让 openEuler 操作系统支持更多嵌入式设备

马亮

目录

作者介绍

我是谁

- 马亮

  中国科学院软件研究所智能软件研究中心内核工程师

- 中国科学院软件研究所智能软件研究中心

  中科院软件所（ISCAS）智能软件研究中心（ISRC）的使命是以智能驱动软件发展，以软件支撑智能创新。面向智能计算发展趋势，瞄准高性能、高安全、低功耗、低延时等需求，研究智能基础理论与模型、软件新结构与编译构造方法、内核和运行时环境等，打造适配通用处理器、智能芯片和开放指令集的操作系统和编译工具链，建设开源软件可靠供应链和安全漏洞平台，研发智能无人系统仿真测试环境，支撑智能计算生态和重要行业应用

openEuler 简介

openEuler是什么

- openEuler 是什么

  openEuler 是一个开源、免费的 Linux 发行版平台，通过开放的社区形式与全球的开发者共同构建一个开放、多元和架构包容的软件生态体系。同时，openEuler 也是一个创新的平台，鼓励任何人在该平台上提出新想法、开拓新思路、实践新方案

openEuler

openEuler 已适配的嵌入式硬件

开源嵌入式硬件开发板

目前我们已支持的开源嵌入式开发板有以下 3 款：

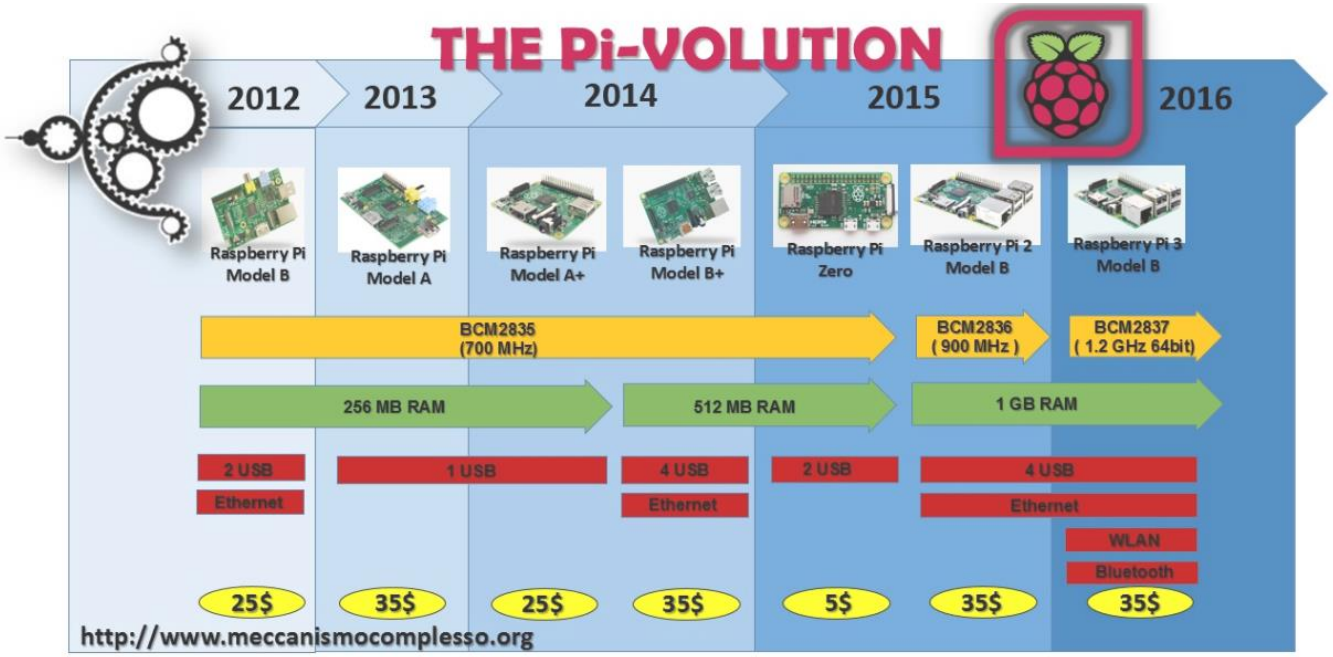
- 树莓派
- Firefly-RK3399
- OrangePi Zero2 (香橙派 Zero2)

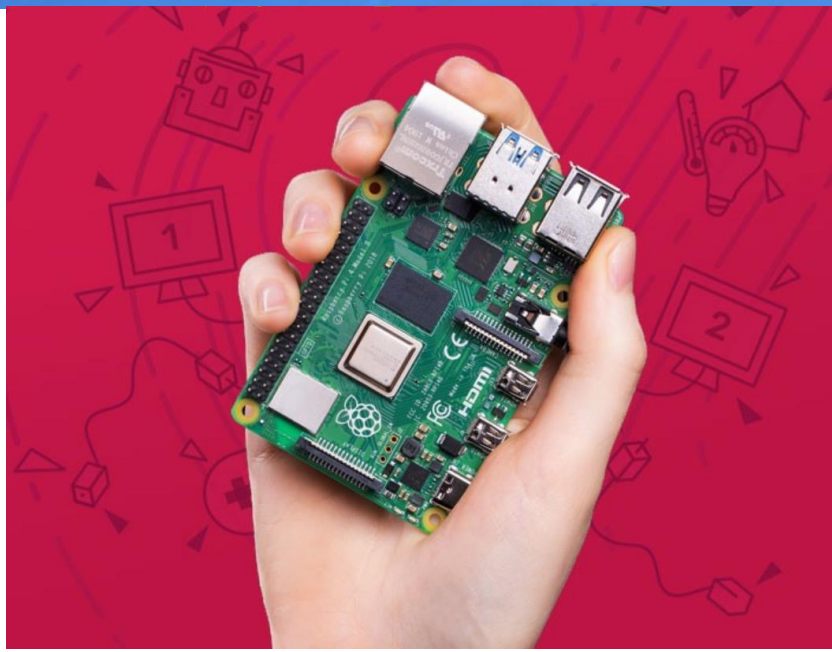
下面我将逐个介绍每个板子的硬件特性，openEuler 适配进展、openEuler 源码地址

RaspberryPi 介绍

发展历程

- 英国“Raspberry Pi 基金会”开发，创始人为 Eben Upton
- 基于 ARM 的微型电脑主板
- 目的：低价硬件及自由软件促进学校的基本计算机科学教育



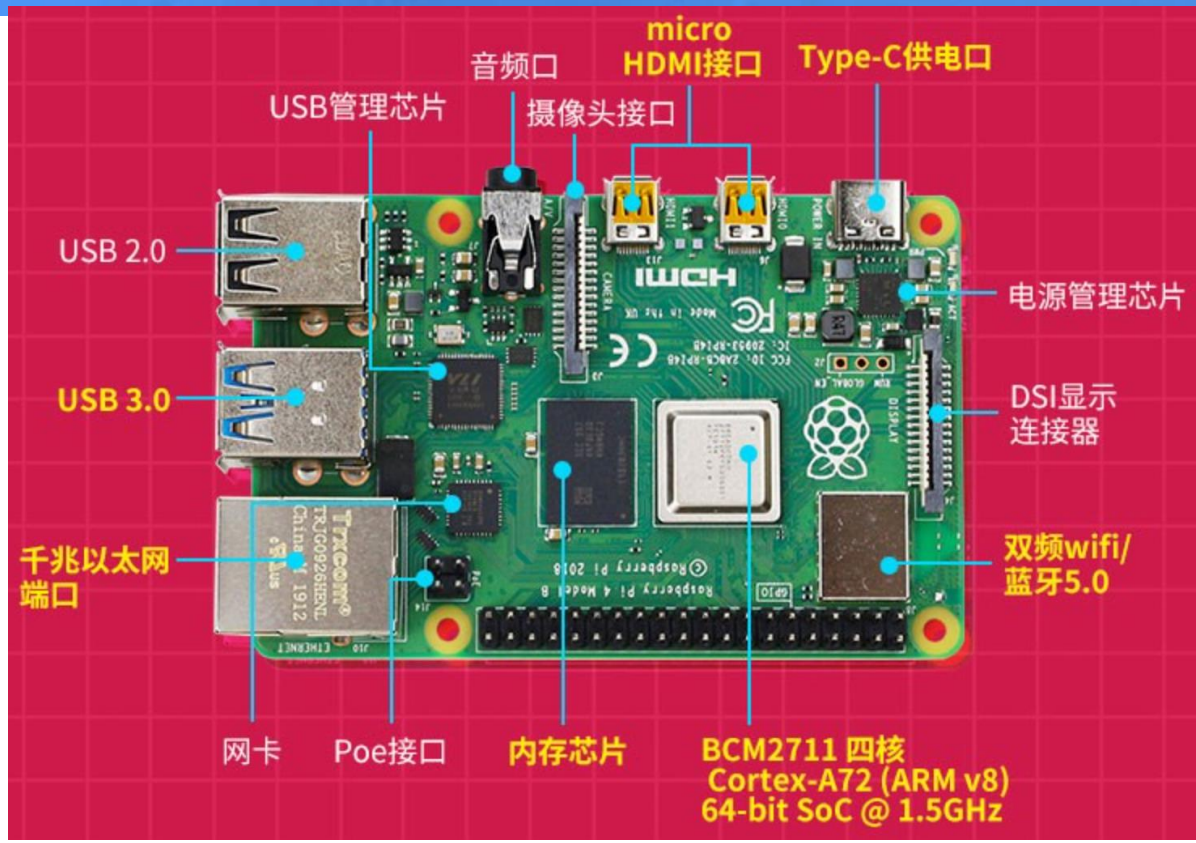


2017.03.01，树莓派 Zero W（$10）
2018.03.04，树莓派 3B+（$35）
2018.11.15，树莓派 3A+（$25）
2019.06.24，树莓派 4B（$35）
2020.11.02，树莓派 400（$70）
2021.01.21，树莓派 Pico（$4）

RaspberryPi 介绍

特点

- 硬件开源
- 外设丰富（显示器、摄像头、传感器、开放的接口）
- 操作系统（较强的视频编解码能力，板载网络等功能）
  - Raspbian、Ubuntu、OSMC（影音系统）
  - Windows 10 IoT
  - RISC OS
  - CentOS
  - openEuler



openEuler 移植到树莓派

最新版本

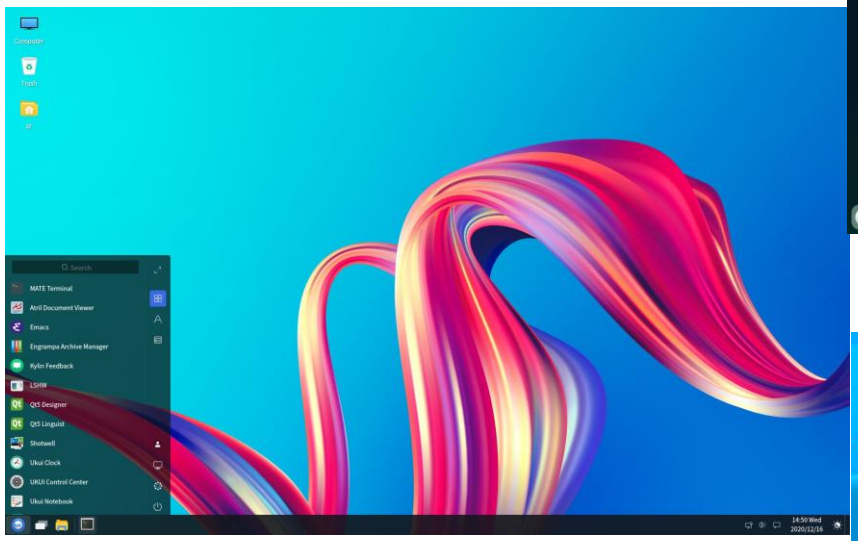
- 版本地址 https://gitee.com/openeuler/raspberrypi/blob/master/README.md

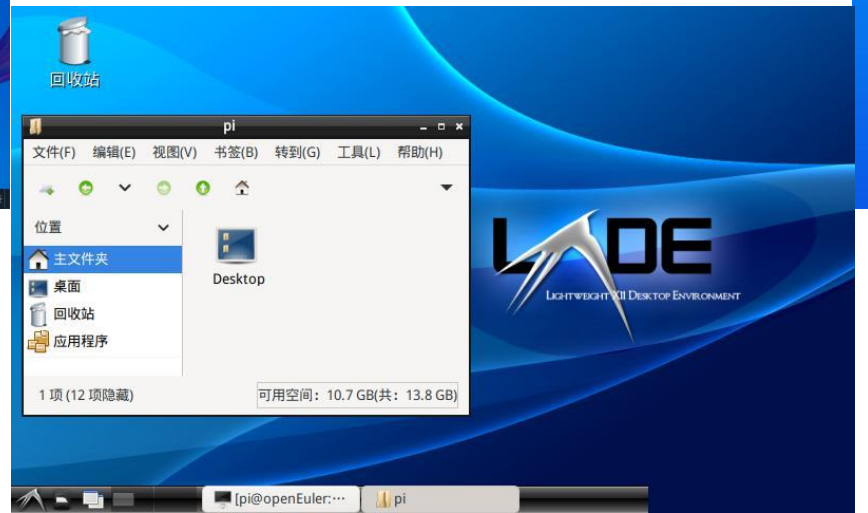
镜像版本	系统用户（密码）	更新日志	发布时间	大小	内核版本	构建文件系统的源仓库
openEuler 20.09 内测版	• root（openeuler） • pi（raspberry）	更新日志	2021/04/12	236 MiB	4.19.140-2104.1.0.0010	openEuler 20.09 每日构建源仓库
openEuler 20.09 内测版（包含 Xfce 桌面环境）	• root（openeuler） • pi（raspberry）	更新日志	2021/01/19	903 MiB	4.19.138-2008.1.0.0001	openEuler 20.09 源仓库
openEuler 20.09	root（openeuler）	-	2020/09/30	259 MiB	4.19.138-2008.1.0.0001	openEuler 20.09 源仓库
openEuler 20.03 LTS SP1 内测版	• root（openeuler） • pi（raspberry）	更新日志	2021/04/12	236 MiB	4.19.90-2104.1.0.0017	openEuler 20.03 LTS SP1 每日构建源仓库
openEuler 20.03 LTS SP1 内测版（UKUI 桌面、中文输入法）	• root（openeuler） • pi（raspberry）	更新日志	2021/04/12	1.1 GiB	4.19.90-2104.1.0.0017	openEuler 20.03 LTS SP1 每日构建源仓库
openEuler 20.03 LTS SP1 内测版（DDE 桌面、中文输入法）	• root（openeuler） • pi（raspberry）	更新日志	2021/04/12	1.1 GiB	4.19.90-2104.1.0.0017	openEuler 20.03 LTS SP1 每日构建源仓库

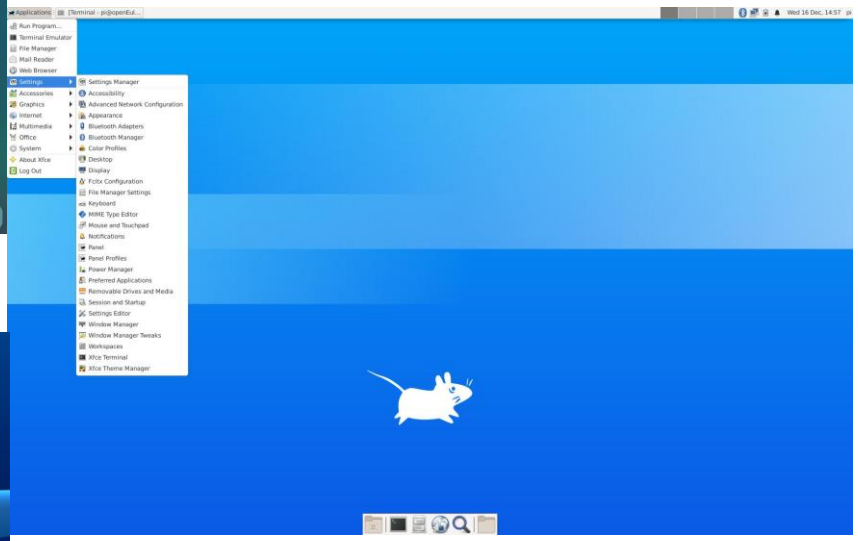
openEuler 移植到树莓派

桌面

- UKUI、DDE、Xfce、LXDE







Firefly-RK3399 介绍

RK3399 特点

- 高性能

  双核 Cortex-A72 和四核 Cortex-A53 和单独的 NEON 协处理器以及 Mali T860 GPU 集成在一起

- 嵌入式 3D GPU

  Mali T860 使 RK3399 完全兼容 OpenGL ES1.1/2.0/3.0/3.1 、OpenCL 和 DirectX 11.1

- 开源程度高

  可以使用完全开源的代码实现启动

- 应用广泛

  以下开发板都基于 RK3399：

  Firefly: Firefly-RK3399、ROC-RK3399-PC

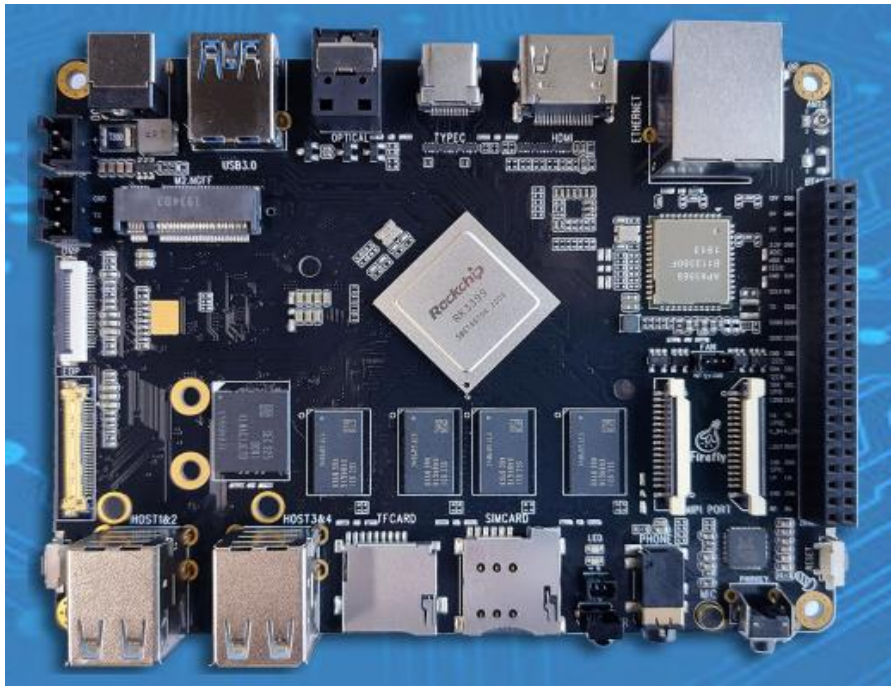
  Radxa: Rock Pi 4

  FriendlyARM: Nano PC T4、NanoPi NEO4、NanoPi M4

  Khadas: Edge-RK3399

  Lenovo: Leez P710

  Orange Pi: OrangePi-RK3399



Firefly-RK3399 介绍

适配的进展

- 已经适配完成部分

  openEuler 20.03 LTS 版本支持从 EMMC 启动， 板载 Wifi、蓝牙、以太网等设备正常

- 待完成工作

  系统只能通过 Firefly-SDK 构建，仅支持 Firefly-RK3399 一个型号

  仅支持从板载 EMMC 进行启动，不支持使用 MicroSD 卡启动 openEuler 镜像

  使用 Android Boot 启动方式，使用者更新内核以及编辑启动参数较为困难

- 项目仓库地址

  https://gitee.com/openeuler/raspberrypi

OrangePi Zero2 介绍

OrangePi Zero2 特点

- 性价比高

  Cortex-A53 四核 64 位 1.5GHZ CPU,集成了Mali-T720 GPU， 性能与树莓派相当，价格相较树莓派有较大优势

- 开源程度高

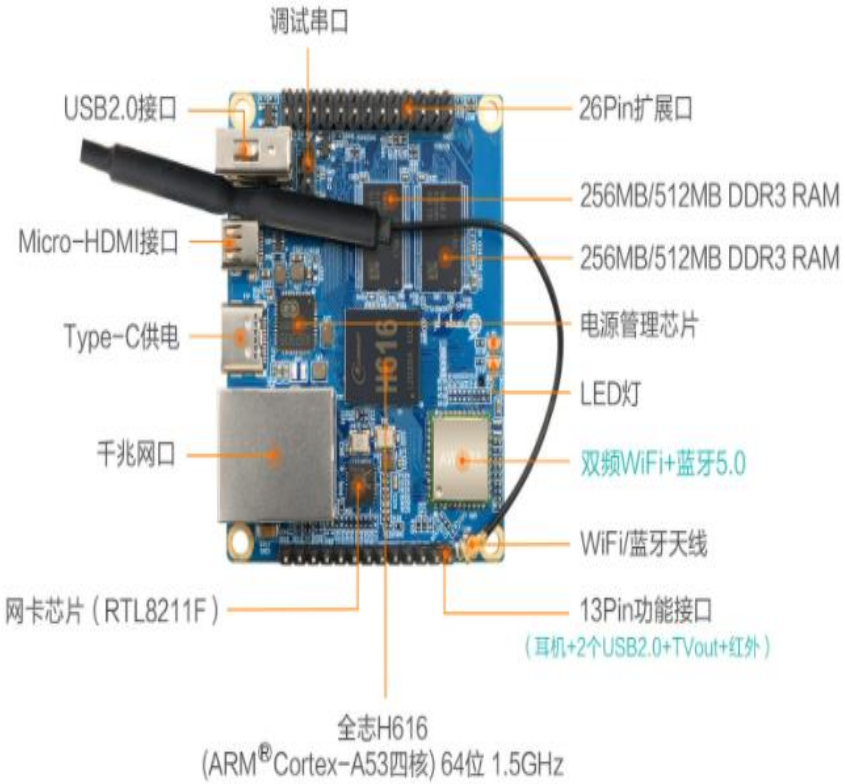
  SDk 中 u-boot 、内核全开源

- 外观小巧、功耗低

  Orange Pi Zero2大小仅有55mm*55mm，单板采用 USB Type-C 供电

- 不足

  对 Linux 内核支持的版本较低，只支持 4.9 内核。高版本内核移植到板子上的难度较高



OrangePi Zero2 介绍

OrangePi Zero2 适配进展

- 已完成部分

  整版从 4.9 内核升级到 openEuler 4.19 内核，板载主要外设驱动移植基本完成。

- 待完成工作

  RTC 、ALSA 音频等驱动还未完成

- 项目地址

  https://gitee.com/openeuler/allwinner-kernel

内核移植

步骤

- 配置内核

  包括 bootargs 配置、 arch\arm64\configs\config 中默认配置、dts 配置
- 移植 Console 驱动

  包括 uart 驱动、pinctrl驱动、clk 驱动、irq 驱动
- 移植 mmc 驱动

  包括所依赖的 sdio 驱动、regulator 驱动、gpio 驱动
- 挂载 Rootfs

  包括 bootargs 中指定挂载的 rootfs 类型、Config 中需要配置内核支持该 rootfs 类型

内核移植

设备驱动

- MMC 驱动

  现在市面上大多数开发板都是从 MicroSD 卡(或者 eMMC)启动，下面我以 OrangePi Zero2 的 4.9.170 内核为例，详细介绍 Linux 内核中 MMC 驱动的框架，及系统启动时MMC 驱动的加载流程

内核移植-MMC 驱动

SD/SDIO/MMC/EMMC概念

- MMC

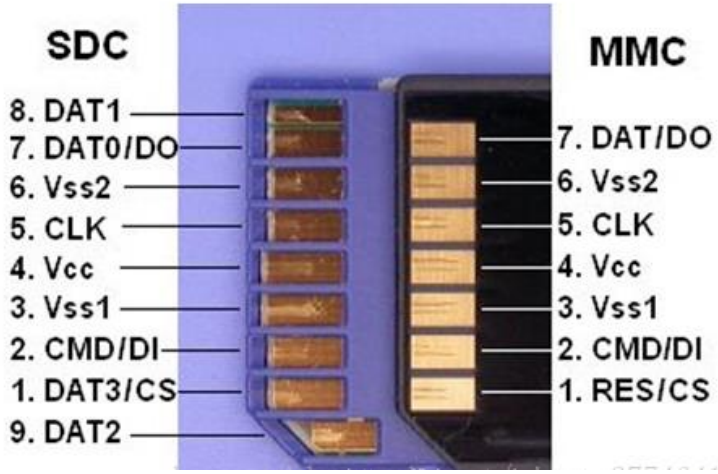
  MMC全称MultiMedia Card，由西门子公司和SanDisk公司1997年推出的多媒体记忆卡标准。MMC卡尺寸为32mm x24mm x 1.4mm，它将存贮单元和控制器一同做到了卡上，智能的控制器使得MMC保证兼容性和灵活性。MMC卡具有MMC和SPI两种工作模式，MMC模式是默认工作模式，具有MMC的全部特性。而SPI模式则是MMC协议的一个子集，主要用于低速系统

内核移植-MMC 驱动

SD/SDIO/MMC/EMMC概念

- SD

  SD卡全称Secure DigitalMemory Card，由松下、东芝和SanDisk公司于1999年8月共同开发的新一代记忆卡标准，已完全兼容MMC标准。SD卡比MMC卡多了一个进行数据著作权保护功能防止数据被他人复制，读写速度比MMC卡快4倍



内核移植-MMC 驱动

SD/SDIO/MMC/EMMC概念

- SDIO

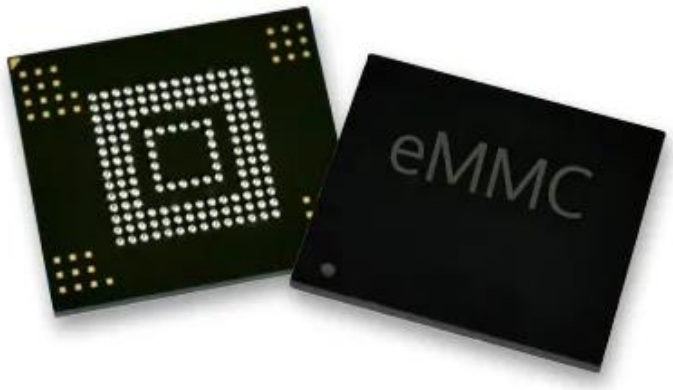
  SDIO全称Secure DigitalInput and Output Card，SDIO是在SD标准上定义了一种外设接口，它使用SD的I/O接口来连接外围设备，并通过SD上的I/O数据接口与这些外围设备传输数据。强调的是接口（IO，Input/Output），不再关注另一端的具体形态（可以是WIFI设备、Bluetooth设备、GPS等等）

内核移植-MMC 驱动

SD/SDIO/MMC/EMMC概念

- eMMC

  eMMC (Embedded Multi Media Card) 为MMC协会所订立的、主要是针对手机或平板电脑等产品的内嵌式存储器标准规格。eMMC的一个明显优势是在封装中集成了一个控制器，它提供标准接口并管理闪存，一般是BGA封装，焊接在PCB上



内核移植-MMC 驱动

MMC 驱动代码目录结构

- card 目录
  存放闪存卡(块设备)的相关驱动
- core 目录
  存放MMC的核心层代码
- host 目录
  存放不同厂商控制器的驱动

/linux/v4.9.170/source/drivers/mmc

/ drivers / mmc

Parent directory	
card	
core	
host	
Kconfig	625 bytes
Makefile	194 bytes

/linux/v4.19.184/source/drivers/mmc

/ drivers / mmc

Parent directory	
core	
host	
Kconfig	400 bytes
Makefile	119 bytes

内核 4.9 版本的目录结构是上述 3 个， 4.19 版本后将 card 目录与 core 目录合并

内核移植-MMC 驱动

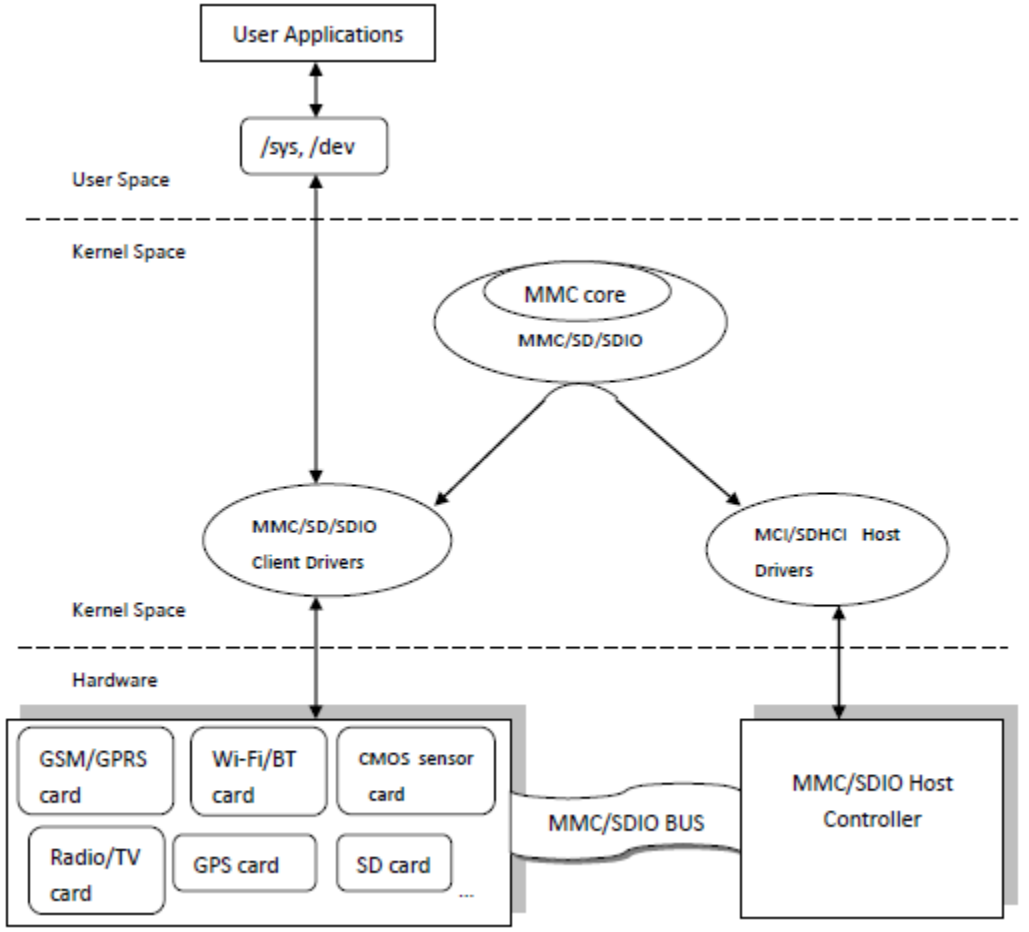
MMC 驱动分层

- MMC核心层

  完成不同协议和规范的实现，为host层和设备驱动层提供接口函数。MMC核心层由三个部分组成：MMC，SD和SDIO，分别为三类设备驱动提供接口函数
- Host 驱动层

  不同厂商的SDHC、MMC控制器的驱动，MMC 驱动移植的主要工作就在这一层
- Client 驱动层

  针对不同客户端的设备驱动程序。如SD卡、T-flash卡、SDIO接口的GPS和wi-fi等设备驱动



内核移植-MMC 驱动

MMC 驱动加载过程

- 设备初始化

  mmc驱动注册完成后，系统会扫描 sdio 总线，读取 dts 配置。

  将 dts 中设备的 device id 与 驱动的 id 进行匹配，匹配成功则执行驱动的探测函数 probe

```
static struct platform_driver sunxi_mmc_driver = {
    .driver = {
            .name = "sunxi-mmc",
            .of_match_table = of_match_ptr(sunxi_mmc_of_match),
            .pm = sunxi_mmc_pm_ops,
            },
    .probe = sunxi_mmc_probe,
    .remove = sunxi_mmc_remove,
    .shutdown = sunxi_shutdown_mmc,
};

module_platform_driver(sunxi_mmc_driver);

MODULE_DESCRIPTION("Allwinner's SD/MMC Card Controller Driver");
MODULE_LICENSE("GPL v2");
MODULE_AUTHOR("David Lanzend\ufffdrfer <david.lanzendoerfer@o2s.ch>");
MODULE_ALIAS("platform:sunxi-mmc");
```

kernel\drivers\mmc\host\sunxi-mmc.c

内核移植-MMC 驱动

MMC 驱动加载过程

- 设备初始化

  Host 驱动中的 match id 与 设备树 dts 中定义的 id

```
static const struct of_device_id sunxi_mmc_of_match[] = {
    {.compatible = "allwinner,sun4i-a10-mmc",},
    {.compatible = "allwinner,sun5i-a13-mmc",},
    {.compatible = "allwinner,sun8iw10p1-sdmmc3",},
    {.compatible = "allwinner,sun8iw10p1-sdmmc1",},
    {.compatible = "allwinner,sun8iw10p1-sdmmc0",},
    {.compatible = "allwinner,sun50i-sdmmc2",},
    {.compatible = "allwinner,sun50i-sdmmc1",},
    {.compatible = "allwinner,sun50i-sdmmc0",},
    {.compatible = "allwinner,sunxi-mmc-v4p1x",},
    {.compatible = "allwinner,sunxi-mmc-v4p10x",},
    {.compatible = "allwinner,sunxi-mmc-v4p00x",},
    {.compatible = "allwinner,sunxi-mmc-v4p5x",},
    {.compatible = "allwinner,sunxi-mmc-v4p6x",},
    {.compatible = "allwinner,sunxi-mmc-v5p3x",},
    { /* sentinel */ }
};

MODULE_DEVICE_TABLE(of, sunxi_mmc_of_match);
```

kernel\drivers\mmc\host\sunxi-mmc.c

```
sdc0: sdmmc@04020000 {
    compatible = "allwinner,sunxi-mmc-v4p1x";
    device_type = "sdc0";
    reg = <0x0 0x04020000 0x0 0x1000>;
    interrupts = <GIC_SPI 35 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&clk_hosc>,
        <&clk_pll_periph1x2>,
        <&clk_sdmmc0_mod>,
        <&clk_sdmmc0_bus>,
        <&clk_sdmmc0_rst>;
    clock-names = "osc24m","pll_periph","mmc","ahb","rst";
    pinctrl-names = "default","sleep","uart_jtag";
    pinctrl-0 = <&sdc0_pins_a>;
    pinctrl-1 = <&sdc0_pins_b>;
    pinctrl-2 = <&sdc0_pins_c>;
    max-frequency = <50000000>;
    bus-width = <4>;
    /*non-removable;*/
    /*broken-cd;*/
    /*cd-inverted*/
    cd-gpios = <&pio PF 6 6 1 3 0xffffffff>;
    /* vmmc-supply = <&reg_3p3v>;*/
    /* vqmc-supply = <&reg_3p3v>;*/
    /* vdmc-supply = <&reg_3p3v>;*/
    /*vmmc = "vcc-card";*/
```

kernel\arch\arm64\boot\dts\sunxi\sun50iw9p1.dtsi

mmc_rescan函数就是用来扫描识别eMMC、SD、SDIO设备的，设备识别时会一个较低的频率去与设备交互，当设备识别成功后，可以将工作频率提

内核移植-MMC 驱动

MMC 驱动加载过程

- 设备识别

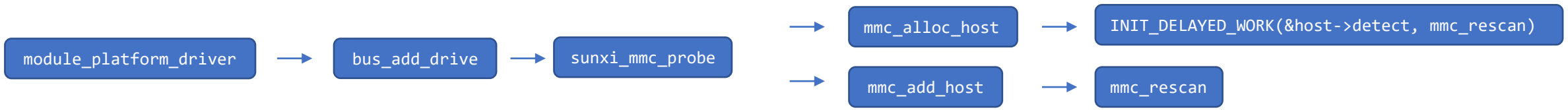
  驱动在执行 probe 函数的过程中，通过调用 mmc_add_host 函数开启一个延迟工作队列，任务是调用 mmc_rescan 函数进行设备识别。mmc_rescan函数就扫描识别 eMMC、SD、SDIO设备，设备识别时会以一个较低的频率去与设备交互，当设备识别成功后，将工作频率提高。如果低频扫描过程中没有识别到设备，就对设备下电

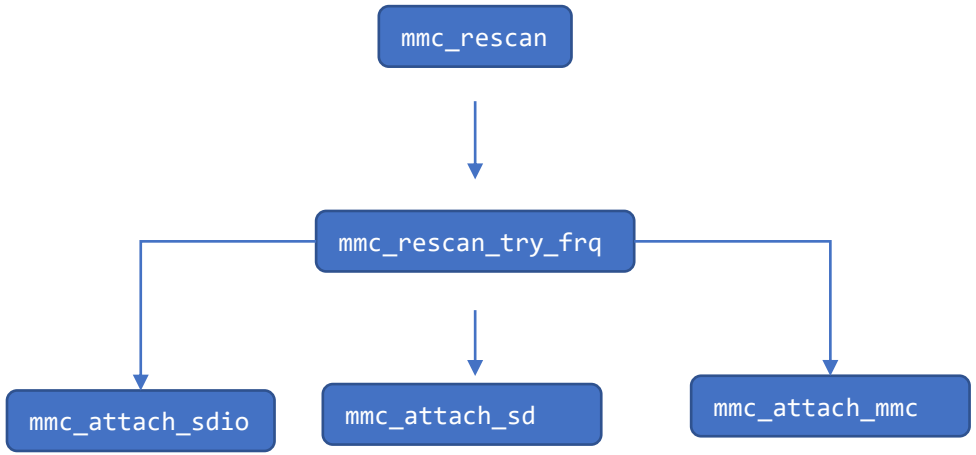
```
static int sunxi_mmc_probe(struct platform_device *pdev)
{…
#if (defined CONFIG_ARCH_SUN50IW9) || (defined CONFIG_ARCH_SUN50IW10)…
#else…
#endif…
#ifndef CONFIG_REGULATOR…
#endif

    mmc_of_parse(mmc);
    sunxi_mmc_extra_of_parse(mmc);
    if (mmc->sunxi_caps3 & MMC_SUNXI_CAP3_DAT3_DET)
        host->dat3_imask = SDXC_CARD_INSERT | SDXC_CARD_REMOVE;
    if (mmc->sunxi_caps3 & MMC_SUNXI_CAP3_CD_USED_24M) {
        ctx = (struct mmc_gpio *)mmc->slot.handler_priv;
        if (ctx  && ctx->cd_gpio) {
            ret = gpiod_set_debounce(ctx->cd_gpio, 1);
            if (ret < 0) {
                dev_info(&pdev->dev, "set cd-gpios as 24M fail\n");
            }
        }
    }

    dev_dbg(&pdev->dev, "base:0x%p irq:%u\n", host->reg_base, host->irq);
    platform_set_drvdata(pdev, mmc);

    ret = mmc_add_host(mmc);
    if (ret)
        goto error_free_dma;
```

kernel\drivers\mmc\host\sunxi-mmc.c

内核移植-MMC 驱动

MMC 驱动加载流程





内核调试技巧

在没有Console 输出的情况下调试驱动

- 问题

  内核移植初期，由于串口等外设驱动还没有移植过来，板子的 Console 没有打印输出。

  而移植 uart 、pinctrl、clk、irq 等 Console 依赖的驱动时，又需要有打印信息才能完成移植。

  这里介绍一种通过 u-boot 的内存 dump 命令，查看驱动调试信息的方法，可实现直接在内存中查看打印信息，完成 console 驱动的移植。下面以 OrangePi Zero2 为例介绍具体的操作过程

内核调试技巧

在没有Console 输出的情况下调试驱动

- 获得内核日志缓存的绝对地址

  首先在 System.map 文件中获得 __log_buf 数组的内存映射地址 0xffff000009512cc8，

  __log_buf 即为内核日志的缓存。然后获取内核代码段_text的

  起始地址 0xffff000008080000，并获取内核加载到内存中的入口地址，

  OragePi Zer2 的入口地址为 0x41000000 。 最后获得 __log_buf 数组

  在板子内存中的绝对地址:
  0xffff000009512cc8 - 0xffff000008080000 +0x41000000 = 0x42492cc8

```
ffff0000095108c0 b console_idx
ffff0000095108c8 b exclusive_console_stop_seq
ffff0000095108d0 b exclusive_console
ffff0000095108d8 b has_preferred.40993
ffff0000095108e0 b textbuf.40715
ffff000009510cc0 B oops_in_progress
ffff000009510cc4 b always_kmsg_dump
ffff000009510cc8 b ext_text.40912
ffff000009512cc8 b __log_buf
ffff000009612cc8 b read_lock.8365
ffff000009612cd0 b irq_kobj_base
ffff000009612cd8 b __key.22973
ffff000009612cd8 b allocated_irqs
ffff0000096130e8 B irq_default_affinity
ffff000009613168 b mask_lock.32244
ffff000009613170 b mask.32246
```

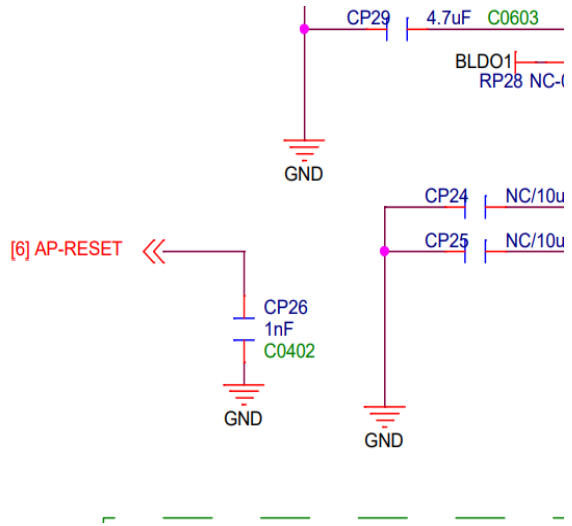
```
0000000000de0000 A __efistub_stext_offset
0000000000fd6b58 A __rela_offset
0000000001649000 A _kernel_size_le_lo32
ffff000008080000 t __efistub__text
ffff000008080000 t _head
ffff000008080000 T _text
ffff000008080040 t pe_header
ffff000008080044 t coff_header
ffff000008080058 t optional_header
ffff000008080070 t extra_header_fields
ffff0000080800f8 t section_table
ffff000008081000 T __exception_text_start
ffff000008081000 T _stext
ffff000008081000 T do_undefinstr
```

内核调试技巧

在没有Console 输出的情况下调试驱动

- 带电复位CPU

  之所以要带电复位 CPU, 是让板子上的 DDR 不掉电、不复位，这样 DDR 中就保存了上一次板子启动后的调试信息。CPU 复位重启后可在 u-boot 中执行相应的 dump 命令，查看调试信息。查看 OrangePi Zero2 板子电路图，找到 CPU 的 reset 引脚电路，手动拉低 AP-RESET 引脚，实现 CPU reset



内核调试技巧

在没有Console 输出的情况下调试驱动

- 查看内存中日志

  CPU reset 后在 u-boot 中停下，输入命令: md.w 42492cc8 1000 查看内存中日志。 42492cc8 为上面计算出的 __log_buf 的绝对地址，1000 是需要查看的内存大小，即 : 输出 DDR 中 0x42492cc8 地址后的 1000 字节

```
host_caps:0x3000003f
** Unrecognized filesystem type **
** Unrecognized filesystem type **
[04.296]boot_gui_init:finish
[04.299]bmp_name=/boot/boot.bmp
** Unrecognized filesystem type **
[04.307]sunxi bmp info error : unable to open logo file /boot/boot.bmp
[04.316]update dts
Net:   [04.319]No ethernet found.
Hit any key to stop autoboot:  0
orangepi#
orangepi#
orangepi#
orangepi#
orangepi#
orangepi#
orangepi# md.w 42492cc8   1000
42492cc8: 0000 0000 0000 0000 0048 0037 0000 c600    ........H.7.....
42492cd8: 6f42 746f 6e69 2067 694c 756e 2078 6e6f    Booting Linux on
42492ce8: 7020 7968 6973 6163 206c 5043 2055 7830     physical CPU 0x
42492cf8: 3030 3030 3030 3030 3030 5b20 7830 3134    0000000000 [0x41
42492d08: 6630 3064 3433 005d 0000 0000 0000 0000    0fd034].........
42492d18: 00bc 00a9 0000 a600 694c 756e 2078 6576    ........Linux ve
42492d28: 7372 6f69 206e 2e34 3931 392e 2b30 2820    rsion 4.19.90+ (
42492d38: 6f72 746f 6b40 2969 2820 6367 2063 6576    root@ki) (gcc ve
42492d48: 7372 6f69 206e 2e39 2e32 2031 3032 3931    rsion 9.2.1 2019
42492d58: 3031 3532 2820 4e47 2055 6f54 6c6f 6863    1025 (GNU Toolch
42492d68: 6961 206e 6f66 2072 6874 2065 2d41 7270    ain for the A-pr
42492d78: 666f 6c69 2065 7241 6863 7469 6365 7574    ofile Architectu
42492d88: 6572 3920 322e 322d 3130 2e39 3231 2820    re 9.2-2019.12 (
42492d98: 7261 2d6d 2e39 3031 2929 2029 3223 3335    arm-9.10))) #253
42492da8: 5320 504d 5420 6575 5320 7065 3220 2038     SMP Tue Sep 28
42492db8: 3931 333a 3a39 3330 4320 5453 3220 3230    19:39:03 CST 202
42492dc8: 0031 0000 0000 0000 0000 0000 0030 001f    1...........0...
42492dd8: 0000 c600 614d 6863 6e69 2065 6f6d 6564    ....Machine mode
42492de8: 3a6c 4f20 6172 676e 2065 6950 5a20 7265    l: Orange Pi Zer
42492df8: 206f 0032 0000 0000 0000 0000 0038 0025    o 2.........8.%.
42492e08: 0000 c600 6665 3a69 4720 7465 6974 676e    ....efi: Getting
42492e18: 4520 4946 7020 7261 6d61 7465 7265 2073     EFI parameters
42492e28: 7266 6d6f 4620 5444 003a 0000 0000 0000    from FDT:.......
42492e38: 0000 0000 0024 0014 0000 c600 6665 3a69    ....$.......efi:
42492e48: 5520 4645 2049 6f6e 2074 6f66 6e75 2e64     UEFI not found.
42492e58: 0000 0000 0000 0000 003c 002a 0000 c600    .........<.*.....
42492e68: 6d63 3a61 5220 7365 7265 6576 2064 3631    cma: Reserved 16
42492e78: 4d20 4269 6120 2074 7830 3030 3030 3030     MiB at 0x000000
42492e88: 3030 6637 3030 3030 3030 0000 0000 0000    007f000000......
42492e98: 0000 0000 0034 0021 0000 c600 554e 414d    ....4.!.....NUMA
42492ea8: 203a 6f4e 4e20 4d55 2041 6f63 666e 6769    : No NUMA config
42492eb8: 7275 7461 6f69 206e 6f66 6e75 0064 0000    uration found...
42492ec8: 0000 0000 0000 0000 0054 0042 0000 c600    .........T.B.....
42492ed8: 554e 414d 203a 6146 696b 676e 6120 6e20    NUMA: Faking a n
42492ee8: 646f 2065 7461 5b20 656d 206d 7830 3030    ode at [mem 0x00
```

内核调试技巧

获得函数的调用栈，快速阅读驱动源代码

- 主动输出 Oops 信息

  阅读驱动源码时，有时需要知道驱动的加载过程、函数的调用栈。可以在函数中增加 dump_stack 函数输出当前的栈信息。如： 在 mmc host 驱动的 probe 函数中增加 dump_stack(), 获得驱动注册到设备 match id 的函数调用过程

```c
static int sunxi_mmc_probe(struct platform_device *pdev)
{
    struct sunxi_mmc_host *host;
    struct mmc_host *mmc;
    struct mmc_gpio *ctx;
    int ret;
    dev_info(&pdev->dev, "%s\n", DRIVER_VERSION);

    mmc = mmc_alloc_host(sizeof(struct sunxi_mmc_host), &pdev->dev);
    if (!mmc) {
        dev_err(&pdev->dev, "mmc alloc host failed\n");
        return -ENOMEM;
    }

    dump_stack();

    host = mmc_priv(mmc);
    host->mmc = mmc;
    spin_lock_init(&host->lock);

    ret = sunxi_mmc_resource_request(host, pdev);
    if (ret)
        goto error_free_host;
```

内核调试技巧

获得函数的调用栈，快速阅读驱动源代码

- 输出 Oops 信息

  增加 dump_stack() 函数后，获得 sunxi_mmc_probe 函数的调用栈

```
[    3.378829] Hardware name: Orange Pi Zero 2 (DT)
[    3.378832] Call trace:
[    3.378846]  dump_backtrace+0x0/0x190
[    3.378852]  show_stack+0x24/0x30
[    3.395217]  dump_stack+0xa4/0xc4
[    3.395228]  sunxi_mmc_probe+0xb0/0x3e0
[    3.413455]  platform_drv_probe+0x58/0xa8
[    3.413459]  really_probe+0x1ac/0x38c
[    3.413466]  driver_probe_device.part.0+0xcc/0x10c
[    3.433155]  __driver_attach+0x16c/0x170
[    3.433161]  bus_for_each_dev+0x84/0xd8
[    3.433164]  driver_attach+0x30/0x40
[    3.433170]  bus_add_driver+0x16c/0x270
[    3.441797]  driver_register+0x64/0x110
[    3.441801]  __platform_driver_register+0x54/0x60
[    3.441808]  sunxi_mmc_driver_init+0x20/0x28
[    3.441816]  do_one_initcall+0x54/0x1e4
[    3.456220]  kernel_init_freeable+0x288/0x32c
[    3.456227]  kernel_init+0x18/0x110
[    3.456234]  ret_from_fork+0x10/0x18
```

RaspberryPi SIG

加入我们

- 树莓派
  - https://gitee.com/openeuler/raspberrypi
  - https://gitee.com/openeuler/raspberrypi-kernel
  - https://gitee.com/openeuler/raspberrypi-build
  - https://gitee.com/src-openeuler/raspberrypi-kernel
  - https://gitee.com/src-openeuler/raspberrypi-firmware
  - https://gitee.com/src-openeuler/raspberrypi-bluetooth
  - https://gitee.com/src-openeuler/raspberrypi-build
  - https://gitee.com/src-openeuler/raspi-config
  - https://gitee.com/src-openeuler/pigpio
  - https://gitee.com/src-openeuler/raspberrypi-userland
  - https://gitee.com/src-openeuler/raspberrypi-eeprom
- 瑞芯微
  - https://gitee.com/openeuler/rockchip
  - https://gitee.com/openeuler/rockchip-kernel
- 全志
  - https://gitee.com/openeuler/allwinner-kernel

RaspberryPi SIG

加入我们

- 目标

  降低 openEuler 使用门槛，致力于将 openEuler 移植到树莓派/瑞芯微/全志等开发板，以及后续维护、更新。
- Slack

  https://openeuler-raspberrypi.slack.com
- 公开会议

  北京时间，每个月第一个和第三个周二，17:00-17:30
- 邮件列表

  dev@openeuler.org

基本信息

谢 谢

欢迎交流合作

2021/10/24