

# Trust 指南

发布版本：1.0

作者邮箱：[chenjh@rock-chips.com](mailto:chenjh@rock-chips.com)

日期：2017.12

文件密级：公开资料

## 前言

### 概述

Trust 作为Rockchip平台SDK里的固件之一，因为涉及安全性和保密性，目前完整源码仅对内部的部分工程师开放（RK322x/RK3328/RK3368/RK3399/平台的基础功已经开源[0]）。本文档仅对Trust 进行概要介绍（以64位平台作为范例），意在让读者明白它在整个系统架构中的角色和作用。同时指导读者在实际使用中遇到问题时如何进行问题收集和反馈。

### 读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

### 产品版本

芯片名称	内核版本
RK3036/RK3126C/RK3288/RK322X/RK3368/RK3328/RK3399	3.10、4.4

### 修订记录

日期	版本	作者	修改说明
2017-12-30	V1.0	陈健洪	初始版本

Trust 指南

ARM TrustZone

1. 系统架构

2. CPU特权等级

Rockchip平台的Trust

1. 实现机制

2. 启动流程

3. 固件获取

4. DTS使能

- 4.1 内核3.10
  - 4.1.1 32位平台
  - 4.1.2 64位平台
- 4.2 内核4.4
  - 4.2.1 32位平台
  - 4.2.2 64位平台
- 4.3 内核Document
- 5. 运行内存和生命周期
  - 5.1 运行内存
  - 5.2 生命周期
- 6. Security
- 7. 功能
  - 7.1 PSCI ( Power State Coordination Interface )
  - 7.2 Secure Monitor
  - 7.3 安全信息的配置
  - 7.4 安全数据的保护
- Rockchip 平台的Trust问题处理
  - 1. 开机log示例
  - 2. 打印信息识别
  - 3. 固件版本号识别
  - 4. PANIC信息识别
    - 4.1 ARM Trusted Firmware 发生panic
    - 4.2 OP-TEE OS发生panic
- 附录参考

---

## ARM TrustZone

---

ARM TrustZone [1]技术是所有Cortex-A类处理器的基本功能，是通过ARM架构安全扩展引入的。这些扩展可在供应商、平台和应用程序中提供一致的程序员模型，同时提供真实的硬件支持的安全环境。

ARM TrustZone 技术是系统范围的安全方法，针对高性能计算平台上的大量应用，包括安全支付、数字版权管理(DRM)、企业服务和基于 Web 的服务。TrustZone技术与Cortex-A处理器紧密集成，并通过AMBA AXI总线和特定的TrustZone系统IP块在系统中进行扩展，所以ARM TrustZone技术是从硬件层次上提供的安全机制。此系统方法意味着可以保护安全内存、加密块、键盘和屏幕等外设，从而可确保它们免遭软件攻击。

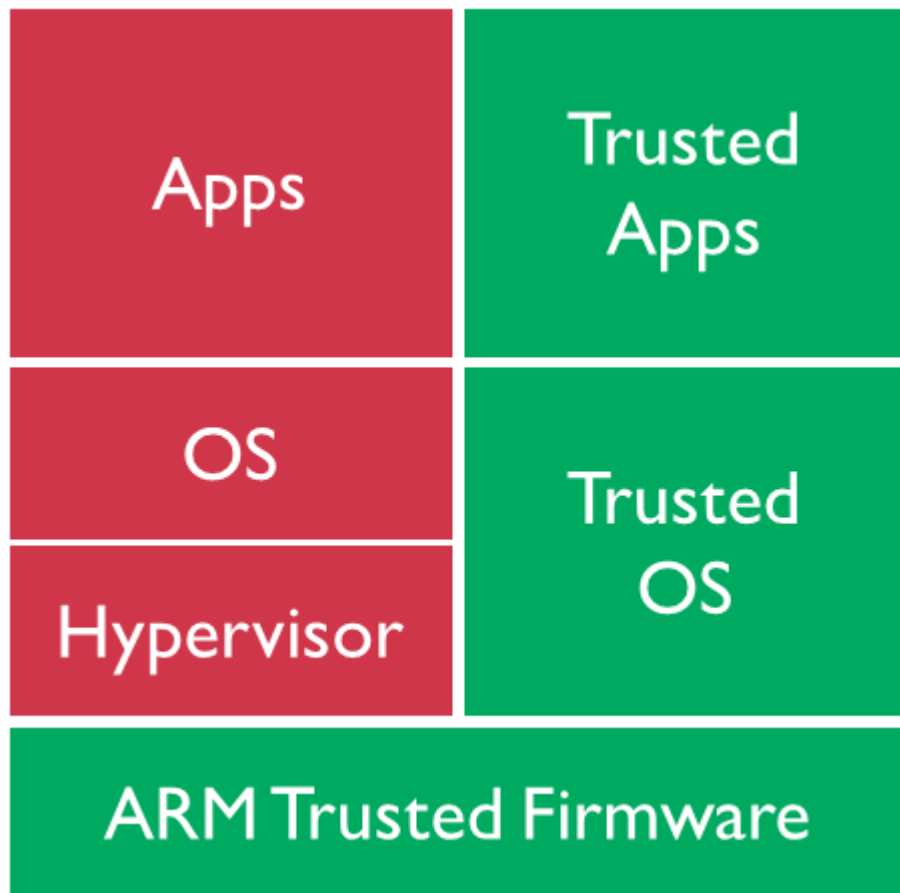
目前用于ARM TrustZone 技术的开源项目中，使用比较广泛的有ARM Trusted Firmware和OP-TEE OS[2]，它们都是针对ARM芯片给出的底层固件开源项目，二者之间可以配合使用或单独使用。

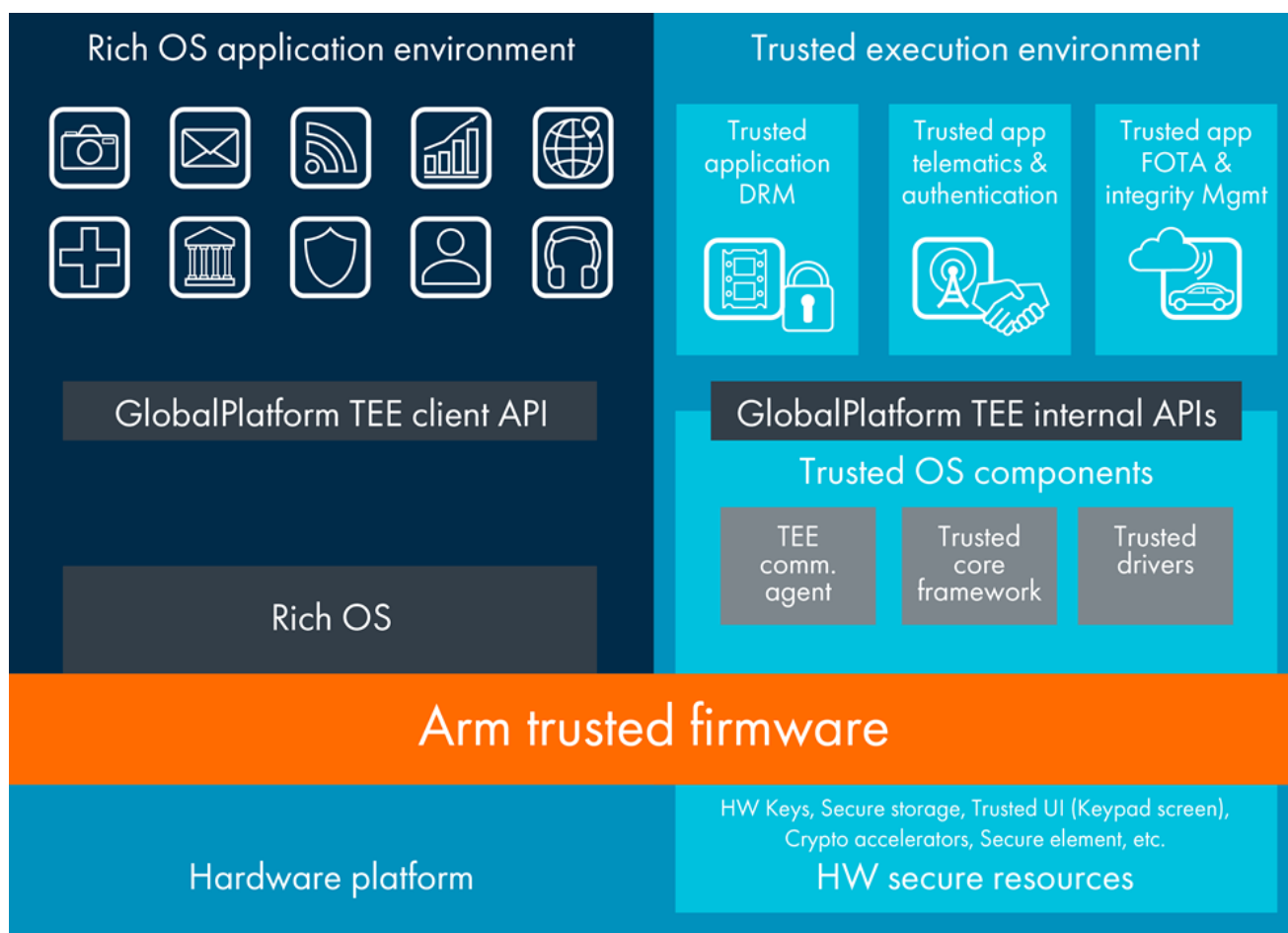
### 1. 系统架构

从系统架构角度看，如下是启用了ARM TrustZone 技术后的64位平台系统架构图。整个系统被分成了两个世界：左边是非安全世界，右边是安全世界。安全世界可以访问两个世界的所有资源，非安全世界只能访问非安全世界的资源，如果非安全世界访问安全世界的资源，则将产生系统硬件总线报错等异常，是无法获取到资源的。

这两个世界之间的往来需要通过ARM Trusted Firmware作为桥梁。当CPU处于非安全世界时，如果想进入安全世界则需要先进入ARM Trusted Firmware（通过ARM的SMC硬件指令），在ARM Trusted Firmware里的Secure Monitor代码会把CPU从非安全身份切换成安全的身份，然后再以安全身份进入安全世界。反之亦然。这个是完全从硬件级别上进行的安全和非安全身份转变。

Rockchip的Trust可以理解为是ARM Trusted Firmware + OP-TEE OS 的功能集合，它实现了安全世界里我们需求的功能以及Secure Monitor(两个世界转换的核心代码)的功能。

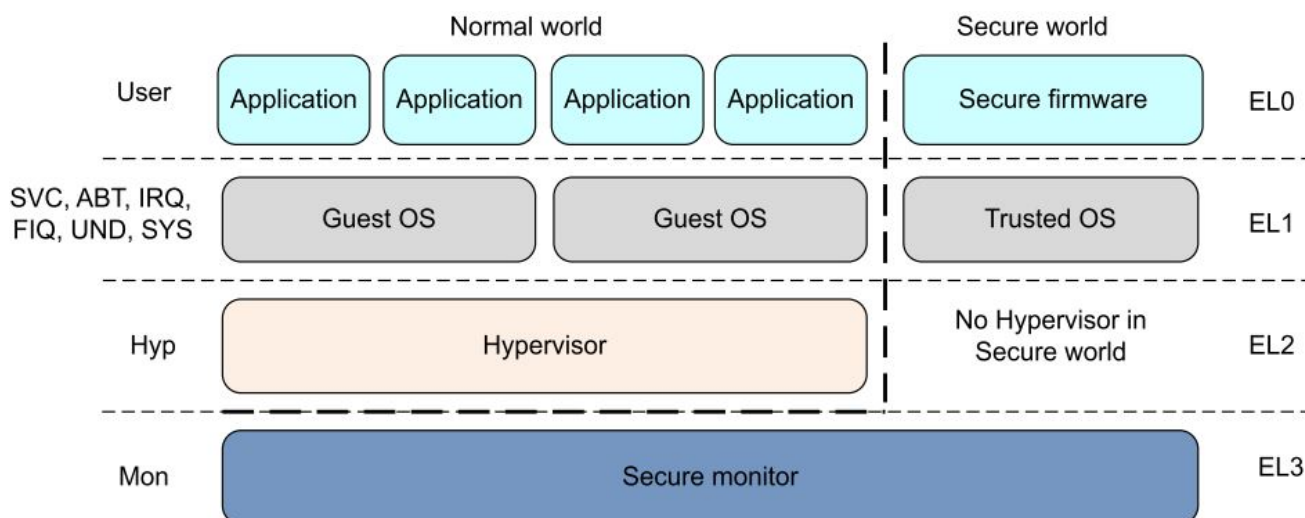


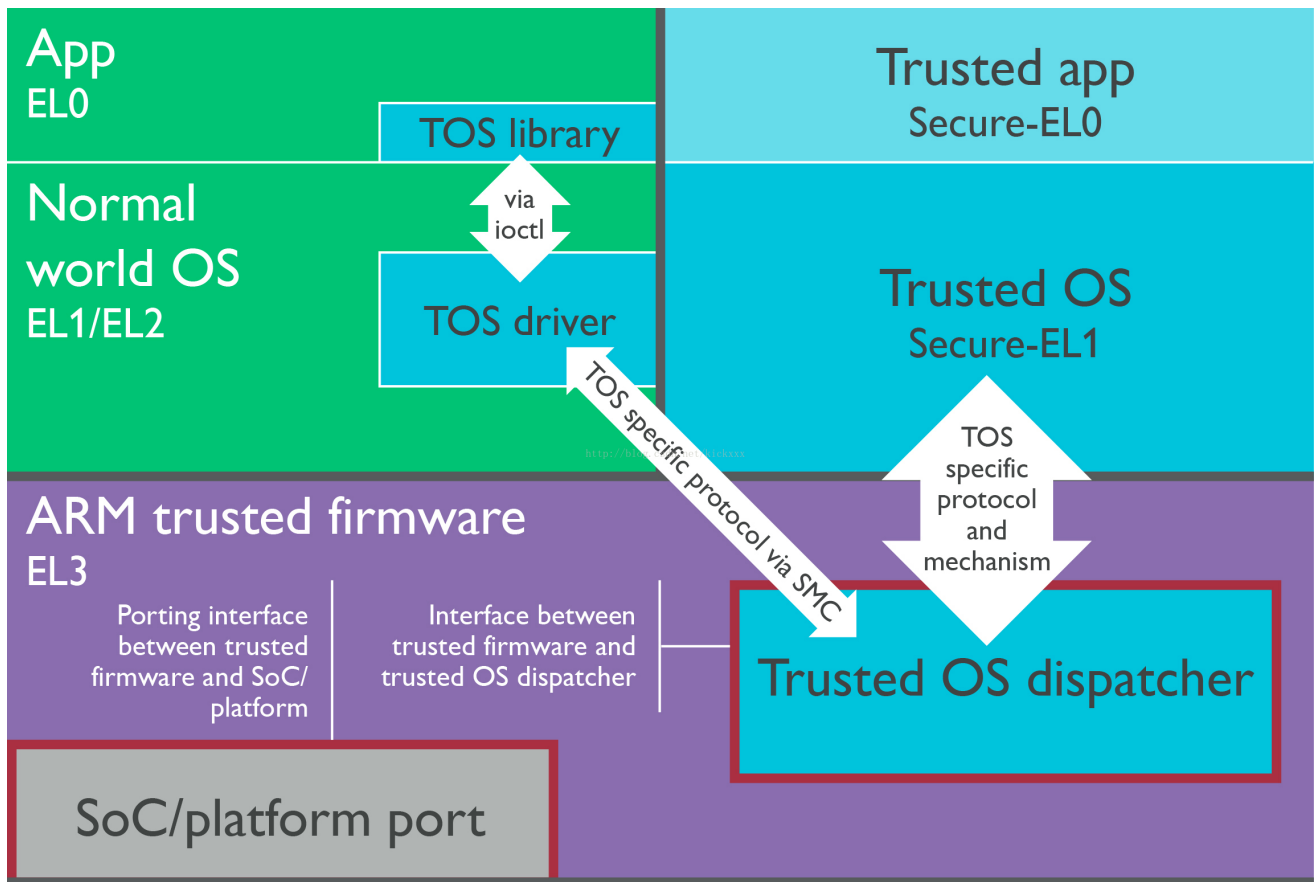


## 2. CPU特权等级

从CPU的视角看，如下是一个标准的启用了ARM TrustZone 技术后的CPU特权模式等级架构图。如果是64位CPU，它的特权等级分为EL0、EL1、EL2、EL3，其中根据CPU所处的世界又分为安全EL0、安全EL1或者非安全EL0、非安全EL1。如果是32位CPU，它的特权等级分为Mon、Hyp、SVC、ABT、IRQ、FIQ、UND、SYS、USER模式，其中SVC、ABT、IRQ、FIQ、UND、SYS、USER也如64位一样有安全和非安全模式之分。

Rockchip的Trust可以理解为是 EL3 + 安全EL1的功能集合。





## Rockchip平台的Trust

### 1. 实现机制

目前Rockchip平台上的64位SoC平台上使用的是ARM Trusted Firmware + OP-TEE OS的组合；32位SoC平台上使用的是OP-TEE OS。

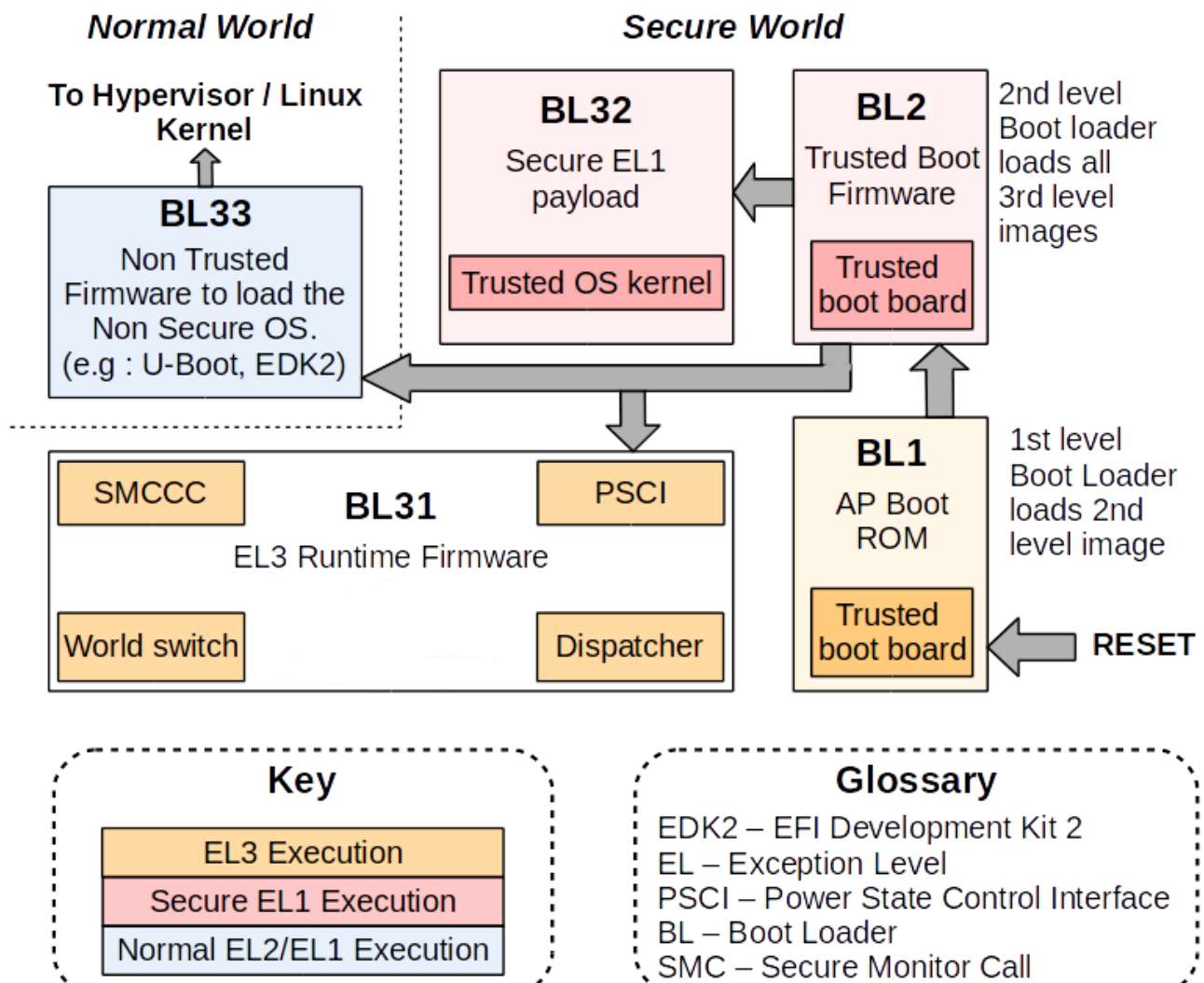
### 2. 启动流程

ARM Trusted Firmware的体系架构里将整个系统分成四种安全等级，分别为：EL0、EL1、EL2、EL3。将整个安全启动的流程阶段定义为：BL1、BL2、BL31、BL32、BL33，其中ARM Trusted Firmware自身的源代码里提供了BL1、BL2、BL31的功能。Rockchip平台仅使用了其中的BL31的功能，BL1和BL2我们有自己的一套实现方案。所以在Rockchip平台上我们一般也可以“默认”ARM Trusted Firmware 指的就是BL31，而BL32使用的则是OP-TEE OS。

如果把上述这种阶段定义映射到Rockchip的平台各级固件上，对应关系为：Maskrom ( BL1 )、Loader ( BL2 )、Trust ( BL31 : ARM Trusted Firmware + BL32 : OP-TEE OS )、U-Boot ( BL33 )。

Android系统的固件启动顺序：

```
1 | Maskrom -> Loader -> Trust -> U-Boot -> kernel -> Android
```



### 3. 固件获取

目前只提供binary文件，不提供源代码。Trust的binary文件提交在U-Boot工程里：

```
1 ./tools/rk_tools/bin/rk30/
2 ./tools/rk_tools/bin/rk31/
3 ./tools/rk_tools/bin/rk32/
4 ./tools/rk_tools/bin/rk33/
```

当编译某个平台的uboot.img的时候，相应平台的trust.img也会同时打包生成在U-Boot的根目录下。其中binary打包成trust.img的时候是通过ini文件进行索引，ini文件在U-Boot工程里：

```
1 tools/rk_tools/RKTRUST/
```

### 4. DTS使能

#### 4.1 内核3.10

##### 4.1.1 32位平台

(1) 增加psci节点

```

1 psci {
2     compatible      = "arm,psci";
3     method          = "smc";
4     cpu_suspend     = <0x84000001>;
5     cpu_off         = <0x84000002>;
6     cpu_on          = <0x84000003>;
7     affinity_info   = <0x84000004>;
8 };

```

( 2 ) 在chosen节点或者parameter里增加 : psci=enable

```

1 chosen {
2     bootargs = "psci=enable vmalloc=496M cma=4M rockchip_jtag";
3 };

```

#### 4.1.2 64位平台

( 1 ) 增加psci节点 :

```

1 psci {
2     compatible = "arm,psci-0.2";
3     method = "smc";
4 };

```

( 2 ) cpu节点下面增加 : enable-method = "psci";

```

1 cpus {
2     #address-cells = <2>;
3     #size-cells = <0>;
4
5     cpu@0 {
6         device_type = "cpu";
7         compatible = "arm,cortex-a53", "arm,armv8";
8         reg = <0x0 0x0>;
9         enable-method = "psci";
10        cpu-idle-states = <&CPU_SLEEP>;
11    };
12    cpu@1 {
13        device_type = "cpu";
14        compatible = "arm,cortex-a53", "arm,armv8";
15        reg = <0x0 0x1>;
16        enable-method = "psci";
17        cpu-idle-states = <&CPU_SLEEP>;
18    };
19    cpu@2 {
20        device_type = "cpu";
21        compatible = "arm,cortex-a53", "arm,armv8";
22        reg = <0x0 0x2>;
23        enable-method = "psci";
24        cpu-idle-states = <&CPU_SLEEP>;
25    };

```

```

26     cpu@3 {
27         device_type = "cpu";
28         compatible = "arm,cortex-a53", "arm,armv8";
29         reg = <0x0 0x3>;
30         enable-method = "psci";
31         cpu-idle-states = <&CPU_SLEEP>;
32     };
33
34     .....
35 };

```

## 4.2 内核4.4

### 4.2.1 32位平台

增加psci节点即可：

```

1 psci {
2     compatible = "arm,psci-1.0";
3     method = "smc";
4 };

```

### 4.2.2 64位平台

(1) 增加psci节点：

```

1 psci {
2     compatible = "arm,psci-1.0";
3     method = "smc";
4 };

```

(2) cpu节点下面增加：enable-method = "psci";

```

1 cpus {
2     #address-cells = <2>;
3     #size-cells = <0>;
4
5     cpu@0 {
6         device_type = "cpu";
7         compatible = "arm,cortex-a53", "arm,armv8";
8         reg = <0x0 0x0>;
9         enable-method = "psci";
10        cpu-idle-states = <&CPU_SLEEP>;
11    };
12    cpu@1 {
13        device_type = "cpu";
14        compatible = "arm,cortex-a53", "arm,armv8";
15        reg = <0x0 0x1>;
16        enable-method = "psci";
17        cpu-idle-states = <&CPU_SLEEP>;
18    };
19    cpu@2 {

```



```

20         device_type = "cpu";
21         compatible = "arm,cortex-a53", "arm,armv8";
22         reg = <0x0 0x2>;
23         enable-method = "psci";
24         cpu-idle-states = <&CPU_SLEEP>;
25     };
26     cpu@3 {
27         device_type = "cpu";
28         compatible = "arm,cortex-a53", "arm,armv8";
29         reg = <0x0 0x3>;
30         enable-method = "psci";
31         cpu-idle-states = <&CPU_SLEEP>;
32     };
33
34     .....
35 };

```

### 4.3 内核Document

内核Document里提供了关于psci的相关说明：

```
1 | ./Documentation/devicetree/bindings/arm/psci.txt
```

## 5. 运行内存和生命周期

### 5.1 运行内存

ARM Trusted Firmware 运行在DRAM起始偏移0M~2M的空间，以0x10000（64KB）作为程序入口地址。

OP-TEE OS 运行在DRAM起始偏移132M~148M之间（结束地址依各平台需求而定）以0x08400000（132M）作为入口地址。

### 5.2 生命周期

Trust自上电初始化之后就始终常驻于内存之中，完成着自己的使命。

## 6. Security

在第一章节里我们介绍了启用ARM TrustZone 后系统被分为了安全世界和非安全世界。那么在Rockchip平台上CPU运行在哪些固件时属于安全世界，哪些固件又属于非安全世界呢？具体区分如下：Loader、Trust运行在安全世界；U-Boot、kernel、Android运行在非安全世界里（安全的driver、APP除外）。

## 7. 功能

### 7.1 PSCI ( Power State Coordination Interface )

通常各家SoC厂商的芯片在IC设计上具有明显差异，尤其是CPU的电源状态管理部分。各家SoC厂商有自己的一套软件流程来管理CPU电源状态，所以内核里的这部分代码碎片化比较明显，很难进行高度统一，显然内核很不愿意这方面一直维持碎片化的现状。而且普通开发者一般也不是很关心这部分实现，因为这部分软件实现跟CPU体系架构、IC设计紧密相关，要完全理解或者自己实现都存在一定难度。

基于上述原因，内核更倾向于把CPU的电源管理放到各SoC厂商自己的firmware里，内核只要专注于CPU控制策略，让内核代码更加高度统一。因此后来内核框架增加了PSCI ( Power State Coordination Interface ) [3]接口来实现这一目的。

PSCI是一套CPU core电源管理相关的接口，本质上是通过ARM的SMC硬件指令陷入到Trust里完成以上相关的操作: CPU打开、CPU关闭、系统深度休眠、系统复位、系统关机，等等。主要包括：

```
1 PSCI_VERSION
2 PSCI_FEATURES
3 CPU_ON
4 CPU_OFF
5 CPU_SUSPEND
6 SYSTEM_SUSPEND
7 AFFINITY_INFO
8 SYSTEM_OFF
9 SYSTEM_RESET
10 .....
```

#### 4.4 内核相关代码路径

```
1 ./arch/arm/kvm/psci.c
2 ./arch/arm/kernel/smccc-call.S
3 ./arch/arm64/kernel/psci.c
4 ./arch/arm64/kernel/smccc-call.S
5 ./drivers/firmware/psci.c
6 ./drivers/firmware/rockchip-sip.c
```

#### 3.10 内核相关代码路径

```
1 ./arch/arm/kernel/psci.c
2 ./arch/arm64/kernel/psci.c
3 ./arch/arm/mach-rockchip/psci.c
```

## 7.2 Secure Monitor

Secure Monitor是CPU往来安全世界和非安全世界进行状态转换的桥梁。Secure Monitor的代码是在Trust中实现的。如果没有这部分代码，CPU将无法进行安全/非安全状态的切换，ARM TrustZone技术也就失去了它的意义和作用。

那么如何进入Secure Monitor模式呢？需要通过SMC硬件指令实现，如下是ARM技术手册的明确说明：

The Secure Monitor Call exception is implemented only as part of the Security Extensions. The Secure Monitor Call instruction, SMC, requests a Secure Monitor function, causing the processor to enter Monitor mode.

## 7.3 安全信息的配置

ARM TrustZone技术除了本身Cortex-A处理器紧密集成，还需要通过AMBA AXI总线和特定的TrustZone系统IP块在系统中进行扩展，因此有一系列相关IP模块的安全信息需要进行配置，这部分配置一般都在Trust里完成。

## 7.4 安全数据的保护

安全数据保护。例如：安全支付、数字版权管理 (DRM)、企业服务和基于 Web 的服务等相关安全信息的存储保护。

## Rockchip 平台的Trust问题处理

目前对外发布的固件只提供Trust的binary文件，不提供源代码。目前对于Trust的调试方式比较少，更多需要借助专门的jtag工具来进行分析，当Trust 出问题的时候普通使用者一般并不具备自行调试和解决问题的能力，所以出现问题时请尽量保护好现场、收集足够多的信息反馈给负责Trust的maintainer。因此通常使用者应当知道哪些是Trust的打印信息、Trust对应的版本号、哪些是Trust的PANIC信息等。

### 1. 开机log示例

```
1 NOTICE: BL31: v1.3(debug):4c793da
2 NOTICE: BL31: Built : 18:13:44, Dec 25 2017
3 NOTICE: BL31:Rockchip release version: v1.3
4 INFO: ARM GICv2 driver initialized
5 INFO: Using opteed sec cpu_context!
6 INFO: boot cpu mask: 1
7 INFO: plat_rockchip_pmu_init: pd status 0xe
8 INFO: BL31: Initializing runtime services
9 INFO: BL31: Initializing BL32
10 INF [0x0] TEE-CORE:init_primary_helper:337: Initializing (1.1.0-127-g27532f4 #54 Mon Dec 18
    02:01:14 UTC 2017 aarch64)
11 INF [0x0] TEE-CORE:init_primary_helper:338: Release version: 1.4
12 INF [0x0] TEE-CORE:init_tee_core:83: tee_core inits done
13 INFO: BL31: Preparing for EL3 exit to normal world
14 INFO: Entry point address = 0x200000
15 INFO: SPSR = 0x3c9
```

### 2. 打印信息识别

除去开机阶段的打印信息，通常在运行过程中：

ARM Trusted Firmware 打印格式（不带有时间戳）：

```
1 INFO: *****
```

OP-TEE OS打印格式（不带有时间戳）：

```
1 INF [0x0] TEE-CORE: *****
```

### 3. 固件版本号识别

ARM Trusted Firmware的版本号：4c793da。

```
1 NOTICE: BL31: v1.3(debug):4c793da
```

OP-TEE OS的版本号：27532f4（忽略最前面的g）。

```
1 INF [0x0] TEE-CORE:init_primary_helper:337: Initializing (1.1.0-127-g27532f4 #54 Mon Dec 18
  02:01:14 UTC 2017 aarch64)
```

## 4. PANIC信息识别

### 4.1 ARM Trusted Firmware 发生panic

```
1 Unhandled Exception in EL3.
2 x30 = 0x00000000ff00fff0
3 x0 = 0x00000000000101c0
4 x1 = 0x0000000000000000
5 x2 = 0x0000000000000000
6 x3 = 0x0000000000000000
7 x4 = 0x0000000000cd383b
8 x5 = 0x0000000000080001
9 x6 = 0x0000000080803520
10 x7 = 0x0000000000342a0
11 x8 = 0x00000000000101c0
12 x9 = 0x0000000000000000
13 x10 = 0x0000000000000000
14 x11 = 0x0000000000000000
15 x12 = 0x0000000000000001
16 x13 = 0x00000000000101b8
17 x14 = 0x000000000001a950
18 x15 = 0x0000000000000000
19 x16 = 0x00000000000101c0
20 x17 = 0x0000000000000000
21 x18 = 0x0000000000000000
22 x19 = 0x0000000000000000
23 x20 = 0x0000000040000000
24 x21 = 0x0000000000000040
25 x22 = 0x00000000000305b0
26 x23 = 0x000000000001016c
27 x24 = 0x00000000000101c0
28 x25 = 0x0000000000000000
29 x26 = 0x0000000000000000
30 x27 = 0x0000000000000000
31 x28 = 0x0000000000035bf8
32 x29 = 0x0000000000000000
33 scr_el3 = 0x00000000000101c0
34 sctlr_el3 = 0x0000000000000000
35 cptra_el3 = 0x0000000000000000
36 tcr_el3 = 0x0000000000000000
37 daif = 0x0000000000000238
38 mair_el3 = 0x0000000000cd383b
39 spsr_el3 = 0x0000000000000000
40 elr_el3 = 0x0000000080803520
41 ttbr0_el3 = 0x00000000000101c0
42 esr_el3 = 0x0000000000000000
43 far_el3 = 0x0000000000000000
```

```
44 | spsr_el1 =      0x000000000000101c0
45 | elr_el1 =      0x00000000000000000
46 | spsr_abt =      0x00000000000000000
47 | .....
```

## 4.2 OP-TEE OS发生panic

```
1 | core data-abort at address 0xc121b16c
2 |
3 | fsr 0x000000805  ttbr0 0x6847446a  ttbr1 0x6847006a  cidr 0x2
4 | cpu #0          cpsr 0x200001d1
5 | r0 0x20068000    r4 0x68407195    r8 0x00000000    r12 0x00000000
6 | r1 0x00000049    r5 0x6848068b    r9 0x6840a3bd    sp 0xc121b1a4
7 | r2 0x6848068c    r6 0x6848068c    r10 0x684808cc   lr 0x684296a6
8 | r3 0x0000001f    r7 0x00000001    r11 0x68404f9d   pc 0x6840041c
9 |
10 | ERR [0x0] TEE-CORE:tee_pager_handle_fault:125: Unexpected page fault! Trap CPU
11 | PANIC: tee_pager_handle_fault core/arch/arm/include/mm/tee_pager.h:126
```

---

## 附录参考

[0] 开源代码下载：

ARM Trusted Firmware：<https://github.com/ARM-software/arm-trusted-firmware>

OP-TEE OS：[https://github.com/OP-TEE/optee\\_os](https://github.com/OP-TEE/optee_os)

[1] ARM TrustZone：

<https://www.arm.com/products/security-on-arm/trustzone>

<https://developer.arm.com/technologies/trustzone>

[2] op-tee官网：<https://www.op-tee.org/>

[3] PSCI：

[http://infocenter.arm.com/help/topic/com.arm.doc.den0022c/DEN0022C\\_Power\\_State\\_Coordination\\_Interface.pdf](http://infocenter.arm.com/help/topic/com.arm.doc.den0022c/DEN0022C_Power_State_Coordination_Interface.pdf) "Power State Coordination Interface PDD (ARM DEN 0022C)"