet   unavailable                --
can0     can        unmanaged                  --
can1     can        unmanaged                  --
can2     can        unmanaged                  --
```

插入网线：

```
[  347.425305] am65-cpsw-nuss 8000000.ethernet eth0: Link is Up - 1Gbps/Full - flow control rx/tx
```

- root@arm:~# **nmcli dev**

```
DEVICE   TYPE       STATE                      CONNECTION
eth0     ethernet   connected                  Wired connection 2
lo       loopback   connected (externally)   lo
enu1u1   ethernet   unavailable                --
eth1     ethernet   unavailable                --
can0     can        unmanaged                  --
can1     can        unmanaged                  --
can2     can        unmanaged                  --
```

- 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 功能默认启用。ARM 板可以自动从本地网络中的 DHCP 服务器请求有效的 IP 地址。

- 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
```

如果需要断开 eth0：

- root@arm:~# **nmcli dev disconnect eth0**

```
Device 'eth0' successfully disconnected.
```

配置静态 IP:

- root@arm:~# **nmcli con add type ethernet con-name EMTOP-ETH0 ifname eth0 ip4 192.168.1.102/24 gw4 192.168.1.1**

```
Connection 'EMTOP-ETH0' (6ba9a893-8d80-41a1-a0c1-d48f054ae83b) successfully added.
```

- root@arm:~# **nmcli con up EMTOP-ETH0 ifname eth0**

Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/5)
Parameter: * **EMTOP-ETH0**: 指的是应用配置文件/etc/NetworkManager/system-connections/EMTOP-ETH0.nmconnection

检查 IP 配置：

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

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.102  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::e5e4:12:5ac7:7d2  prefixlen 64  scopeid 0x20<link>
        ether 1c:63:49:22:d2:e0  txqueuelen 1000  (Ethernet)
        RX packets 864  bytes 54478 (53.2 KiB)
        RX errors 0  dropped 820  overruns 0  frame 0
        TX packets 96  bytes 6763 (6.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

恢复 DHCP：

- root@arm:~# **nmcli con up eth0 ifname eth0**

Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/7)
Parameter: * **eth0** 位于"nmcli con up"后面: 指的是应用配置文件/etc/NetworkManager/system-connections/eth0.nmconnection

然后，eth0 将请求一个动态 IP 地址。

Note:

🕮 ARM 板重启后当前配置依然有效，不会恢复为默认配置。

1.10.6 TFT-LCD

已测试的设备：

MODEL	DESCRIPTION	DTB
LCD8000-70T	800 * 480, with touch panel	emtop-evk-et-am62-lcd8000-800x480.dtb

编辑 **uEnv.txt**：让 **name_fdt** 指向上表中的 DTB。

1.10.7 LVDS

已测试的设备：

MODEL	DESCRIPTION	DTB
VISLCD-101HYS145ACT02	1280 * 720, with touch panel	emtop-evk-et-am62.dtb

编辑 **uEnv.txt**：让 **name_fdt** 指向上表中的 DTB。

Figure 1-2 VISLCD-101HYS145ACT02

1.10.8 LVDS BACKLIGHT

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

Note:

背光值范围： **0 ~ 8**.

1.10.9 TOUCH PANEL

MODEL	TYPE	I2C BUS
GT9271	I2C CTP	I2C3

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

```
No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:      30370000.snvs:snvs-powerkey
/dev/input/event1:      generic gt9271
/dev/input/event2:      gpio-keys
Select the device event number [0-2]: 1
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 330 (BTN_TOUCH)
  Event type 3 (EV_ABS)
    Event code 0 (ABS_X)
      Value      799
      Min          0
      Max        799
    Event code 1 (ABS_Y)
      Value      479
      Min          0
      Max        479
    Event code 47 (ABS_MT_SLOT)
      Value        0
      Min          0
      Max          9
    Event code 53 (ABS_MT_POSITION_X)
      Value        0
      Min          0
      Max        799
    Event code 54 (ABS_MT_POSITION_Y)
      Value        0
      Min          0
      Max        479
    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 1707131270.474572, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X),
value 93
Event: time 1707131270.474572, type 3 (EV_ABS), code 54 (ABS_MT_POSITION_Y),
value 93
Event: time 1707131270.474572, type 3 (EV_ABS), code 0 (ABS_X), value 93
Event: time 1707131270.474572, type 3 (EV_ABS), code 1 (ABS_Y), value 93
Event: time 1707131270.474572, ---------- SYN_REPORT ------------
Event: time 1707131270.641322, type 3 (EV_ABS), code 57 (ABS_MT_TRACKING_ID),
```

```
value -1
Event: time 1707131270.641322, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 0
Event: time 1707131270.641322, ---------- SYN_REPORT ------------
Event: time 1707131271.588488, type 3 (EV_ABS), code 57 (ABS_MT_TRACKING_ID), value 5
Event: time 1707131271.588488, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value 156
Event: time 1707131271.588488, type 3 (EV_ABS), code 54 (ABS_MT_POSITION_Y), value 114
Event: time 1707131271.588488, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 1
Event: time 1707131271.588488, type 3 (EV_ABS), code 0 (ABS_X), value 156
Event: time 1707131271.588488, type 3 (EV_ABS), code 1 (ABS_Y), value 114
Event: time 1707131271.588488, ---------- SYN_REPORT ------------
Event: time 1707131271.791086, type 3 (EV_ABS), code 57 (ABS_MT_TRACKING_ID), value -1
Event: time 1707131271.791086, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 0
Event: time 1707131271.791086, ---------- SYN_REPORT ------------
Event: time 1707131272.186580, type 3 (EV_ABS), code 57 (ABS_MT_TRACKING_ID), value 6
Event: time 1707131272.186580, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value 107
Event: time 1707131272.186580, type 3 (EV_ABS), code 54 (ABS_MT_POSITION_Y), value 84
Event: time 1707131272.186580, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 1
Event: time 1707131272.186580, type 3 (EV_ABS), code 0 (ABS_X), value 107
Event: time 1707131272.186580, type 3 (EV_ABS), code 1 (ABS_Y), value 84
Event: time 1707131272.186580, ---------- SYN_REPORT ------------
Event: time 1707131272.361357, type 3 (EV_ABS), code 57 (ABS_MT_TRACKING_ID), value -1
Event: time 1707131272.361357, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 0
Event: time 1707131272.361357, ---------- SYN_REPORT ------------
```

1.10.10 NAU88C22 AUDIO

- 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
```

播放：

- root@arm:~# **for wav in `ls /usr/share/sounds/alsa/*.wav`; do aplay $wav; done**

1.10.11 HDMI

MODEL	DESCRIPTION	DTB
HDMI Display		emtop-evk-et-am62.dtb emtop-evk-et-am62-hdmi.dtb

emtop-evk-et-am62.dtb：支持 LVDS 和 HDMI 双显示

emtop-evk-et-am62-hdmi.dtb：仅支持 HDMI 显示，带 HDMI 音频。

编辑 **uEnv.txt**：让 **name_fdt** 指向上表中的 DTB。

1.10.12 HDMI AUDIO

MODEL	DESCRIPTION	DTB
HDMI Display	支持音频播放	emtop-evk-et-am62-hdmi.dtb

编辑 **uEnv.txt**：让 **name_fdt** 指向上表中的 DTB。

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

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

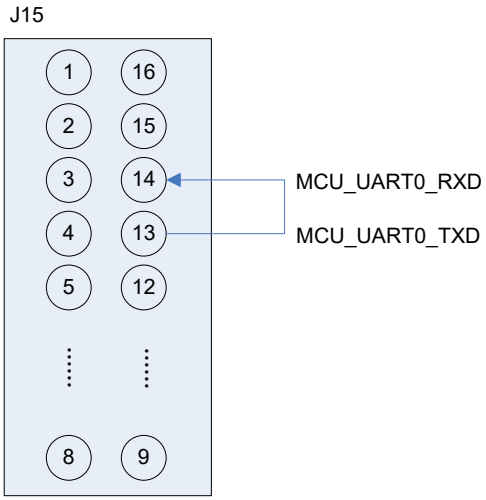
播放：

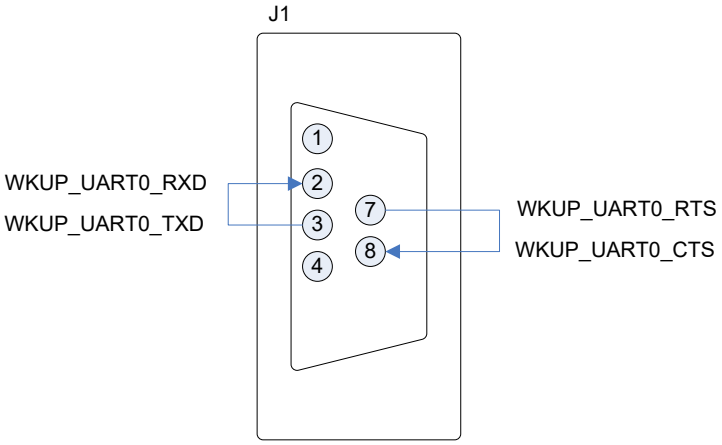
- root@arm:~# **for wav in `ls /usr/share/sounds/alsa/*.wav`; do aplay $wav; done**

1.10.13 UART

DEVICE NODE	HARDWARE	USAGE
/dev/ttyS2	UART0	DEBUG PORT
/dev/ttyS3	UART1	RS485
/dev/ttyS7	UART5	RS485
/dev/ttyS8	UART6	BLUETOOTH
/dev/ttyS9	MCU_UART0	HEADER J15
/dev/ttyS10	WKUP_UART0	DB9 J1

这里先不要测试 UART0、UART1、UART5 和 UART6。我们来测试其他的 [MCU_UART0 和 WKUP_UART0]。连接它们的 RXD 和 TXD 引脚：





并运行以下命令：

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

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

默认波特率为 **115200**。如果要指定其他特定波特率：

- root@arm:~# **/test/app/com -d /dev/ttyS9 -b 9600**

所有支持的波特率请参阅源代码 **com.tar.xz**。

- root@arm:~# **/test/app/com -d /dev/ttyS10 -f**

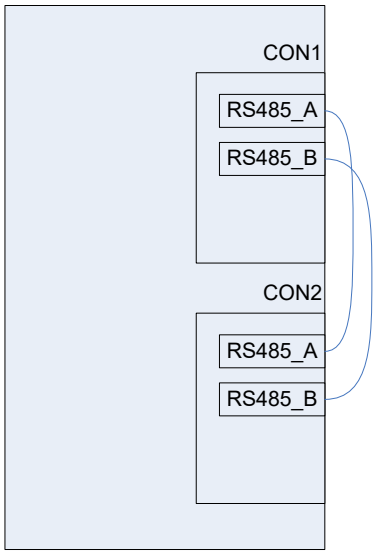
```
SEND: 1234567890
RECV: 1234567890
SEND: 1234567890
RECV: 1234567890
```

参数**-f**：启用硬件流控制[RTS/CTS]功能。

1.10.14 RS485

DEVICE NODE	HARDWARE	USAGE	REMARK
/dev/ttyS3	UART1	RS485	
/dev/ttyS7	UART5	RS485	with RTS

There are 2 RS485 bus on board, lets connect them together.



EVK-ET-AM62

- root@arm:~# **/test/app/com -d /dev/ttyS3 -s "Hello world" &**
- root@arm:~# **/test/app/com -d /dev/ttyS7 -m rs485**

```
SEND: 1234567890
SEND: Hello world
RECV: 1234567890
RECV: Hello world
SEND: 1234567890
SEND: Hello world
RECV: 1234567890
RECV: Hello world
```

1.10.15 EEPROM

核心板上有一个 AT24LC32A，并启用了写保护。

- 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  |................|
```

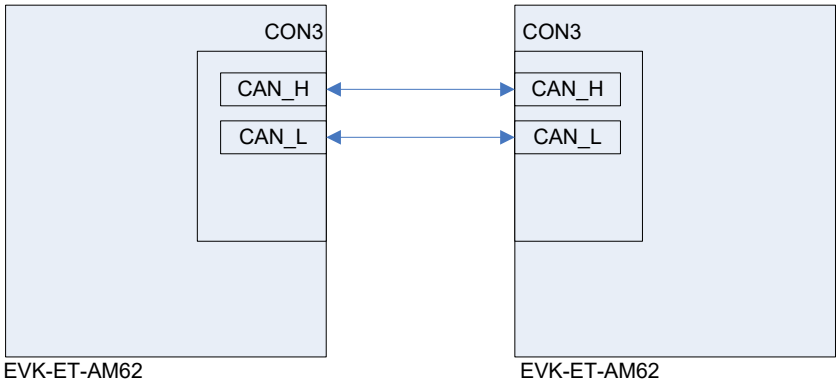
```
00000030    ff ff ff ff ff ff ff ff    ff ff ff ff ff ff ff ff    |................|
00000040    ff ff ff ff ff ff ff ff    ff ff ff ff ff ff ff ff    |................|
00000050    ff ff ff ff ff ff ff ff    ff ff ff ff ff ff ff ff    |................|
......
```

1.10.16 CAN BUS

板上有 3 条 CAN 总线。

ITEM	HW INTERFACE	REGISTER BASEADDR	LINUX INTERFACE	SLOT
1	MCAN0	0x20701000	can2	CON3
2	MCU_MCAN0	0x4e00000	can0	CON4
3	MCU_MCAN1	0x4e10000	can1	CON5

将 2 个 CAN 总线与 2 个 EVK-ET-AM62 连接：



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

```
can0: flags=128<NOARP>  mtu 16
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 10  (U
NSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

配置参数[双方]：

- root@arm:~# **ifconfig can0 down**
- root@arm:~# **ip link set can0 type can bitrate 125000**
- root@arm:~# **ip link set can0 type can restart-ms 100**

- root@arm:~# **ifconfig can0 up**

开始在一个板上监听：

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

另一块板上发送：

- root@arm:~# **cansend can0 "5A1#1122334455667788"**

更多信息请参考项目 can-utils。

1.10.17 BUTTON

PMIC_PBn [S4] 按键：

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

```
No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:          keys
/dev/input/event1:          tps65219-pwrbutton
/dev/input/event2:          ADS7846 Touchscreen
Select the device event number [0-2]: 1
Input driver version is 1.0.1
Input device ID: bus 0x18 vendor 0x0 product 0x0 version 0x0
Input device name: "tps65219-pwrbutton"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 116 (KEY_POWER)
Properties:
Testing ... (interrupt to exit)
Event: time 1730897276.000281, type 1 (EV_KEY), code 116 (KEY_POWER), value 1
Event: time 1730897276.000281, ---------- SYN_REPORT ------------
Event: time 1730897276.192594, type 1 (EV_KEY), code 116 (KEY_POWER), value 0
Event: time 1730897276.192594, ---------- SYN_REPORT ------------
```

用户按键[**S5, S6**]:

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

```
No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:          keys
/dev/input/event1:          tps65219-pwrbutton
/dev/input/event2:          ADS7846 Touchscreen
Select the device event number [0-2]: 0
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "gpio-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)
Event: time 1707132374.777133, type 1 (EV_KEY), code 102 (KEY_HOME), value 1
Event: time 1707132374.777133, ---------- SYN_REPORT ------------
Event: time 1707132374.904456, type 1 (EV_KEY), code 102 (KEY_HOME), value 0
Event: time 1707132374.904456, ---------- SYN_REPORT ------------
Event: time 1707132375.518704, type 1 (EV_KEY), code 105 (KEY_LEFT), value 1
Event: time 1707132375.518704, ---------- SYN_REPORT ------------
Event: time 1707132375.615938, type 1 (EV_KEY), code 105 (KEY_LEFT), value 0
Event: time 1707132375.615938, ---------- SYN_REPORT ------------
```

1.10.18 LED

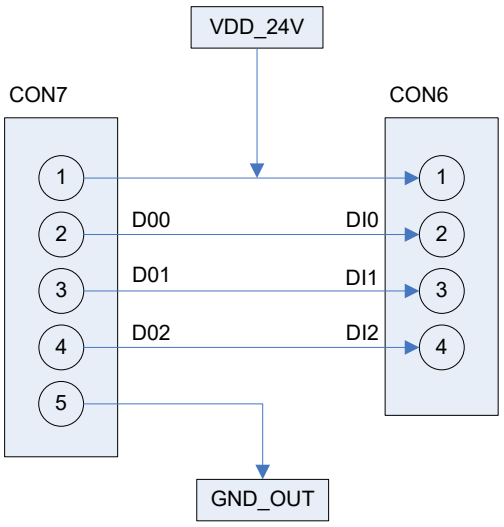
LED	GPIO	LINUX DEVICE
SYS_nLED	GPIO0_12	/sys/class/leds/sys

板上可控制的 LED 只有一个，SYS_nLED 持续闪烁以指示系统运行状态。但我们可以让它像普通 LED 一样工作：

- 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**

您可以看到 **SYS_nLED** 以 2Hz 的频率闪烁。

1.10.19 DI/DO



DI/DO	GPIO	BASEADDR[HEX]	IO OFFSET
DO0	GPIO0_0	600000	0
DO1	GPIO0_3	600000	3
DO2	GPIO0_4	600000	4
DI0	GPIO0_5	600000	5
DI1	GPIO0_2	600000	2
DI2	GPIO0_6	600000	6

我们可以使用以下命令获取 GPIOCHIP 编号：

- root@arm:~# **gpiochip=`gpiodetect | awk '/600000/ {print $1}'`**

Note:

600000 为对应 GPIO 的 **BASEADDR** 十六进制数值。

DO0 输出低电平：

- root@arm:~# **gpioset $gpiochip 0=0**

DO0 输出高电平：

- root@arm:~# **gpioset $gpiochip 0=1**

DO1 输出低电平：

- root@arm:~# **gpioset $gpiochip 3=0**

Note:

🕮 **gpioset <GPIOCHIP> <IO OFFSET>=<output value>**

读取 DI0 输入：

- root@arm:~# **gpioget $gpiochip 5**

```
1 or 0
```

Note:

🕮 **gpioget <GPIOCHIP> <IO OFFSET>**

或者监控 IO 状态改变事件：

- root@arm:~# **gpiomon $gpiochip 5**

```
event:  RISING EDGE offset: 5 timestamp: [      1151.814356387]
event:  FALLING EDGE offset: 5 timestamp: [      1151.815449803]
event:  RISING EDGE offset: 5 timestamp: [      1152.091556803]
```

Note:

🕮 **libgpiod version is 1.6.3.**

🕮 请勿使用命令 **apt-get install gpiod** 安装或更新 libgpiod。该版本的用法与当前版本不同。

1.10.20 SPI ADC

底板上有一个 SPI ADC 芯片 ADC102S051。

- root@arm:~# **cat /sys/bus/iio/devices/iio:device0/name**

```
adc102s051
```

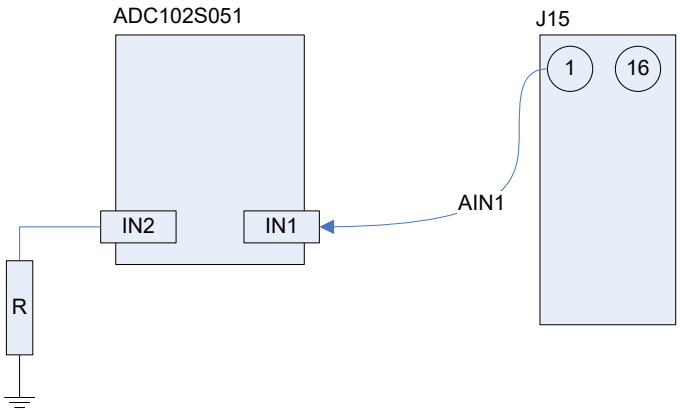


Figure 1-3 SPI ADC 示意图

读取 IN1 通道输入：

- root@arm:~# **cat /sys/bus/iio/devices/iio:device0/in_voltage0_raw**

```
1884
```

- root@arm:~# **cat /sys/bus/iio/devices/iio:device0/in_voltage0_scale**

```
0.805664062
```

公式：

Voltage = **raw** * **scale** (V)

读取 IN2 通道输入：

- root@arm:~# **cat /sys/bus/iio/devices/iio:device0/in_voltage1_raw**

```
3
```

通道 IN2 连接到 GND，因此其读数始终近似为 0。

1.10.21 PWM

底板接头[J15]上的 MCU_GPIO0_8 默认配置为 PWM 输出。

- root@arm:~# **/test/app/mcu_timer1 1000**

上述命令可以设置其输出 1KHz，占空比为 50%。

- root@arm:~# **/test/app/mcu_timer1 1000 80**

上述命令可以设置其输出 1KHz，占空比为 80%。

- root@arm:~# **/test/app/mcu_timer1 -h**

```
Usage:
    /test/app/mcu_timer1 frequency [duty_circle] - Configure MCU_TIMER1 output under
PWM mode

    frequency                unit: Hz, maximum 17000000Hz[17MHz]
    duty_circle              percentage, valid value: 0,10,20,...,90,100, 50 as default

Example:
    /test/app/mcu_timer1 1000          output 1KHz with 50% duty circle
    /test/app/mcu_timer1 1000 80       output 1KHz with 80% duty circle
    -h                                 display help info
```

1.10.22 eMMC

eMMC 主要用于保存系统镜像，无需手动测试。

1.10.23 SPIFLASH

底板上配备了 SPIFlash XT25F64BSOIGT。

- root@arm:~# **dmesg |grep -i spi**

```
[      7.035063] spi-nor spi0.1: unrecognized JEDEC id bytes: 0b fb 7f ff ff ff
```

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

```
dev:      size     erasesize    name
mtd0: 00400000 00010000 "30bb0000.spi"
```

擦除格式化：

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

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

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

在**/mnt** 目录下进行写入和读取，内容将保留在 QSPIFlash 存储器中。

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

下次启动时挂载此 FLASH，就可以看到之前写入的内容。

1.10.24 M.2/KEY B [WIFI and BLUETOOTH]

已测试的设备：

MODEL	CHIPSET	RESOLUTION
1ZM M.2 Module	NXP 88W8987	Support Wi-Fi and Bluetooth 5.1

Figure 1-4 1ZM M.2 Module

1.10.25 M.2 WIFI

- root@arm:~# **modprobe moal sta_name=wlan uap_name=wlan wfd_name=p2p max_vir_bss=1 cfg80211_wext=0xf cal_data_cfg=none fw_name=sdiouart8987_combo_v0.bin**

```
[    27.300912] mlan: loading out-of-tree module taints kernel.
[    27.391413] wlan: Loading MWLAN driver
[    27.392160] wlan: Register to Bus Driver...
[    27.478548] vendor=0x02DF device=0x9149 class=0 function=1
[    27.478688] Attach moal handle ops, card interface type: 0x105
[    27.478716] rps set to 0 from module param
[    27.478721] No module param cfg file specified
[    27.478736] SDIO: max_segs=128 max_seg_size=65536
[    27.478750] rx_work=1 cpu_num=4
[    27.478806] Attach mlan adapter operations.card_type is 0x105.
[    27.479449] wlan: Enable TX SG mode
[    27.479455] wlan: Enable RX SG mode
```

```
[    27.486150] Request firmware: sdiouart8987_combo_v0.bin
[    27.791167] Wlan: FW download over, firmwarelen=612304 downloaded 612304
[    28.662012] WLAN FW is active
[    28.662030] on_time is 28660259810
[    28.721744] fw_cap_info=0x181d7f03, dev_cap_mask=0xffffffff
[    28.721921] max_p2p_conn = 8, max_sta_conn = 8
[    28.722567] SDIO rx aggr: 1 block_size=512
[    28.722648] wlan: Enable RX SG mode
[    28.722651] mpa_rx_buf_size=65280
[    28.750785] Register NXP 802.11 Adapter wlan0
[    28.762135] Register NXP 802.11 Adapter wlan1
[    28.774227] Register NXP 802.11 Adapter p2p0
[    28.774334] wlan: version = SD8987----16.92.21.p76.5-MM5X16391.p3-GPL-(FP92)
[    28.775489] wlan: Register to Bus Driver Done
[    28.775511] wlan: Driver loaded successfully
```

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

```
[   106.018542] wlan: wlan0 START SCAN
[   111.007459] wlan: SCAN COMPLETED: scanned AP count=23
        SSID: EMTOP
......
```

- 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
[  254.521484] wlan: wlan0 START SCAN
[  259.510371] wlan: SCAN COMPLETED: scanned AP count=21
[  259.516932] wlan: HostMlme wlan0 send auth to bssid dc:XX:XX:XX:53:70
[  259.517747] wlan0:
[  259.517755] wlan: HostMlme Auth received from dc:XX:XX:XX:53:70
[  259.522576] CMD_RESP: cmd 0x121 error, result=0x2
[  259.522608] IOCTL failed: 000000005a18a418 id=0x200000, sub_id=0x200024 action=2, status_code=0x3
[  259.528757] wlan: HostMlme wlan0 Connected to bssid dc:XX:XX:XX:53:70 successfully
[  259.531799] wlan0:
[  259.531820] wlan: Send EAPOL pkt to dc:XX:XX:XX:53:70
[  259.539604] wlan0:
[  259.539632] wlan: Send EAPOL pkt to dc:XX:XX:XX:53:70
[  259.548222] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
[  259.548858] woal_cfg80211_set_rekey_data return: gtk_rekey_offload is DISABLE
```

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

```
Internet Systems Consortium DHCP Client 4.4.3-P1
Copyright 2004-2022 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/wlan0/d4:53:83:c3:e4:26
Sending on   LPF/wlan0/d4:53:83:c3:e4:26
Sending on   Socket/fallback
DHCPREQUEST for 192.168.1.10 on wlan0 to 255.255.255.255 port 67
DHCPACK of 192.168.1.10 from 192.168.1.1
bound to 192.168.1.10 -- renewal in 1405 seconds.
```

1.10.26 M.2 BLUETOOTH

- root@arm:~# **hciattach /dev/ttyS8 any 115200 flow**

```
[  447.897177] Bluetooth: Core ver 2.22
```

```
[  447.897924] NET: Registered PF_BLUETOOTH protocol family
[  447.897942] Bluetooth: HCI device and connection manager initialized
[  447.897977] Bluetooth: HCI socket layer initialized
[  447.897988] Bluetooth: L2CAP socket layer initialized
[  447.898038] Bluetooth: SCO socket layer initialized
[  447.920896] Bluetooth: HCI UART driver ver 2.3
[  447.920929] Bluetooth: HCI UART protocol H4 registered
Device setup complete
[  447.921092] Bluetooth: HCI UART protocol LL registered
[  447.921141] Bluetooth: HCI UART protocol Three-wire (H5) registered
[  447.922183] Bluetooth: HCI UART protocol Broadcom registered
[  447.922241] Bluetooth: HCI UART protocol QCA registered
[  447.922286] Bluetooth: HCI UART protocol Marvell registered
[  448.433149] Bluetooth: MGMT ver 1.22
[  448.446176] 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
```

请在网上搜索 **bluetoothctl** 用法以获取更多信息。

Note:

🕮 **hciattach 操作之前必须加载 moal.ko，否则会报错：** **Bluetooth: hci0: Frame reassembly failed (-84)**.

1.10.27 M.2 4G/5G MODULE

已测试的设备：

MODEL	DESCRIPTION
QUECTEL EM05-CE	4G module
QUECTEL RM500Q-GL	5G module

安装 GSM 模块、天线、SIM 卡，然后给 ARM 板通电。

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

```
Bus 002 Device 004: ID 2c7c:0800 Quectel Wireless Solutions Co., Ltd. RM500Q-GL
Bus 002 Device 003: ID 0424:9e00 Microchip Technology, Inc. (formerly SMSC) LAN9500A/LAN9500Ai
Bus 002 Device 002: ID 1a40:0201 Terminus Technology Inc. FE 2.1 7-port Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

终止可能在后台运行的 **pppd** 程序：

- root@arm:~# **killall -q pppd && sleep 3**

- root@arm:~# **pppd call quectel-ppp &**

```
......
Serial connection established.
using channel 1
Using interface ppp0
Connect: ppp0 <--> /dev/ttyGSM03
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x6586363d> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x0 <asyncmap 0x0> <auth chap MD5> <magic 0xafb6e7ff> <pcomp> <accomp>]
sent [LCP ConfAck id=0x0 <asyncmap 0x0> <auth chap MD5> <magic 0xafb6e7ff> <pcomp> <accomp>]
rcvd [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x6586363d> <pcomp> <accomp>]
rcvd [LCP DiscReq id=0x1 magic=0xafb6e7ff]
rcvd [CHAP Challenge id=0x1 <5d8494ff9ffeb38c9d39b711e1dc3f38>, name = "UMTS_CHAP_SRVR"]
sent [CHAP Response id=0x1 <b8f94ab38da425eb339f548b11d591c9>, name = "$LTE_USERNAME"]
rcvd [CHAP Success id=0x1 ""]
CHAP authentication succeeded
```

```
CHAP authentication succeeded
sent [IPCP ConfReq id=0x1 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
sent [IPV6CP ConfReq id=0x1 <addr fe80::2052:fff9:f87b:4287>]
rcvd [IPCP ConfReq id=0x0]
sent [IPCP ConfNak id=0x0 <addr 0.0.0.0>]
rcvd [IPCP ConfNak id=0x1 <addr 10.23.13.247> <ms-dns1 120.196.165.7> <ms-dns2
221.179.38.7>]
sent [IPCP ConfReq id=0x2 <addr 10.23.13.247> <ms-dns1 120.196.165.7> <ms-dns2
221.179.38.7>]
rcvd [IPCP ConfReq id=0x1]
sent [IPCP ConfAck id=0x1]
rcvd [IPCP ConfAck id=0x2 <addr 10.23.13.247> <ms-dns1 120.196.165.7> <ms-dns2
221.179.38.7>]
Could not determine remote IP address: defaulting to 10.64.64.64
not replacing default route to eth2 [192.168.3.1]
local  IP address 10.23.13.247
remote IP address 10.64.64.64
primary  DNS address 120.196.165.7
secondary DNS address 221.179.38.7
```

Note:

🕮 如果 **pppd** 命令报告错误，请尝试再次运行。

配置默认网关：

- root@arm:~# **route del default; route add default ppp0**

配置 **resolv.conf**:

- root@arm:~# **cat /etc/ppp/resolv.conf > /etc/resolv.conf**

Note:

🕮 **resolv.conf** 非常重要，如果它不正确，带 URL 的 ping 命令会报错，类似：Temporary failure in name resolution。

连通性测试：

- root@arm:~# **ping -I ppp0 www.baidu.com**

```
PING www.a.shifen.com (14.215.177.38) from 10.32.232.200 ppp0: 56(84) bytes of data.
64 bytes from 14.215.177.38: icmp_seq=1 ttl=54 time=37.0 ms
64 bytes from 14.215.177.38: icmp_seq=2 ttl=54 time=43.5 ms
64 bytes from 14.215.177.38: icmp_seq=3 ttl=54 time=51.8 ms
64 bytes from 14.215.177.38: icmp_seq=4 ttl=54 time=41.4 ms
^C64 bytes from 14.215.177.38: icmp_seq=5 ttl=54 time=33.4 ms

--- www.a.shifen.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 20329ms
rtt min/avg/max/mdev = 33.408/41.456/51.856/6.272 ms
```

禁用 GSM

它通常被称为“飞行模式”，禁用无线传输。

- root@arm:~# **echo 0 > /sys/class/leds/lte_on/brightness**

使能 GSM

- root@arm:~# **echo 1 > /sys/class/leds/lte_on/brightness**

GSM 复位:

root@arm:~# **echo 0 > /sys/class/leds/lte_reset/brightness; sleep 3; echo 1 > /sys/class/leds/lte_reset/brightness**

1.10.28 MIPI-CSI CAMERA

已测试的设备：

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

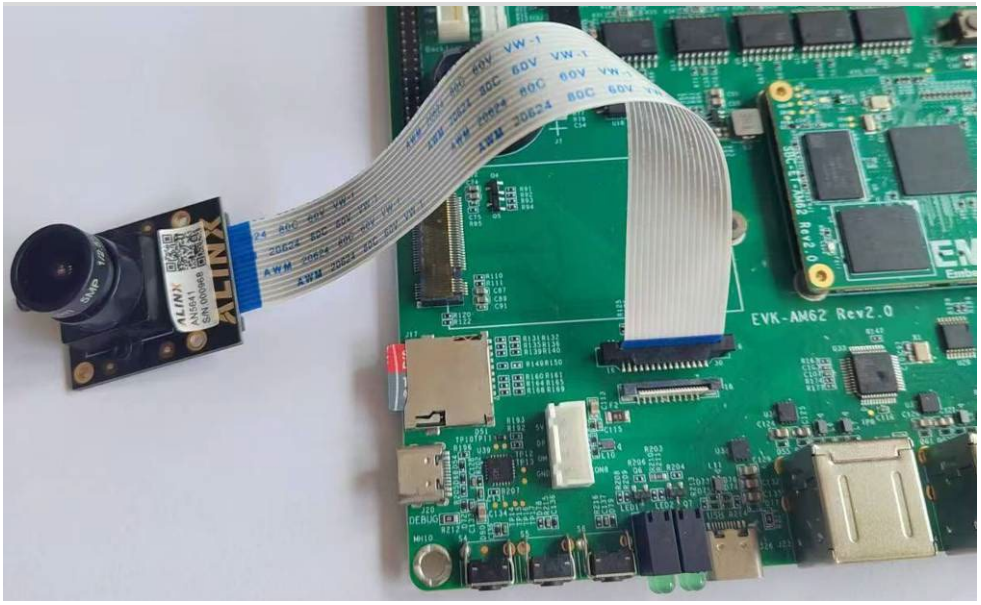


Figure 1-5 AN5641

- root@arm:~# **media-ctl -p**

```
.....
- entity 13: ov5640 0-003c (1 pad, 1 link, 0 route)
                type V4L2 subdev subtype Sensor flags 0
                device node name /dev/v4l-subdev2
        pad0: Source
                [stream:0 fmt:UYVY8_1X16/640x480@1/30 field:none colorspace:srgb
xfer:srgb ycbcr:601 quantization:full-range
                 crop.bounds:(0,0)/2624x1964
                 crop:(16,14)/2592x1944]
                -> "cdns_csi2rx.30101000.csi-bridge":0 [ENABLED,IMMUTABLE]
.....
```

相机设备节点是**/dev/video0**.

相机测试：

- root@arm:~# **media-ctl -V '"30102000.ticsi2rx":0/0 [fmt:UYVY8_1X16/640x480 field:none]'**
- root@arm:~# **media-ctl -V '"cdns_csi2rx.30101000.csi-bridge":0/0 [fmt:UYVY8_1X16/640x480 field:none]'**
- root@arm:~# **media-ctl --set-v4l2 '"ov5640 0-003c':0[fmt:UYVY8_1X16/640x480 fi**

eld:none]"

- root@arm:~# **fswebcam --no-banner -p UYVY -r 640x480 -S 3 image.jpg && we ston-image image.jpg**

```
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
--- Capturing frame...
Skipping 3 frames...
Capturing 1 frames...
Captured 4 frames in 0.10 seconds. (40 fps)
--- Processing captured image...
Disabling banner.
Writing JPEG image to 'image.jpg'.
could not load cursor 'dnd-move'
could not load cursor 'dnd-copy'
could not load cursor 'dnd-none'
```

现在我们可以看到摄像机捕获的实时图像流显示在屏幕上。

1.10.29 WAYLAND GPU

- root@arm:~# **glmark2-es2-wayland --run-foreve**

```
libEGL warning: egl: failed to create dri2 screen
=======================================================
    glmark2 2021.12
=======================================================
    OpenGL Information
    GL_VENDOR:      Mesa/X.org
    GL_RENDERER:    softpipe
    GL_VERSION:     OpenGL ES 3.1 Mesa 22.3.5 (git-54fd9d7dea)
=======================================================
[build] use-vbo=false: FPS: 1 FrameTime: 1000.000 ms
[build] use-vbo=true: FPS: 1 FrameTime: 1000.000 ms
......
```

Figure 1-6 glmark2-es2-wayland

1.10.30 QT6 GPU

- root@arm:~# **systemctl stop weston**
- root@arm:~# **export QT_QPA_EGLFS_KMS_CONFIG=/etc/kms.config**
- root@arm:~# **/usr/lib/aarch64-linux-gnu/qt6/examples/qt3d/simple-cpp/simple-cpp -platform eglfs**

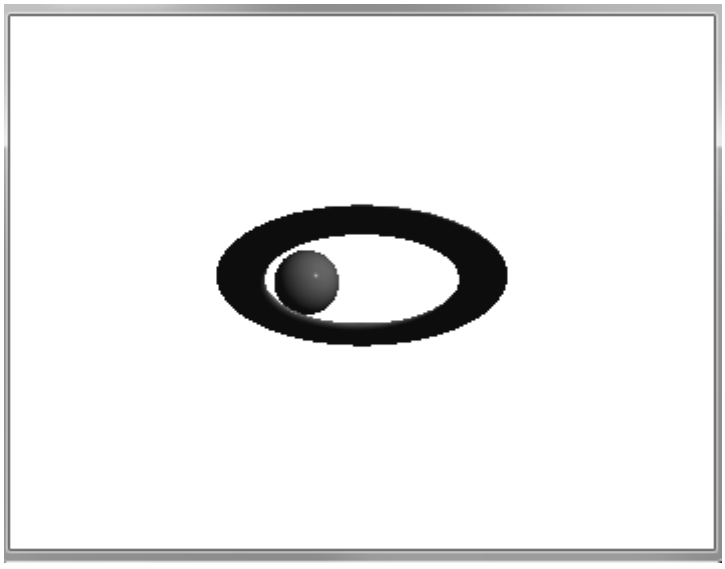


Figure 1-7 Qt3D simple-cpp Example

Note：

- weston 桌面不支持 Qt3D。请使用 **eglfs** 代替。
- **/usr/lib/aarch64-linux-gnu/qt6/examples/qt3d** 下还有一些其他示例。并非所有示例都可以运行，应使用 apt 命令安装相关插件。