# 在调试器下理解ARMv8

## ——生态系统：硬件差异和固件

张银奎 2021-12-26 北京 西山脚下

From Linus Torvalds <>
Date Thu, 17 Mar 2011 19:50:36 -0700
Subject Re: [GIT PULL] omap changes for v2.6.39 merge window share 292
On Thu, Mar 17, 2011 at 11:30 AM, Tony Lindgren <tony@atomide.com> wrote:
>
> Please pull omap changes for this merge window from:

Gaah. Guys, this whole ARM thing is a f*cking pain in the ass.

You need to stop stepping on each others toes. There is no way that your changes to those crazy clock-data files should constantly result in those annoying conflicts, just because different people in different ARM trees do some masturbatory renaming of some random device. Seriously.

That usb_musb_init() thing in arch/arm/mach-omap2/usb-musb.c also seems to be totally insane. I wonder what kind of insanity I'm missing just because I don't happen to see the merge conflicts, just because people were lucky enough to happen to not touch the same file within a few lines.

Somebody needs to get a grip in the ARM community. I do want to do these merges, just to see how screwed up things are, but guys, this is just ridiculous. The pure amount of crazy churn is annoying in itself, but when I then get these "independent" pull requests from four different people, and they touch the same files, that indicates that something is wrong.

And stop the crazy renaming already! Just leave it off. Don't rename boards and drivers "just because", at least not when there clearly are clashes. There's no point. I'm not even talking about the file renames (which happened and can also make it "fun" to try to resolve the conflicts when somebody else then makes _other_ changes), but about the stupid "change human-readable names in board files just to annoy whoever needs to merge the crap".

Somebody in the ARM community really needs to step up and tell people to stop dicking around.

(I'm replying to the omap pull request, because that's the one I did last, but I don't know who to "blame". I don't care. It really doesn't matter. I realize thar ARM vendors do crazy shit and haven't figured out this whole "platform" thing yet, but you guys need to push back on the people sending you crap).

Linus

"As an indication of the scale of this problem, each new kernel release sees about 70,000 new lines of ARM code, whereas there's roughly 5,000 lines of new x86 code added."

David Rusling CTO of the Linaro Linux-on-ARM联盟, 2011

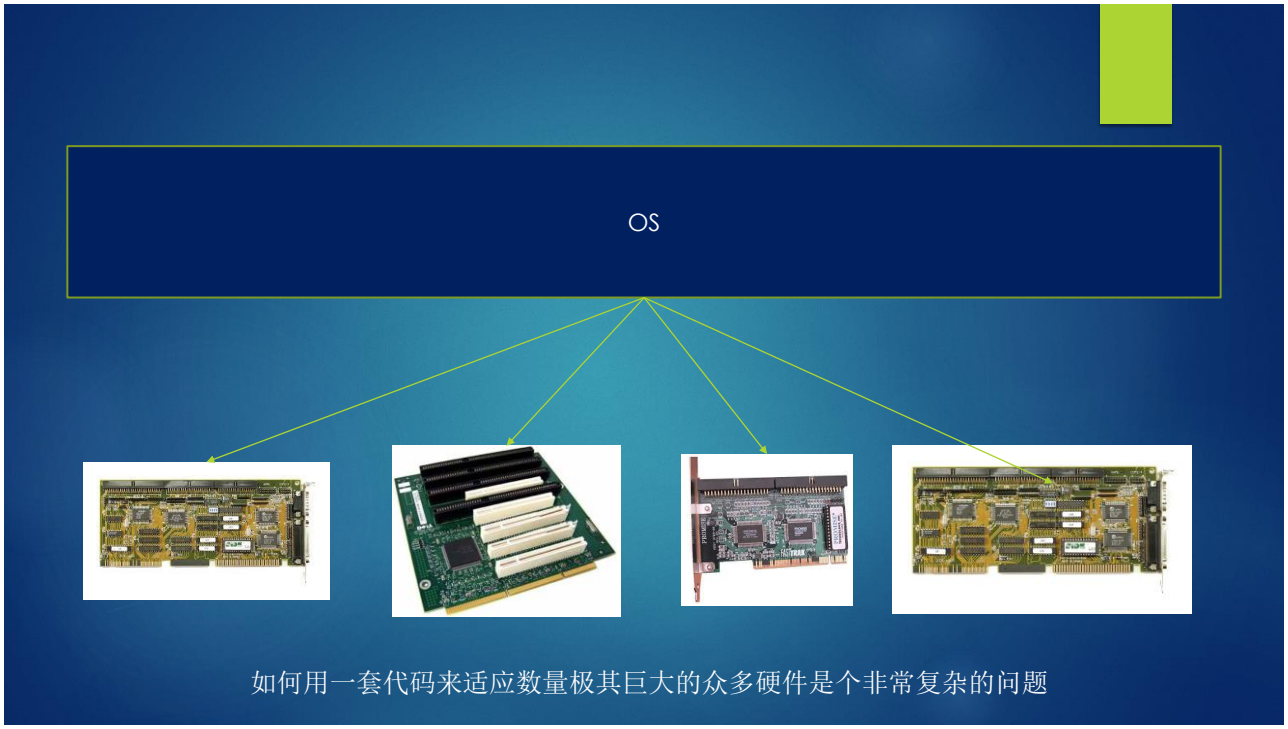
永恒问题 → 经典解法 → DTS → UEFI → TF/OP-TEE → ARM on ARM

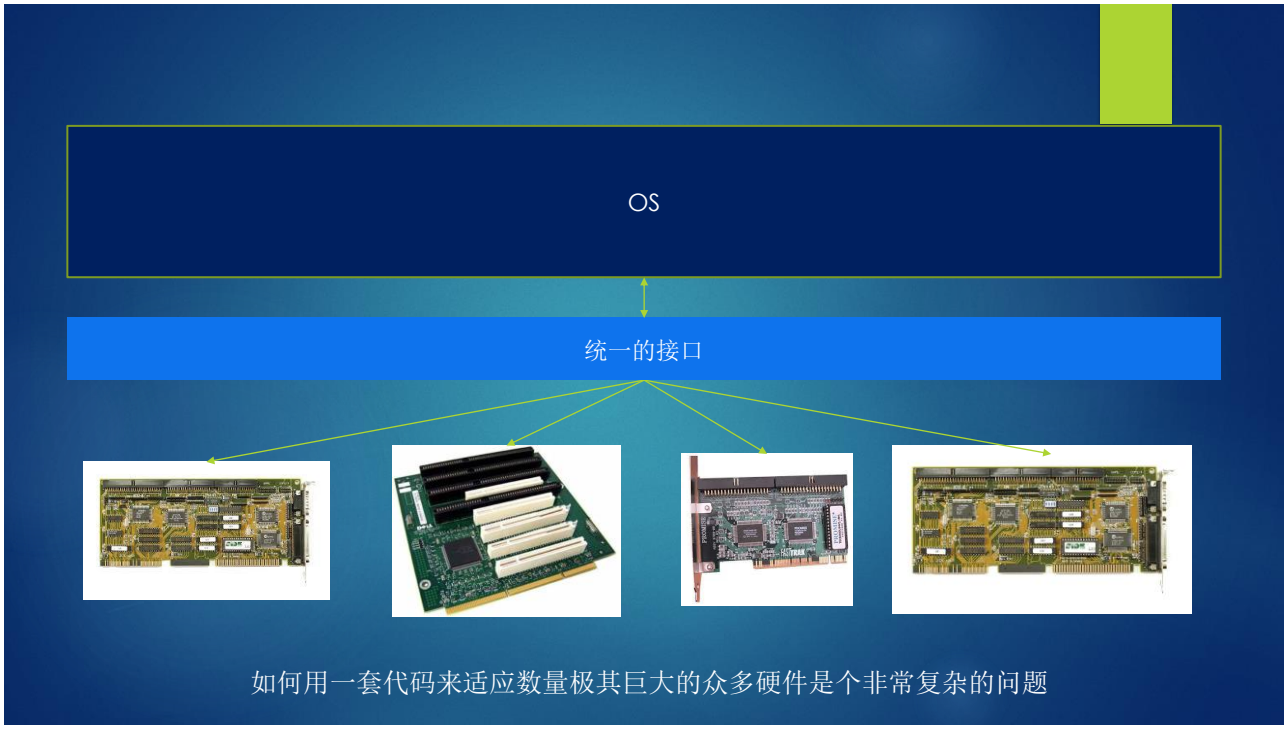
1975年，加里最先提出和实践了通过“基本输入输出系统”（BIOS）来隔离硬件差异性的想法。通过BIOS，CP/M操作系统可以在数百种硬件上工作。

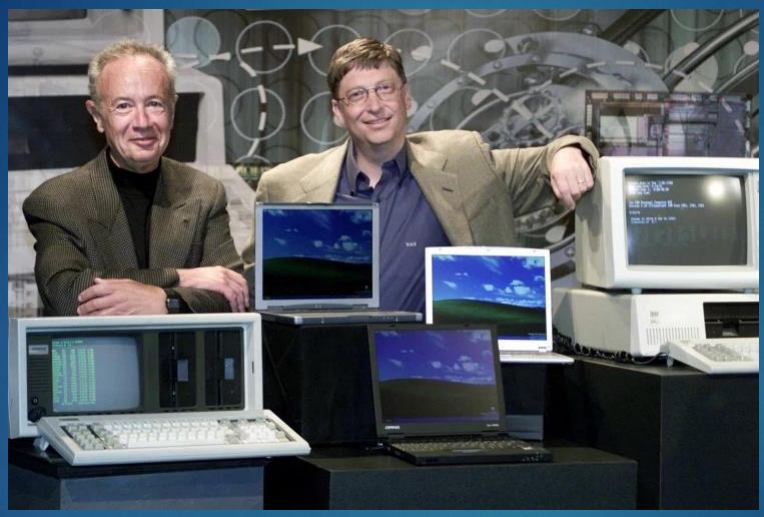
加里•基尔代尔（1942-1994）

Dan Fylstra of VisiCorp, Bill Gates of Microsoft, and Gary Kildall of Digital Research in 1984

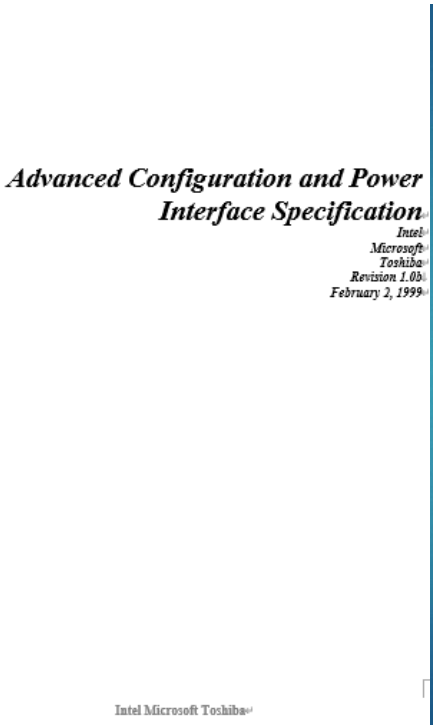
Bill Gates and Andy Grove at Microsoft's headquarters in Redmond in 1993.


OS
如何用一套代码来适应数量极其巨大的众多硬件是个非常复杂的问题


OS
统一的接口
如何用一套代码来适应数量极其巨大的众多硬件是个非常复杂的问题



2001

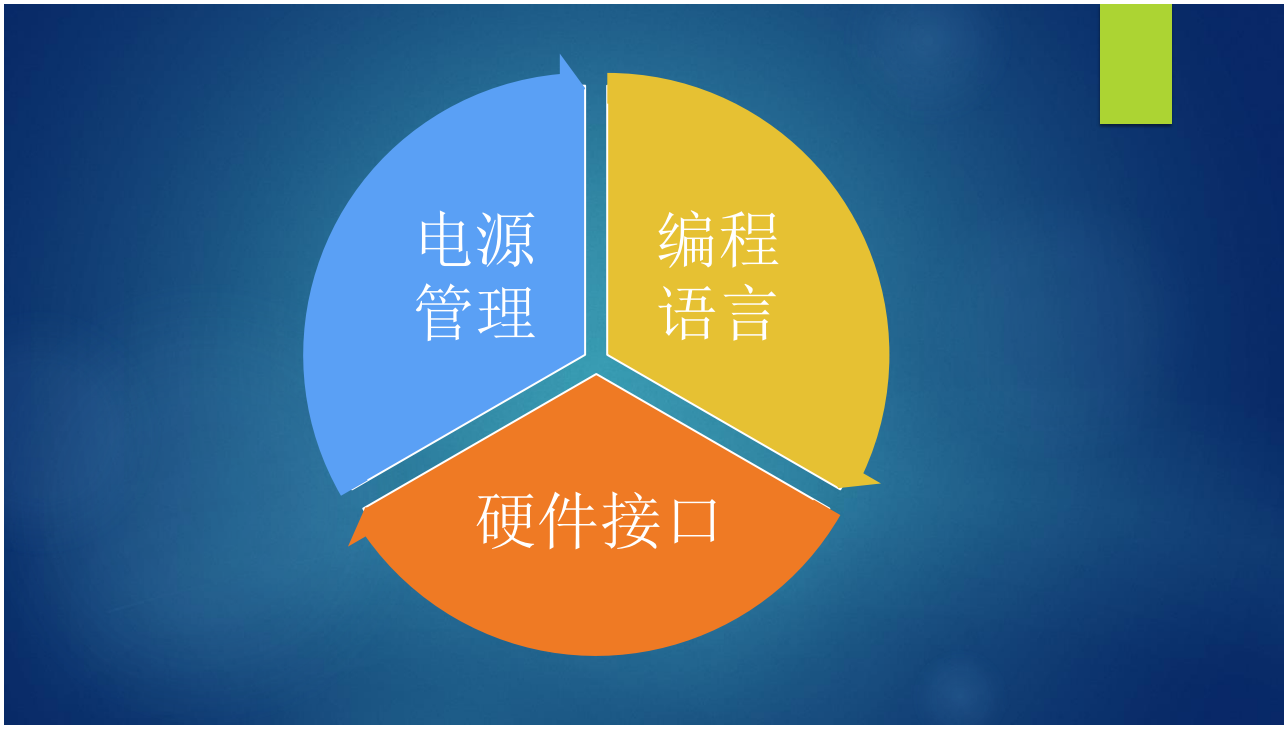


*Advanced Configuration and Power Interface Specification*

*Intel*
*Microsoft*
*Toshiba*
*Revision 1.0b*
*February 2, 1999*

Intel Microsoft Toshiba

1999年2/2
ACPI Spec 1.0b




dsdt.dsl - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
/*
 * Intel ACPI Component Architecture
 * AML/ASL+ Disassembler version 20171110 (32-bit version)
 * Copyright (c) 2000 - 2017 Intel Corporation
 *
 * Disassembling to symbolic ASL+ operators
 *
 * Disassembly of dsdt.dat, Wed Sep 23 21:47:12 2020
 *
 * Original Table Header:
 *     Signature        "DSDT"
 *     Length         0x0002198D (137613)
 *     Revision         0x02
 *     Checksum         0x4F
 *     OEM ID          "LENOVO"
 *     OEM Table ID     "SKL     "
 *     OEM Revision     0x00000000 (0)
 *     Compiler ID      "INTL"
 *     Compiler Version 0x20160527 (538314023)
 */
DefinitionBlock ("", "DSDT", 2, "LENOVO", "SKL     ", 0x00000000)
{
    External (_PR_.BGIA, UnknownObj)
    External (_PR_.BGMA, UnknownObj)
    External (_PR_.BGMS, UnknownObj)
    External (_PR_.CFGD, UnknownObj)
    External (_PR_.CLVL, UnknownObj)
```

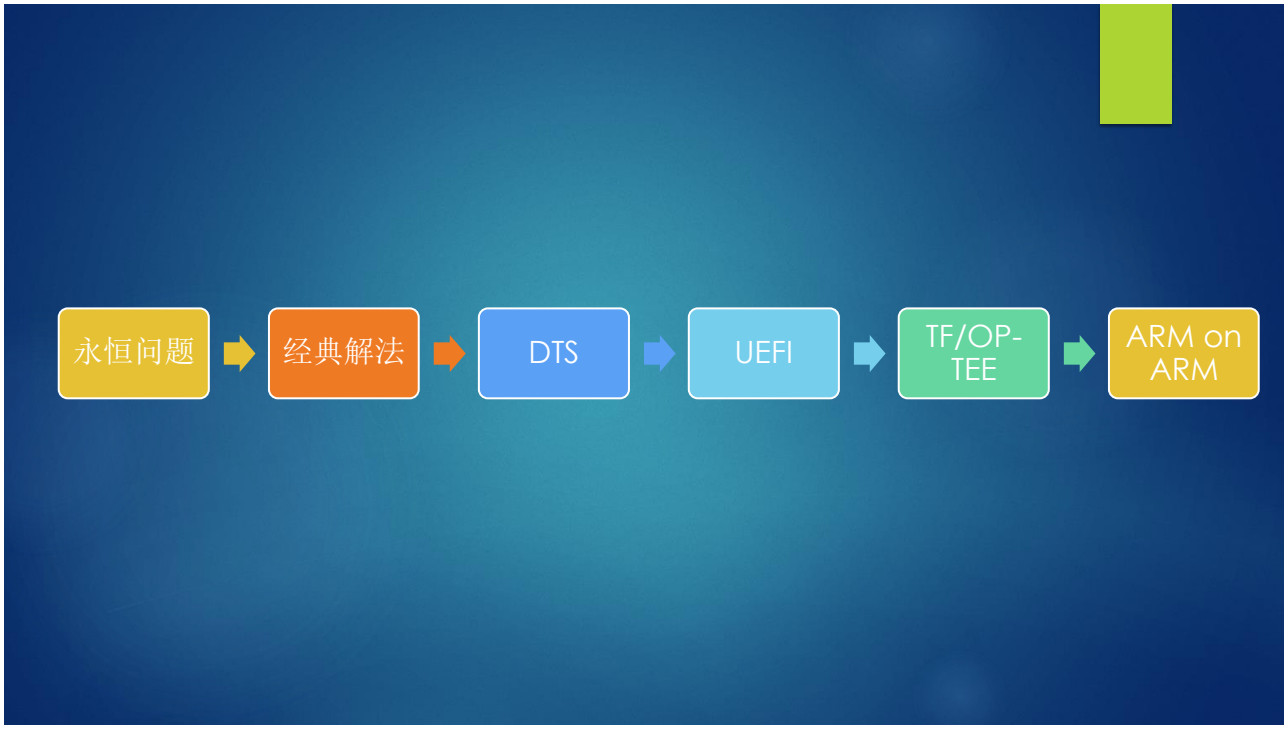
第 254 行，第 24 列 100% Windows (CRLF) UTF-8

dsdt.dsl - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

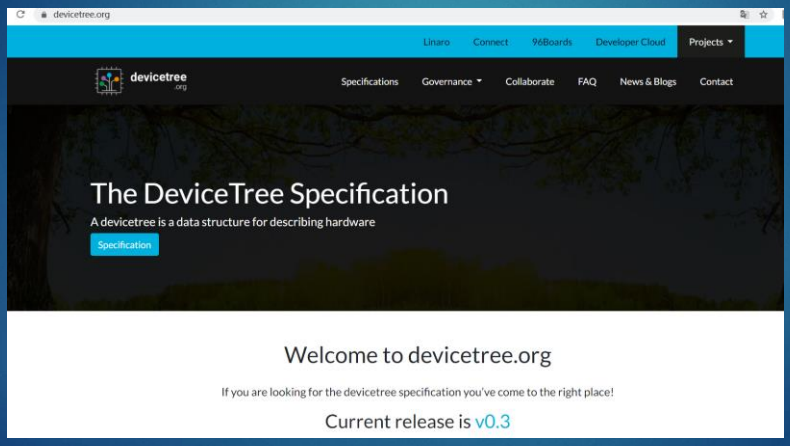
```
        Name (BSWR, 0x00)
        Name (BSWA, 0x00)
        Device (BAT0)
        {
            Name (_HID, EisaId ("PNP0C0A") /* Control Method Battery */)
// _HID: Hardware ID
            Name (_UID, 0x00)  // _UID: Unique ID
            Name (_PCL, Package (0x01)  // _PCL: Power Consumer List
            {
                \_SB
            })
            Name (B0ST, 0x00)
            Name (BT0I, Package (0x0D)
            {
                0x00,
                0xFFFFFFFF,
                0xFFFFFFFF,
                0x01,
                0x2A30,
                0x00,
                0x00,
                0x01,
                0x01,
                "",
                "",
                "",
                "
```

第 17951 行，第 24 列 100% Windows (CRLF) UTF-8




# 官网

devicetree.org

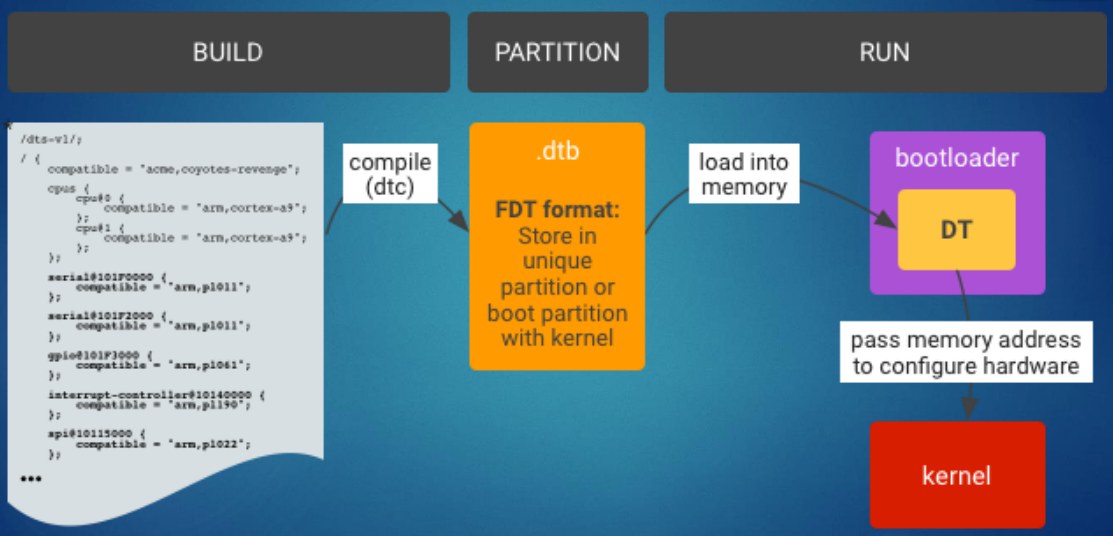


# 版本

Current Release
DeviceTree Specification Release v0.3 - Released 14 February 2020
Latest Release

Past Releases
DeviceTree Specification Release v0.2 - Released 20 December 2017

DeviceTree Specification Release v0.1 - Released 18 December 2017




*http://elinux.org/Device_Tree_Usage#Devices

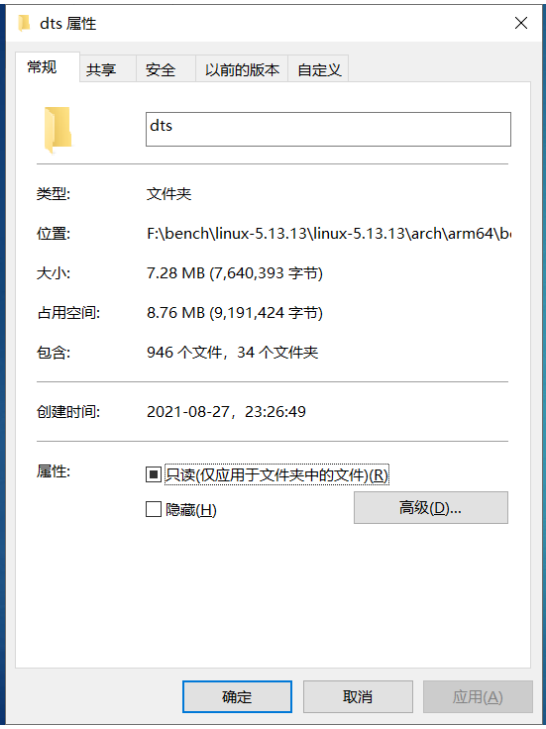
# 术语缩写

| | |
|---|---|
| DT | Device Tree |
| DTB | Device Tree Blob |
| DTBO | Device Tree Blob for Overlay |
| DTC | Device Tree Compiler |
| DTO | Device Tree Overlay |
| DTS | Device Tree Source |
| FDT | Flattened Device Tree, a binary format contained in a .dtb blob file |

bench › linux-5.13.13 › linux-5.13.13 › arch › arm64 › boot › dts › rockchip

搜索"rockchip"

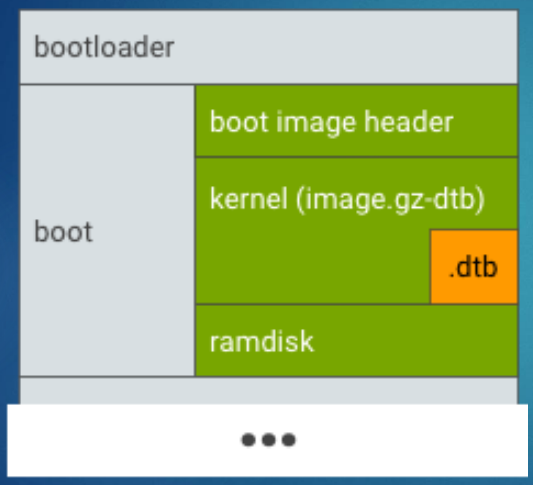
| | | | |
|---|---|---|---|
| Makefile | px30.dtsi | px30-engicam-common.dtsi | px30-engicam-ctouch2.dtsi |
| px30-engicam-edimm2.2.dtsi | px30-engicam-px30-core.dtsi | px30-engicam-px30-core-ctouch2.dts | px30-engicam-px30-core-ctouch2-of10.dts |
| px30-engicam-px30-core-edimm2.2.dts | px30-evb.dts | rk3308.dtsi | rk3308-evb.dts |
| rk3308-roc-cc.dts | rk3318-a95x-z2.dts | rk3326.dtsi | rk3326-odroid-go2.dts |
| rk3328.dtsi | rk3328-a1.dts | rk3328-evb.dts | rk3328-nanopi-r2s.dts |
| rk3328-roc-cc.dts | rk3328-rock64.dts | rk3328-rock-pi-e.dts | rk3368.dtsi |
| rk3368-evb.dtsi | rk3368-evb-act8846.dts | rk3368-geekbox.dts | rk3368-lion.dtsi |
| rk3368-lion-haikou.dts | rk3368-orion-r68-meta.dts | rk3368-px5-evb.dts | rk3368-r88.dts |
| rk3399.dtsi | rk3399-evb.dts | rk3399-ficus.dts | rk3399-firefly.dts |
| rk3399-gru.dtsi | rk3399-gru-bob.dts | rk3399-gru-chromebook.dtsi | rk3399-gru-kevin.dts |
| rk3399-gru-scarlet.dtsi | rk3399-gru-scarlet-inx.dts | rk3399-gru-scarlet-kd.dts | rk3399-hugsun-x99.dts |
| rk3399-khadas-edge.dts | rk3399-khadas-edge.dtsi | rk3399-khadas-edge-captain.dts | rk3399-khadas-edge-v.dts |
| rk3399-kobol-helios64.dts | rk3399-leez-p710.dts | rk3399-nanopc-t4.dts | rk3399-nanopi4.dtsi |
| rk3399-nanopi-m4.dts | rk3399-nanopi-m4b.dts | rk3399-nanopi-neo4.dts | rk3399-nanopi-r4s.dts |
| rk3399-op1-opp.dtsi | rk3399-opp.dtsi | rk3399-orangepi.dts | rk3399-pinebook-pro.dts |
| rk3399pro.dtsi | rk3399pro-rock-pi-n10.dts | rk3399pro-vmarc-som.dtsi | rk3399-puma.dtsi |
| rk3399-puma-haikou.dts | rk3399-rock960.dts | rk3399-rock960.dtsi | rk3399-rock-pi-4.dtsi |
| rk3399-rock-pi-4a.dts | rk3399-rock-pi-4b.dts | rk3399-rock-pi-4c.dts | rk3399-rockpro64.dts |
| rk3399-rockpro64.dtsi | rk3399-rockpro64-v2.dts | rk3399-roc-pc.dts | rk3399-roc-pc.dtsi |
| rk3399-roc-pc-mezzanine.dts | rk3399-sapphire.dts | rk3399-sapphire.dtsi | rk3399-sapphire-excavator.dts |




# 编译器 - DTC

- 安装:
  - sudo apt-get install device-tree-compiler
- 编译:
  - dtc -I dts -O dtb -o devicetree_file_name.dtb devicetree_file_name.dts
- 转换格式:
  - dtc -I dts -O dtb -f devicetree_file_name.dts -o devicetree_file_name.dtb
  - dtc -I dtb -O dts -f devicetree_file_name.dtb -o devicetree_file_name.dts
- 从真实系统转储
  - dtc -I fs -O dts -o extracted.dts /proc/device-tree

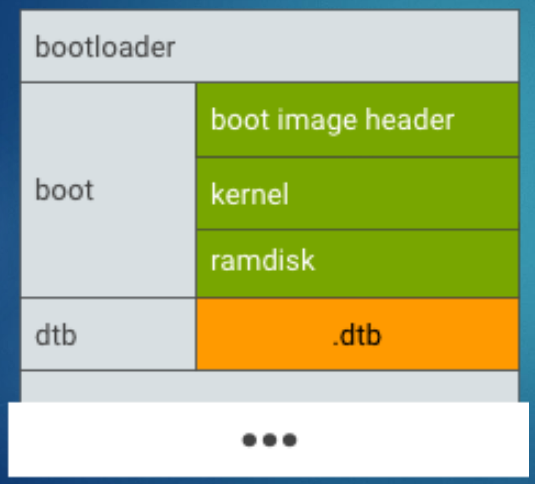
# 将dtb放在boot分区

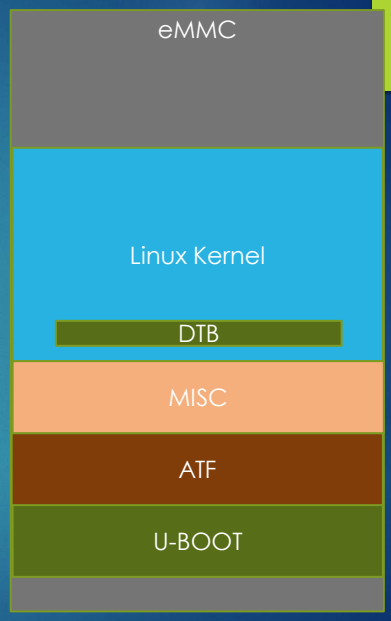



GDK8的做法

https://source.android.com/devices/architecture/dto

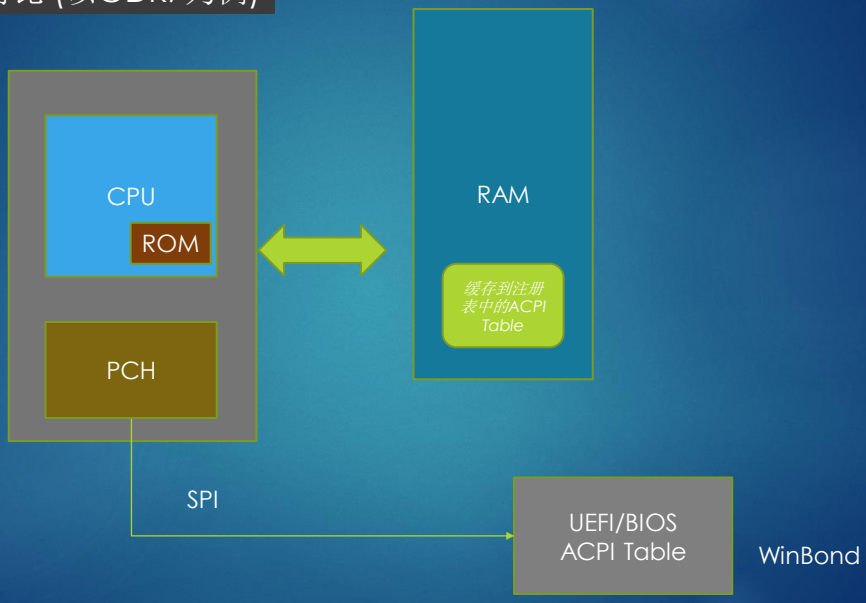
# 将dtb放在单独分区




GPT part: 0, name: uboot, start:0x4000, size:0x2000
GPT part: 1, name: trust, start:0x6000, size:0x2000
GPT part: 2, name: misc, start:0x8000, size:0x2000
GPT part: 3, name: boot, start:0xa000, size:0x10000
GPT part: 4, name: recovery, start:0x1a000, size:0x10000
GPT part: 5, name: backup, start:0x2a000, size:0x10000
GPT part: 6, name: rootfs, start:0x265a000, size:0xc29bfdf

## 与X86对比 (以GDK7为例)



```
geduer@gdk8:/proc/device-tree$ ls
'#address-cells'              gpu-opp-table                  pinctrl                      spdif-sound
'#size-cells'                 gpu@ff300000                   power-management@ff140000    spdif@ff030000
 __symbols__                  h265e@ff330000                 psci                         spi@ff190000
 aliases                      hdmi-sound                     pwm@ff1b0000                 syscon@ff100000
 amba                         hdmi@ff3c0000                  pwm@ff1b0010                 syscon@ff450000
 arm-pmu                      hdmiphy@ff430000               pwm@ff1b0020                 syscon@ff460000
 avsd@ff351000                host-vbus-regulator            pwm@ff1b0030                 thermal-zones
 chosen                       i2c@ff150000                   qos@ff750000                 timer
 clock-controller@ff440000    i2c@ff160000                   qos@ff750080                 tsadc@ff250000
 codec@ff410000               i2c@ff170000                   qos@ff778000                 tsp@ff050000
 compatible                   i2c@ff180000                   regulators                   tve@ff373e00
 cpu0-opp-table               i2s@ff000000                   reserved-memory              usb3-phy@ff470000
 cpuinfo                      i2s@ff010000                   rga@ff3900000                usb@ff580000
 cpus                         i2s@ff020000                   rkvdec-opp-table             usb@ff5c0000
 ddr_timing                   iep@ff3a0000                   rkvdec@ff36000               usb@ff5d0000
 dfi@ff790000                 interrupt-controller@ff811000  rng@ff060000                usb@ff600000
 display-subsystem            interrupt-parent               rockchip-suspend             vcc-phy-regulator
 dmc                          iommu@ff330200                 rockchip-system-monitor      vdd-center
 dmc-opp-table                iommu@ff340800                 rtc-fake                     vdd-log
 dwmmc@ff500000               iommu@ff350800                 saradc@ff280000              venc_srv
 dwmmc@ff510000               iommu@ff360480                 sdio-pwrseq                  vepu@ff340000
 dwmmc@ff520000               iommu@ff373f00                 sdmmc-regulator              vop@ff370000
 dwmmc@ff5f0000               iommu@ff3a0800                 serial-number                vpu_combo
 efuse@ff260000               leds                           serial@ff110000              vpu_service@ff350000
 ethernet@ff540000            memory                         serial@ff120000              watchdog@ff1a0000
 ethernet@ff550000            model                          serial@ff130000              wireless-bluetooth
 external-gmac-clock          name                           skykirin_led                 wireless-wlan
 fiq-debugger                 otg-vbus-regulator             sound                        xin24m
 firmware                     pdm@ff040000                   spdif-out                    xin32k
```

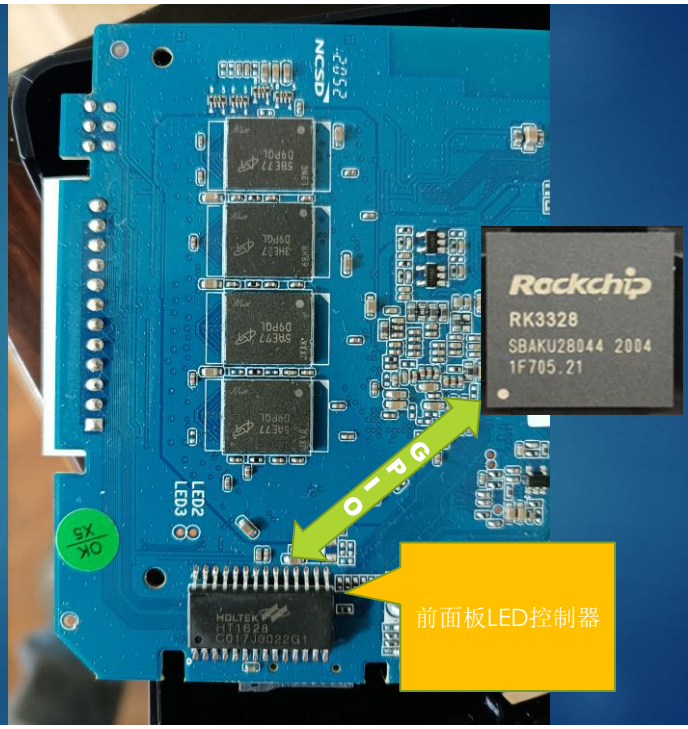
28

# demo

## Dump GDK8的设备树

dtc -I fs -O dts -o gdk8.dts /proc/device-tree


LAN


Rockchip
RK3328
SBAKU28044 2004
1F705.21
LED2
LED3
OK
XS
HOLTEK
HT1628
前面板LED控制器


DTS

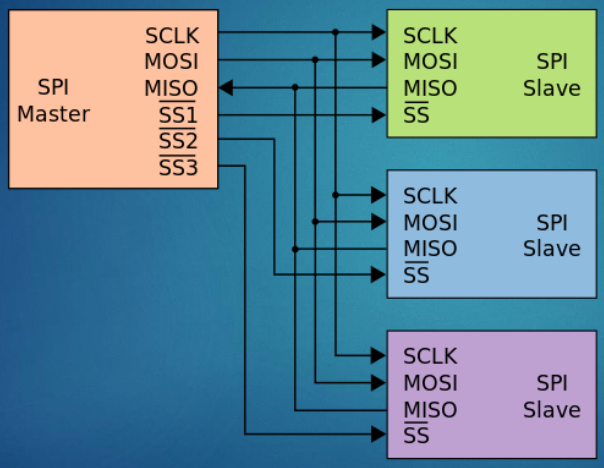
```
skykirin_led {
    compatible = "skykirin-ht1628";
    spi_data = <0x8e 0x16 0x0>;
    spi_clk = <0x8e 0x13 0x0>;
    spi_cs = <0x8e 0x12 0x0>;
    status = "okay";
};
```

出厂时烧录到系统

```
static struct platform_driver HT1628_driver = {
        .probe  = HT1628_probe,
        .remove = HT1628_remove,
   .shutdown     = HT1628_shutdown,
        .driver = {
                .name  = DEVICE_NAME,
                .owner = THIS_MODULE,
                .pm     = &gpio_led_pm_ops,
#ifdef CONFIG_OF
                .of_match_table = of_match_ptr(HT1628_dt_match),
#endif
        },
};
static int __init led_HT1628_init(void)
{
        int ret;
        DBG_PRINT("%s\n==========================================\n", __FUNCTION__);
        ret = platform_driver_register(&HT1628_driver);
        if (ret) {
                printk("[error] %s failed to register HT1628 driver module\n", __FUNCTION__);
                return -ENODEV;
        }
        return ret;
}
```

```
#ifdef CONFIG_OF
static const struct of_device_id HT1628_dt_match[]={
        { .compatible = "skykirin-ht1628",},
        {}
};
MODULE_DEVICE_TABLE(of, HT1628_dt_match);
#endif
```

## Serial Peripheral Interface (SPI)




1980年代由摩托罗拉定义

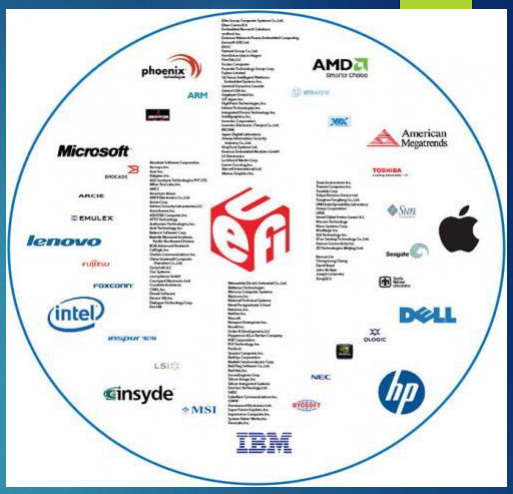
```
skykirin_led {
    compatible = "skykirin-ht1628";
    spi_data = <0x8e 0x16 0x0>;
    spi_clk = <0x8e 0x13 0x0>;
    spi_cs = <0x8e 0x12 0x
    status = "okay";
};
```

三个GPIO的编号

static int gpio_spi_cs,gpio_spi_clk,gpio_spi_data; //gpio_typec_pd,gpio_typec_sel;

永恒问题 → 经典解法 → DTS → UEFI → TF/OP-TEE → ARM on ARM

Intel, Microsoft and Toshiba
ACPI 1.0




UEFI Forum

1999 → 2013.10

# EFI

39

- Extensible Firmware Interface
- Originated from early development of Intel-HP Itanium systems in mid-1990s
- Tiano
- EFI is an interface specification, abstracts the platform from OS
  - Decouples development
- Includes a modular driver model and CPU-independent option ROMs

# UEFI

40

- In July 2005, Intel ceased development of the EFI specification at version 1.10, and contributed it to the Unified EFI Forum, which has evolved the specification as the Unified Extensible Firmware Interface (UEFI).
- UEFI 2.1 Published 2007
- Latest version 2.4
- http://www.uefi.org

arm Developer

Documentation Downloads Community Support

IP Products Tools and Software Architectures Solutions

Home / Architectures / System Architectures / Software Standards / UEFI

# UEFI

## Unified Extensible Firmware Interface

### Overview

The Unified Extensible Firmware Interface (UEFI) is a specification that defines a software interface between an operating system and platform firmware. UEFI is primarily used to define firmware services used to boot an operating system on a platform, but does add a few runtime services. The specification is published by the UEFI forum, where Arm is a board member.
For more detail on the specification, see here.


https://uefi.org/sites/default/files/resources/Arm%20SystemReady%20and%20the%20UEFI%20Firmware%20Ecosystem.pdf

github.com/edk2-porting/edk2-rk3399

Search or jump to... / Pull requests Issues Marketplace Explore

edk2-porting / edk2-rk3399 Public

Code | Issues 2 | Pull requests | Actions | Projects | Wiki | Security | Insights

main | 1 branch | 0 tags | Go to file | Add file | Code

strongtz Update DSDT ... 2 0db6819 on 27 Dec 2020 9 commits

| | | |
|---|---|---|
| device_specific | Initial commit | 13 months ago |
| sdm845Pkg | Update DSDT | 12 months ago |
| .gitignore | EHCI now works | 13 months ago |
| README.md | Update README | 12 months ago |
| build.sh | PoC version | 13 months ago |

README.md

## EDK2 UEFI Firmware For rk3399 SoC

This repo is based on edk2-sdm845 and rk3399-edk2, check out those if you have questions.

Telegram group: https://t.me/UEFIonRK3399

github.com/andreiw/rk3399-edk2

andreiw / rk3399-edk2 Public

forked from pbatard/Rockchip

Code | Pull requests 1 | Actions | Projects | Wiki | Security | Insights

master | 1 branch | 0 tags | Go to file | Add file | Code

This branch is 22 commits ahead of pbatard:master. #2

andreiw Merge pull request #1 from Googulator/master ... bfe5d38 on 24 Jan 2020 42 commits

| | | |
|---|---|---|
| Rk3399Pkg | Use a RockChip HID for UART, not HiSi | 2 years ago |
| README.md | rk3399: update build dirs | 3 years ago |

README.md

## Rockchip RK3399 UEFI Firmware

Now with ACPI...and (almost working) USB!

https://github.com/andreiw/rk3399-edk2

### Notes

This is a version of the Rockchip 3399 UEFI firmware from https://github.com/jeffchenfz/Rockchip updated to build with the latest edk2/edk2-platforms (as of 2018.10.28).

永恒问题 → 经典解法 → DTS → UEFI → TF/OP-TEE → ARM on ARM

# 可信固件组织

https://www.trustedfirmware.org/

TrustedFirmware

Platinum Members

arm, CYPRESS, Google, Linaro, NXP, RENESAS, ST life.augmented

General Members

FUTUREWEI Technologies, NXM, OPEN MOBILE PLATFORM

2021年6月

Diamond Members

arm Google

Platinum Members

CYPRESS EMBEDDED IN TOMORROW Linaro NXP RENESAS

life.augmented

2021年12月

**Membership Class and Fees**

All fees are stated are in US Dollars. Please select your desired Membership class:

| Select | Membership Class | # of Employees [2] | Annual Membership Fees |
|---|---|---|---|
| | Platinum Member[1] | | $50,000 |
| | General Member | 500+ employees | $25,000 |
| | | <500 employees | $10,000 |
| | Community Member | Universities, Not for Profits | $2,500 |
| | Individual Member (by invitation only) | | $500 |

[1] Platinum membership requires an initial two-year membership commitment, annual fee not to be increased during that initial term.

About News & Blogs Security Contact

## Why choose Trusted Firmware?

Trusted Firmware provides a reference implementation of secure world software for processors implementing both the A-Profile and M-Profile Arm architecture. It provides SoC developers and OEMs with a reference trusted code base complying with the relevant Arm specifications. The code on this website is the preferred implementation of Arm specifications, allowing quick and easy porting to modern chips and platforms. This forms the foundations of a Trusted Execution Environment (TEE) on application processors, or the Secure Processing Environment (SPE) of microcontrollers.

### Availability of Trusted Firmware

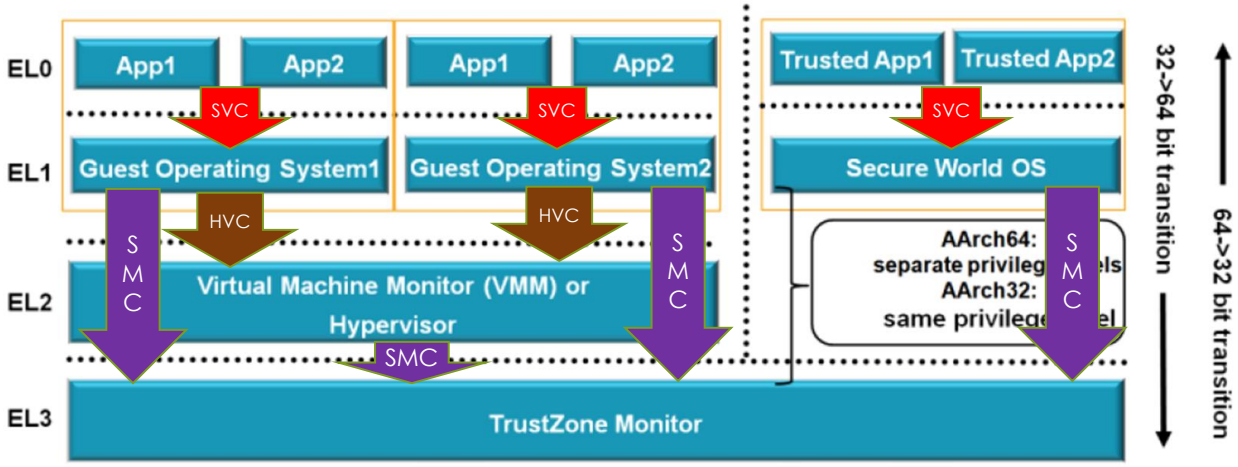
**Support for A-Profile Arm processors / Trusted Firmware-A (TF-A)**

- Support for A-Profile Arm processors (Cortex and Neoverse) is well established and currently available as open source at https://git.trustedfirmware.org/TF-A/trusted-firmware-a.git/ . Functionality focuses on trusted boot and a small trusted runtime (EL3 code).
- Support for the Armv8.4 Secure EL2 architecture extension is also available as open source at https://git.trustedfirmware.org/hafnium/hafnium.git/

# TF-A

https://github.com/ARM-software/arm-trusted-firmware

- 为ARM A处理器开发的固件
- Power State Coordination Interface (PSCI)
- Trusted Board Boot Requirements CLIENT (TBBR-CLIENT)
- SMC Calling Convention
- System Control and Management Interface (SCMI)
- Software Delegated Exception Interface (SDEI)



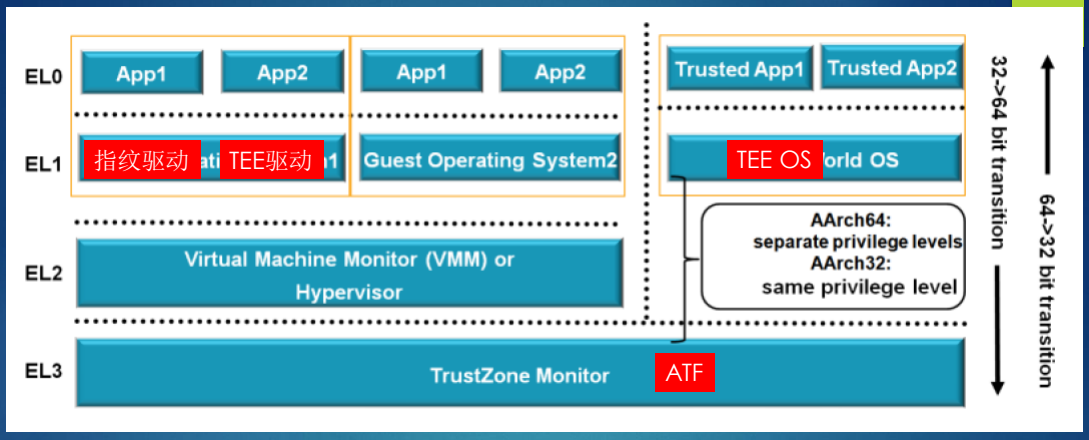

bl1_main.c 9+ | readme.rst | miss_root_option.txt | poplar.rst

bl1 > bl1_main.c > bl1_smc_handler(unsigned int, u_register_t, u_register_t, u_register_t, u_register_t, void *, void *, unsigned int)

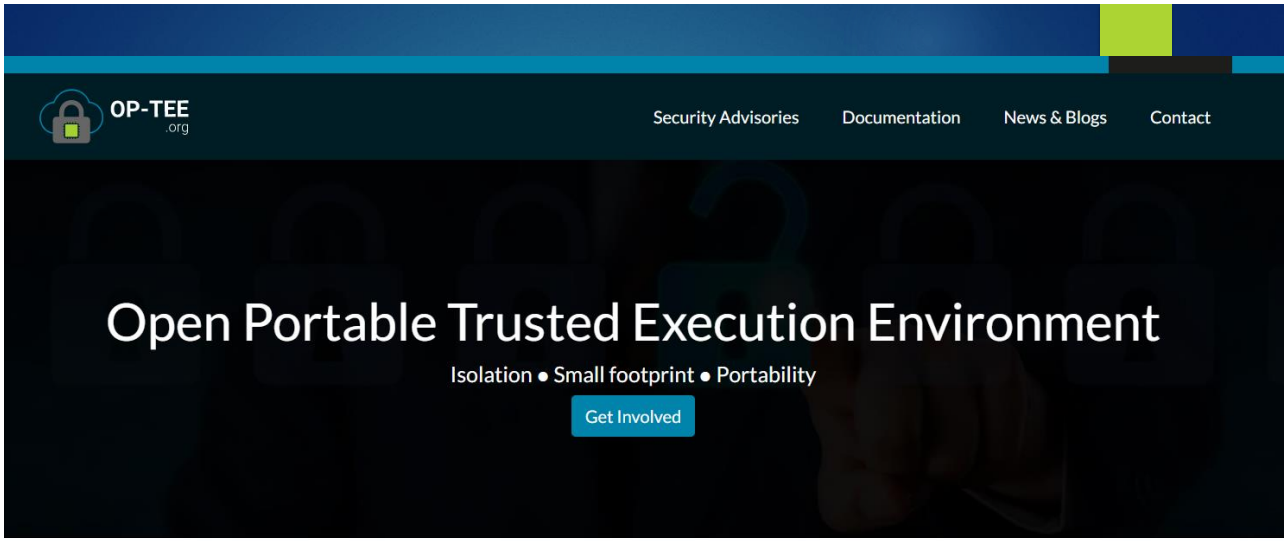
```c
/*******************************************************************************
 * Top level handler for servicing BL1 SMCs.
 *******************************************************************************/
u_register_t bl1_smc_handler(unsigned int smc_fid,
    u_register_t x1,
    u_register_t x2,
    u_register_t x3,
    u_register_t x4,
    void *cookie,
    void *handle,
    unsigned int flags)
{
    /* BL1 Service UUID */
    DEFINE_SVC_UUID2(bl1_svc_uid,
        U(0xd46739fd), 0xcb72, 0x9a4d, 0xb5, 0x75,
        0x67, 0x15, 0xd6, 0xf4, 0xbb, 0x4a);


#if TRUSTED_BOARD_BOOT
    /*
     * Dispatch FWU calls to FWU SMC handler and return its return
     * value
     */
    if (is_fwu_fid(smc_fid)) {
        return bl1_fwu_smc_handler(smc_fid, x1, x2, x3, x4, cookie,
            handle, flags);
    }
#endif

    switch (smc_fid) {
```




ATF - ARM Trusted Firmware
TEE – Trusted Execution Environment




OP-TEE is an open source Trusted Execution Enviroment (TEE) implementing the Arm TrustZone technology. OP-TEE has been ported to many Arm devices and platforms. Originally it was developed as a proprietary TEE solution by ST-Ericsson that later on was moved over to STMicroelectronics.

https://www.op-tee.org/

```
  CC      linuxboot_dma.o
  BUILD   linuxboot_dma.img
  BUILD   linuxboot_dma.raw
  SIGN    linuxboot_dma.bin
  AS      kvmvapic.o
  BUILD   kvmvapic.img
  BUILD   kvmvapic.raw
  SIGN    kvmvapic.bin
  AS      pvh.o
  CC      pvh_main.o
  BUILD   pvh.img
  BUILD   pvh.raw
  SIGN    pvh.bin
make[2]: Leaving directory '/home/yanzi/OP-TEE/qemu/build'
changing dir to build for make ""...
make[2]: Entering directory '/home/yanzi/OP-TEE/qemu/build'
  GIT     ui/keycodemapdb meson tests/fp/berkeley-testfloat-3 tests/fp/berkeley-softfloat-3 dtc capstone slirp
[1/17] Generating qemu-version.h with a custom command (wrapped by meson to capture output)
make[2]: Leaving directory '/home/yanzi/OP-TEE/qemu/build'
make[1]: Leaving directory '/home/yanzi/OP-TEE/qemu'
make run-only
make[1]: Entering directory '/home/yanzi/OP-TEE/build'
ln -sf /home/yanzi/OP-TEE/build/../out-br/images/rootfs.cpio.gz /home/yanzi/OP-TEE/build/../out/bin/

* QEMU is now waiting to start the execution
* Start execution with either a 'c' followed by <enter> in the QEMU console or
* attach a debugger and continue from there.
*
* To run OP-TEE tests, use the xtest command in the 'Normal World' terminal
* Enter 'xtest -h' for help.

# Option "-x" is deprecated and might be removed in a later version of gnome-terminal.#
# Option "-x" is deprecated and might be removed in a later version of gnome-terminal.#
# Use "-- " to terminate the options and put the command line to execute after it.#
# Use "-- " to terminate the options and put the command line to execute after it.#
cd /home/yanzi/OP-TEE/build/../out/bin && /home/yanzi/OP-TEE/build/../qemu/build/aarch64-softmmu/qemu-system-aar
ch64 \
        -nographic \
        -serial tcp:localhost:54320 -serial tcp:localhost:54321 \
        -smp 2 \
        -s -S -machine virt,secure=on,gic-version=3,virtualization=false \
        -cpu cortex-a57 \
        -d unimp -semihosting-config enable=on,target=native \
        -m 1057 \
        -bios bl1.bin            \
        -initrd rootfs.cpio.gz \
        -kernel Image -no-acpi \
        -append 'console=ttyAMA0,38400 keep_bootcon root=/dev/vda2 ' \
        \
        -object rng-random,filename=/dev/urandom,id=rng0 -device virtio-rng-pci,rng=rng0,max-bytes=1024,period=1
000 -netdev user,id=vmnic -device virtio-net-device,netdev=vmnic
QEMU 6.2.50 monitor - type 'help' for more information
(qemu)
```

格蠹上海实验室
里构建的OP-TEE

永恒问题 → 经典解法 → DTS → UEFI → TF/OP-TEE → ARM on ARM

# ARM on ARM

Self-hosted development
Develop ARM software on ARM

arm Developer

Documentation Downloads

IP Products Tools and Software Architectures Solutions

Home / Architectures / System Architectures / Arm SystemReady Certification Program

## Arm SystemReady Certification Program

Overview Specifications SystemReady SR SystemReady ES SystemReady IR SystemReady LS

### Key resources

- Arm SystemReady specifications
- Arm SystemReady launch blog
- SystemReady question and answer video
- SystemReady white paper- Software Just Works on Arm Based Devices
- SystemReady and the UEFI Firmware Ecosystem
- SystemReady high-level overview
- SystemReady ES Test and Certification guide
- SystemReady ES integration guide
- SystemReady IR - IoT integration, test, and certification guide
- Deploying Yocto on SystemReady IR Compliant Hardware guide
- UEFI Drivers
- Contact Arm to become SystemReady

### Other resources

- Software standards
- Platform Security Architecture (PSA)
- PSA Certified
- Neoverse
- Project Cassini

Overview | Specifications | SystemReady SR | SystemReady ES | SystemReady IR | SystemReady LS

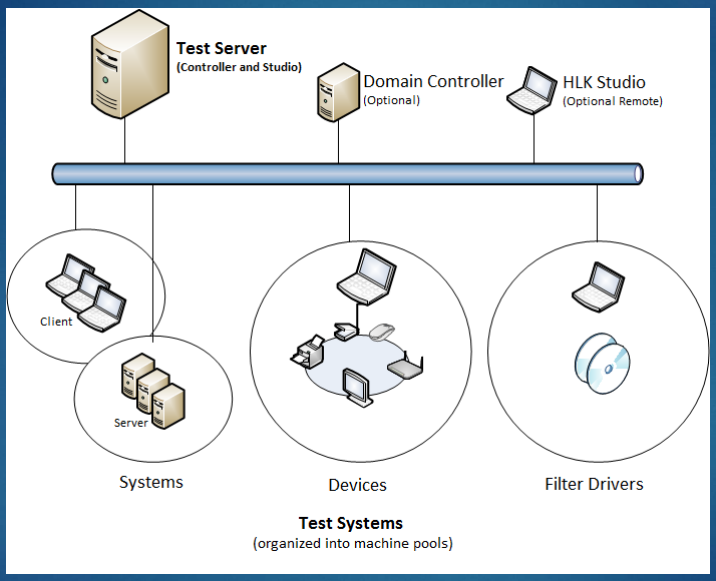
Arm SystemReady is a set of standards and a compliance certification program that ensures software just works. Systems that are designed to just work should install and run generic or specified, off-the-shelf operating systems straight out of the box.

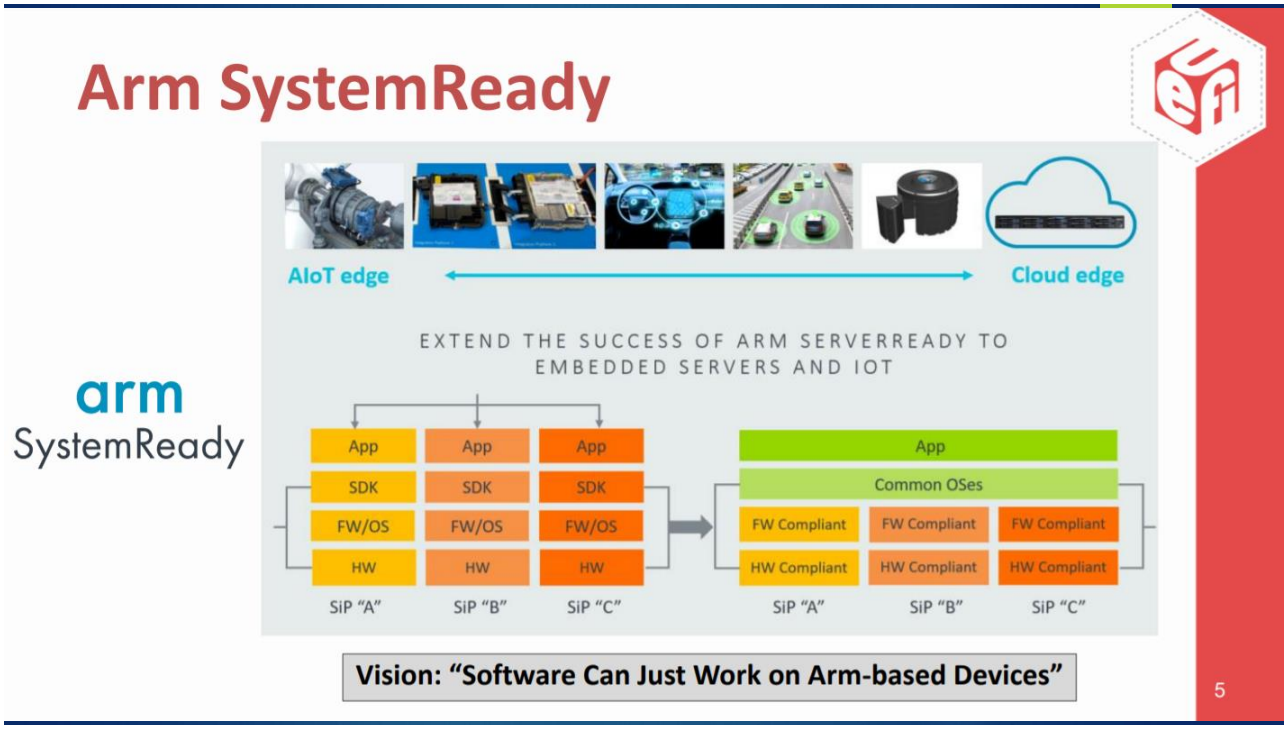
To do this, the system design must follow a set of minimum hardware and firmware requirements. Arm SystemReady builds on and replaces the former successful Arm ServerReady program, originally aimed at servers. Arm SystemReady applies the standards framework to a broader set of devices, initially applying across the server, embedded server and high-performance IoT ecosystems, extending from the cloud to the infrastructure and IoT edge.

This page describes and provides links to the following specifications:

- Arm SystemReady Requirements (SRS) specification
- Arm Base System Architecture (BSA) specification
- Arm Server Base System Architecture (SBSA) supplement specification
- Arm Base Boot Requirements (BBR) specification
- Arm Embedded Base Boot Requirements (EBBR) supplement specification
- Arm Base Boot Security Requirements (BBSR) specification
- Architectural Compliance Suite (ACS)

Windows 10 Compatible | Windows 8 Compatible | Compatible with Windows 7


Test Server
(Controller and Studio)
Domain Controller
(Optional)
HLK Studio
(Optional Remote)
Client
Server
Systems
Devices
Filter Drivers
Test Systems
(organized into machine pools)


Arm SystemReady
arm
SystemReady
AIoT edge
Cloud edge
EXTEND THE SUCCESS OF ARM SERVERREADY TO
EMBEDDED SERVERS AND IOT
App
SDK
FW/OS
HW
SiP "A"
SiP "B"
SiP "C"
App
Common OSes
FW Compliant
HW Compliant
Vision: "Software Can Just Work on Arm-based Devices"
5

# Embedded Base Boot Requirements (EBBR) Specification

***Release v2.0.1***

**Arm Limited and Contributors**




成立于2010年，致力于ARM上的开源软件
目的是打造ARM生态系统

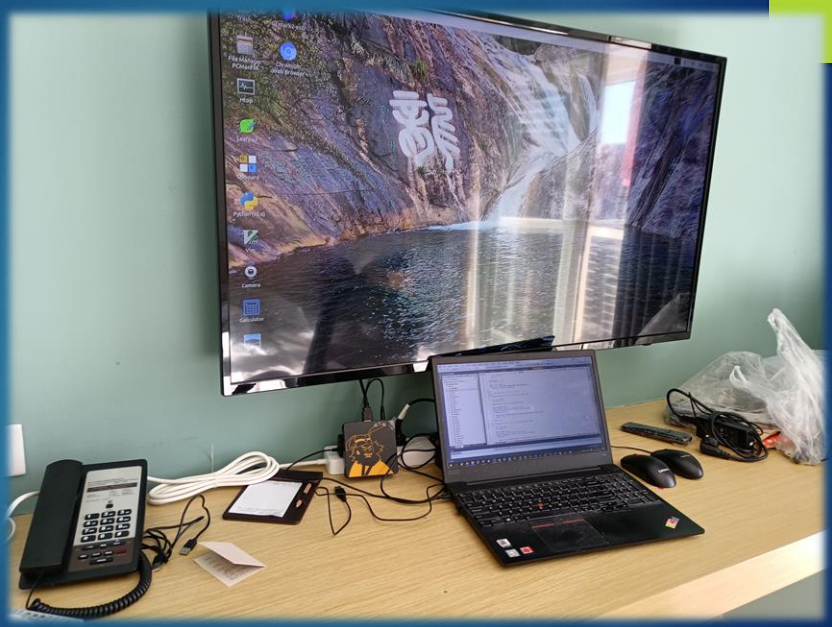
## Linaro Members

GDK8示范了ARM on ARM

## 在GDK8上安装DTC编译器

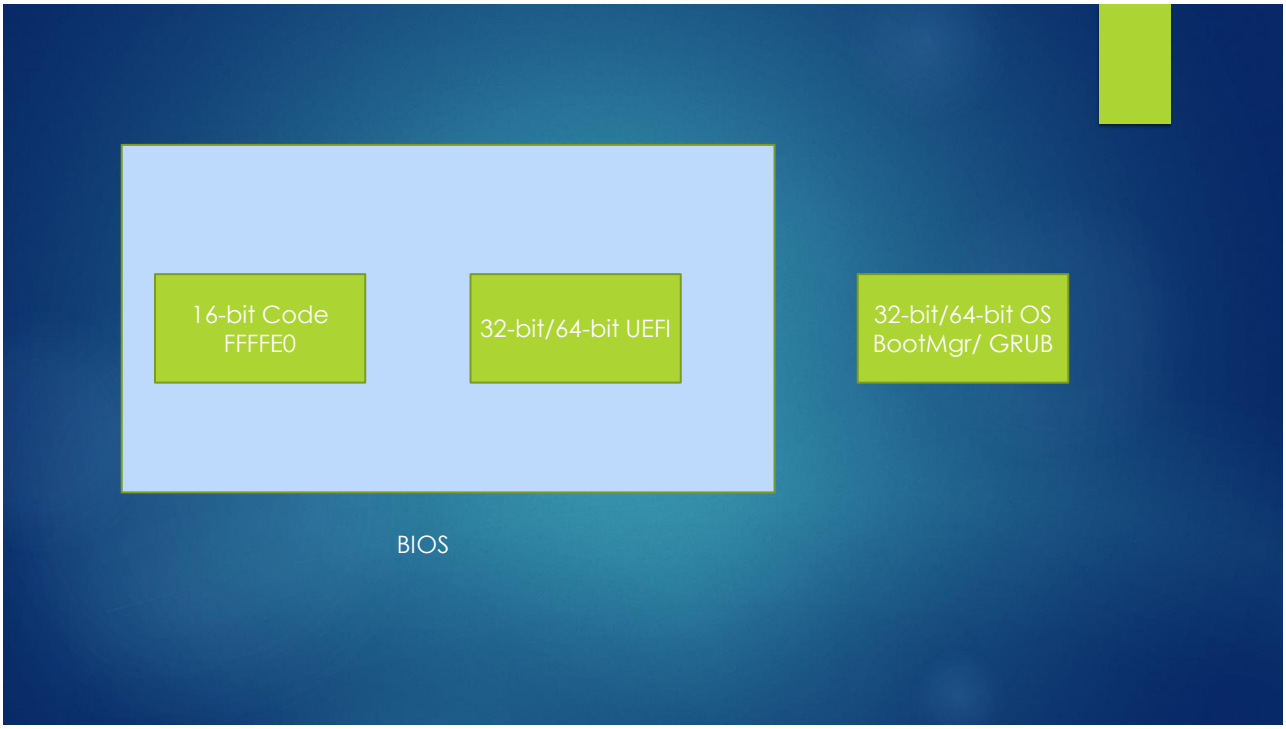
```
Fetched 22.1 MB in 19s (1170 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
334 packages can be upgraded. Run 'apt list --upgradable' to see them.
geduer@gdk8:/proc/device-tree$ sudo apt install device-tree-compiler
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  device-tree-compiler
0 upgraded, 1 newly installed, 0 to remove and 334 not upgraded.
Need to get 230 kB of archives.
After this operation, 427 kB of additional disk space will be used.
Get:1 http://mirrors.aliyun.com/ubuntu-ports bionic/main arm64 device-tree-compiler arm64 1.4.5-3 [230 kB]
Fetched 230 kB in 3s (90.8 kB/s)
Selecting previously unselected package device-tree-compiler.
dpkg: warning: files list file for package 'firefly-multi-rtsp' missing; assuming package has no files currently installed
dpkg: warning: files list file for package 'firefly-3399pronpu-driver' missing; assuming package has no files currently insta
lled
(Reading database ... 93731 files and directories currently installed.)
Preparing to unpack .../device-tree-compiler_1.4.5-3_arm64.deb ...
Unpacking device-tree-compiler (1.4.5-3) ...
Setting up device-tree-compiler (1.4.5-3) ...
geduer@gdk8:/proc/device-tree$
```

NDSTUB就是在GDK8上直接编译出来的，NDK正在用这种方式开发和测试，新的GDK8 RootFS也是在GDK8上制作的



切问而近思

欢迎关注格友公众号


16-bit Code
FFFFE0
32-bit/64-bit UEFI
32-bit/64-bit OS
BootMgr/ GRUB
BIOS