



D1-H Linux SID 开发指南

版本号: 1.0
发布日期: 2021.04.14

版本历史

版本号	日期	制/修订人	内容描述
1.0	2021.04.14	AWA0480	添加初版



目 录

1	前言	1
1.1	编写目的	1
1.2	适用范围	1
1.3	相关人员	1
1.4	术语、定义、缩略语	1
2	模块描述	2
2.1	模块功能	2
2.1.1	Chip ID 功能	2
2.1.2	SoC Version 功能	2
2.1.3	Efuse 功能	2
2.2	模块位置	3
2.3	模块 device tree 配置说明	3
2.4	kernel menuconfig 配置	5
2.4.1	sunxi_info 的驱动的配置	5
2.4.2	sid 的驱动的配置	5
2.5	模块源码结构	6
3	模块设计	7
3.1	结构框图	7
3.2	关键数据定义	7
3.2.1	常量及宏定义	7
3.2.1.1	key 的名称定义	7
3.2.2	关键数据结构	8
3.2.2.1	soc_ver_map	8
3.2.2.2	soc_ver_reg	8
3.2.3	全局变量	9
3.3	模块流程设计	9
3.3.1	SoC 信息读取流程	9
4	接口设计	10
4.1	接口函数	10
4.1.1	sunxi_get_platform()	10
4.1.2	sunxi_get_soc_chipid()	10
4.1.3	sunxi_get_serial()	10
4.1.4	sunxi_get_soc_chipid_str()	11
4.1.5	sunxi_get_soc_ver()	11
4.2	内部函数	11
4.2.1	sunxi_get_base()	11
4.2.2	sunxi_put_base()	11
4.2.3	sid_rd_bits()	12

5 可测试性	13
5.1 sysfs 调试接口	13



插 图

2-1 带有 SRAM 的 SID 模块硬件结构	3
2-2 SID 与其他模块的关系	3
2-3 内核 menuconfig 菜单	5
2-4 内核 menuconfig 菜单	6
3-1 SID 驱动的内部结构	7
3-2 SoC 信息读取流程	9



1 前言

1.1 编写目的

介绍 Linux 内核中基于 Sunxi 硬件平台的 SID 模块驱动的详细设计，为软件编码和维护提供基础。

1.2 适用范围

内核版本 Linux-5.4 的平台。

1.3 相关人员

SID 驱动、Efuse 驱动、Sysinfo 驱动的维护、应用开发人员等。

1.4 术语、定义、缩略语

表 1-1: 术语表

术语或缩略	描述
Efuse	Electronic Fuse，电子熔丝
SID	Security ID，特指 AW SoC 中的 SID 模块
Sysinfo	一个字符型设备驱动，用于方便用户空间获取、调试 SID 信息
sunxi	指 Allwinner 的一系列 SOC 硬件平台。

2 模块描述

2.1 模块功能

SID 提供的功能可以分为四大部分：ChipID、SoC Version、Efuse 功能、一些状态位。

2.1.1 Chip ID 功能

对于全志的 SoC 来说，ChipID 用于该 SoC 的唯一标识。

ChipID 由 4 个 word（16 个 byte）组成，共 128bit，通常放在 Efuse（见 2.1.3 节）的起始 4 个 word。

2.1.2 SoC Version 功能

严格讲 SoC Version 包含两部分信息：

1. Bonding ID，表示不同封装。
2. Version，表示改版编号。

说明

Version 所在的寄存器不在 **SID** 模块内部，但软件上为了统一管理，都归属为 **SID** 模块。

API 会返回这两个信息的组合值，由应用去判断和做出相应的处理。

2.1.3 Efuse 功能

对软件来说，Efuse 中提供了一个可编程的永久存储空间，特点是每一位只能写一次（从 0 到 1）。

Efuse 接口方式，Efuse 容量有 2048bit，采用 SRAM 方式。带有 SRAM 的硬件结构示意图如下：

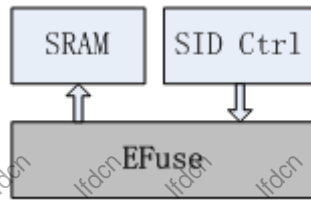


图 2-1: 带有 SRAM 的 SID 模块硬件结构

2.2 模块位置

SID 是一个比较独立的模块，在 Linux 内核中没有依赖其他子系统，在 Sunxi 平台默认是关闭状态，存放在 drivers/soc/sunxi 目录中。

SID 为其他模块提供 API 的调用方式。关系如下图：

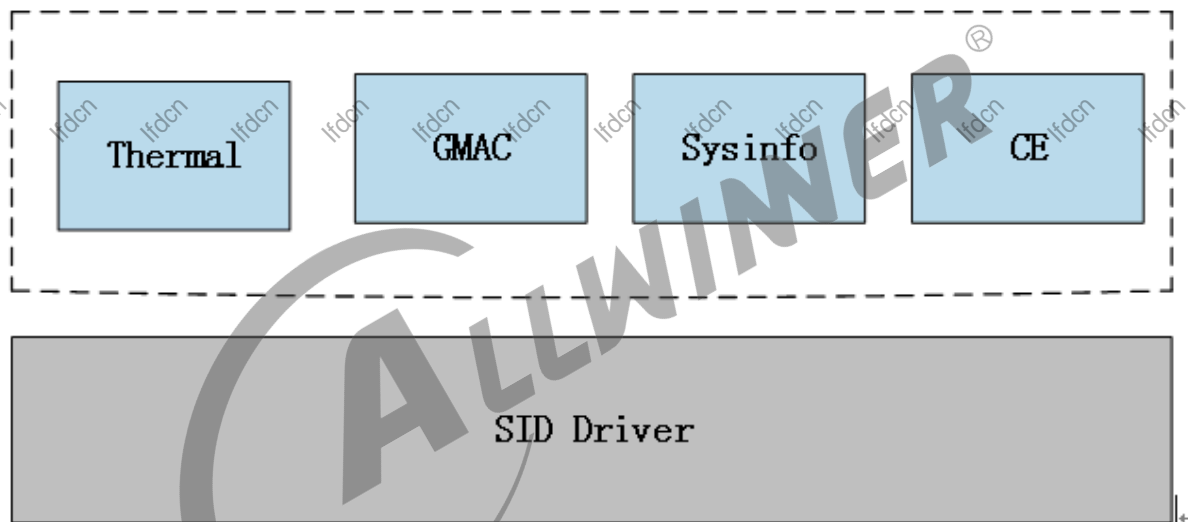


图 2-2: SID 与其他模块的关系

- 1) Thermal、GMAC 的校准参数保存在 SID 中；
- 2) CE 会用到 SID 中的一些 Key；
- 3) Sysinfo 比较特殊，为了方便用户空间获取、调试 SID 信息，专门设计的一个字符型设备驱动。

2.3 模块 device tree 配置说明

SID 模块在 Device tree 中通常会用到两个模块的配置信息：以 D1-H 为例，需要在 sun20iw1p1.dtsi 中添加节点：


```
sram_ctrl: sram_ctrl@3000000 {
    compatible = "allwinner,sram_ctrl";
    reg = <0x0 0x3000000 0 0x16C>;
    soc_ver {
        offset = <0x24>;
        mask = <0x7>;
        shift = <0>;
        ver_a = <0x18590000>;
        ver_b = <0x18590002>;
    };

    soc_id {
        offset = <0x200>;
        mask = <0x1>;
        shift = <22>;
    };

    soc_bin {
        offset = <0x0>;
        mask = <0x3ff>;
        shift = <0x0>;
    };
};

sid@3006000 {
    compatible = "allwinner,sun20iw1p1-sid", "allwinner,sunxi-sid";
    reg = <0x0 0x03006000 0 0x1000>;
    #address-cells = <1>;
    #size-cells = <1>;

    chipid {
        reg = <0x0 0>;
        offset = <0x200>;
        size = <0x10>;
    };

    secure_status {
        reg = <0x0 0>;
        offset = <0xa0>;
        size = <0x4>;
    };

    vf_table: vf-table@00 {
        reg = <0x00 2>;
    };

    ths_calib: calib@14 {
        reg = <0x14 8>;
    };
};
```

在 sid 下增加子节点 chipid, ths_calib。就可以用 key_info 来访问。

```
console:/ # echo chipid > /sys/class/sunxi_info/key_info ; cat /sys/class/sunxi_info/
key_info
console:/ # 00000400
```

2.4 kernel menuconfig 配置

Linux-5.4 内核版本执行：./build.sh menuconfig 进入配置主界面，并进入如下目录勾选对应驱动：

2.4.1 sunxi_info 的驱动的配置

进入配置 Device Drivers 界面，勾选对应驱动：

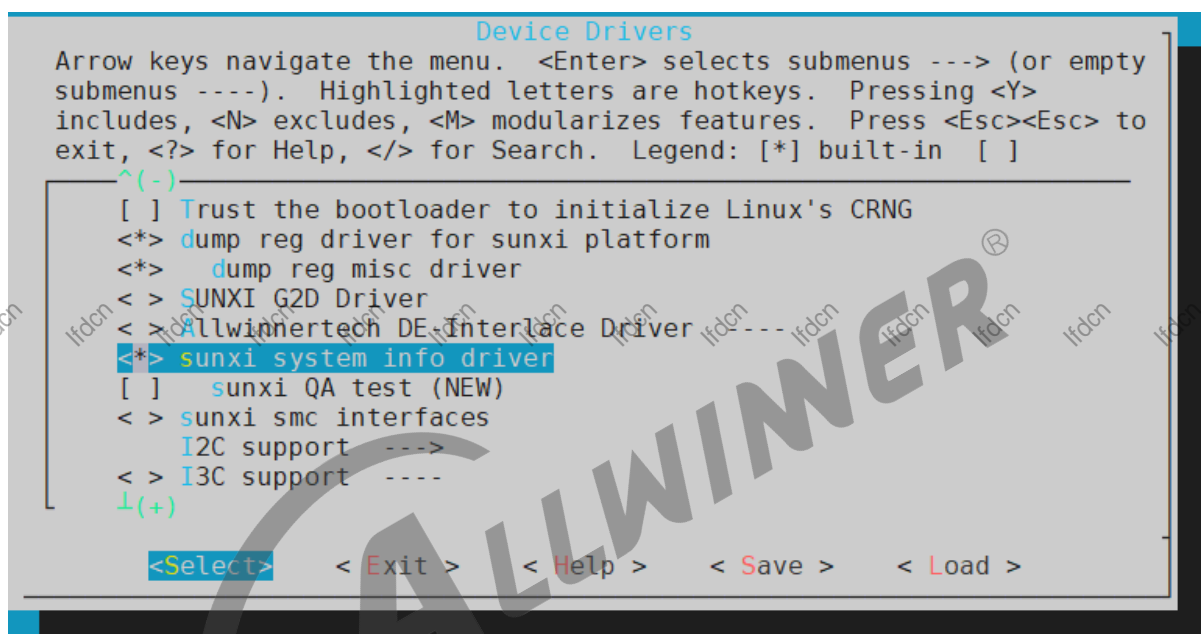


图 2-3: 内核 menuconfig 菜单

2.4.2 sid 的驱动的配置

进入配置 SOC(System On chip) sepcific Drivers 界面，勾选对应驱动：

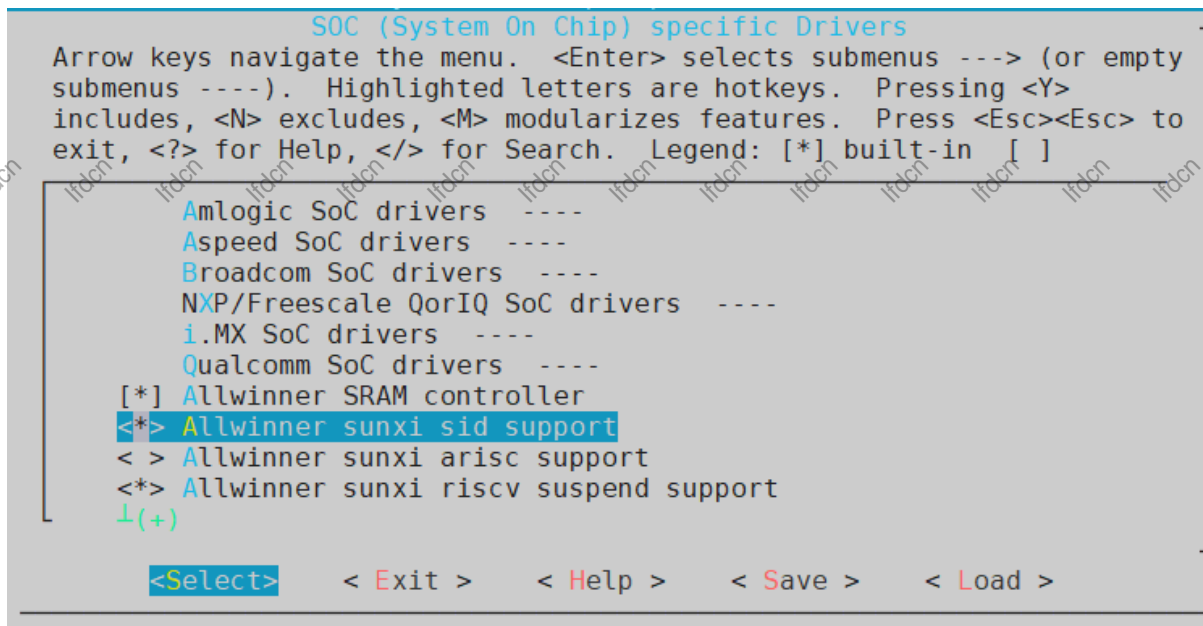


图 2-4: 内核 menuconfig 菜单

2.5 模块源码结构

SID 驱动的源代码目录下:

```
linux-5.4
./drivers/soc/sunxi/
└─ sunxi-sid.c // 实现了SID对外的所有API接口
```

对外提供的接口头文件: `./include/linux/sunxi-sid.h`

3 模块设计

3.1 结构框图

SID 驱动内部的功能划分如下图所示：

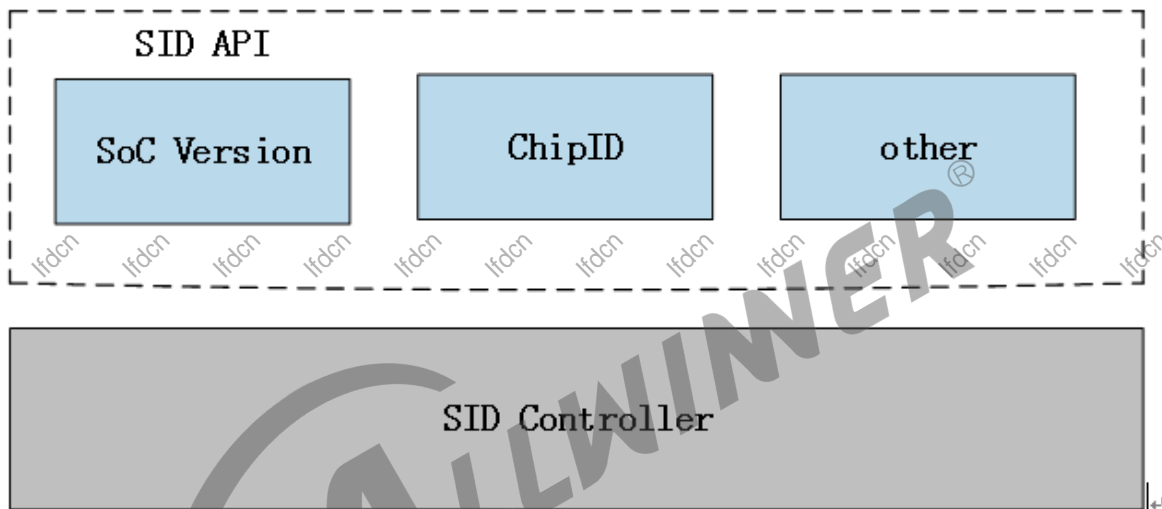


图 3-1: SID 驱动的内部结构

总体上，SID 驱动内部可以分为两大部分：

1.SID Controller，封装了对寄存器按位读取的接口，以及获取指定 compatible 的模块基地址等。

2.SID API，以 API 的方式提供一些功能接口：获取 SoC Version、获取 ChipID 等。

3.2 关键数据定义

3.2.1 常量及宏定义

3.2.1.1 key 的名称定义

在获取 Key 的时候，调用者需要知道 Key 的名称，以此作为索引的依据。Key 名称详见 sunxi-sid.h：

```
1 #define EFUSE_CHIPID_NAME "chipid"
2 #define EFUSE_BROM_CONF_NAME "brom_conf"
3 #define EFUSE_BROM_TRY_NAME "brom_try"
4 #define EFUSE_THM_SENSOR_NAME "thermal_sensor"
5 #define EFUSE_FT_ZONE_NAME "ft_zone"
6 #define EFUSE_OEM_NAME "oem"
```

说明

不是所有 **key** 都能访问，一般可以访问的已经在 **fts** 定义。

3.2.2 关键数据结构

3.2.2.1 soc_ver_map

为了管理 SoC 的 Version 信息，因此定义一个 soc_ver_map 结构，具体在 sunxi-sid.c 中：

```
#define SUNXI_VER_MAX_NUM 8
struct soc_ver_map {
    u32 id;
    u32 rev[SUNXI_VER_MAX_NUM];
};
```

结构成员变量具体如下：

- 1) id，即 BondingID。
- 2) rev[]，保存 SoC Version 的各个版本的值，比如 A 版、B 版等。

3.2.2.2 soc_ver_reg

为了记录 SoC Version、BondingID 的存储位置信息，所以定义了一个结构来记录这类信息，定义见 sunxi-sid.c：

```
#define SUNXI_SOC_ID_INDEX 1
#define SUNXI_SECURITY_ENABLE_INDEX 2
struct soc_ver_reg {
    s8 compatile[48];
    u32 offset;
    u32 mask;
    u32 shift;
    struct soc_ver_map ver_map;
};
```

结构成员变量具体如下：

- 1) compatile - 该模块的基地址。
- 2) offset - 偏移。
- 3) mask - 掩码。
- 4) shift - 位移。

5) ver_map, 保存该 SOC 的 BondingID 和 SoC Version 的各个版本的值。

3.2.3 全局变量

定义几个 static 全局变量, 用于保存解析后的 ChipID、SoC_Ver 等信息:

```
static unsigned int sunxi_soc_chipid[4];  
static unsigned int sunxi_serial[4];  
static unsigned int sunxi_soc_bin;  
static unsigned int sunxi_soc_ver;
```

3.3 模块流程设计

3.3.1 SoC 信息读取流程

本节中, 这里把 SoC Ver、ChipID 信息统称为 “SoC 信息”, 因为他们的读取过程非常相似。都是遵循以下流程:

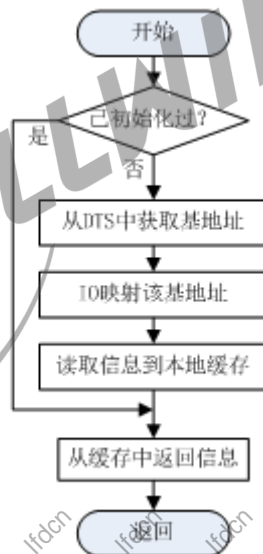


图 3-2: SoC 信息读取流程

4 接口设计

4.1 接口函数

4.1.1 sunxi_get_platform()

- 作用：获取 SoC 平台的名称，实际上是一个 BSP 研发代号，如 sun20iw1。
- 参数：
 - buf: 用于保存平台名称的缓冲区
 - size: buf 的大小
- 返回：
 - 返回 buf 中平台名称的实际拷贝长度（如果 size 小于名称长度，返回 size）。

4.1.2 sunxi_get_soc_chipid()

- 作用：获取 SoC 的 ChipID（从 Efuse 中读到的原始内容，包括数据内容和顺序）。
- 参数：
 - chipid: 用于保存 ChipID 的缓冲区
- 返回：
 - 会返回 0，无实际意义

4.1.3 sunxi_get_serial()

- 作用：获取 SoC 的序列号。
- 参数：
 - serial: 用于保存序列号的缓冲区
- 返回：
 - 会返回 0，无实际意义

说明

Serial num 就是 **ChipID** 的顺序调换的组合值。

4.1.4 sunxi_get_soc_chipid_str()

- 作用：获取 SoC 的 ChipID 的第一个字节，并且转化为 8 个字节的字符串。

- 参数：

- serial：至少要有 8 个字节长度的 buf

- 返回：

- 只会返回 8（4 个字节的十六进制打印长度），无实际意义

4.1.5 sunxi_get_soc_ver()

- 作用：获取 SoC 的版本信息。

- 参数：

- 无

- 返回：

- 返回一个十六进制的编号，需要调用者去判断版本号然后做出相应的处理。详情参看 dts，sid 节点。

4.2 内部函数

4.2.1 sunxi_get_base()

- 作用：从 DTS 中获取指定模块的寄存器基地址。

- 参数：

- pnode：用于保存获取到的模块 node 信息
 - base：用于保存获取到的寄存器基地址
 - compatible：模块名称，用于匹配 DTS 中的模块

- 返回：

- 0: success
 - other: fail

4.2.2 sunxi_put_base()

- 作用：释放一个模块的基地址。

- 参数：

- pnode: 保存模块 node 信息
- base: 该模块的寄存器基地址
- 返回:
 - 无

4.2.3 sid_rd_bits()

- 作用: 从一个模块的寄存器中, 读取指定位置的 bit 信息。
- 参数:
 - name: 模块名称, 用于匹配 DTS 中的模块
 - offset: 寄存器相当于基地址的偏移
 - shift: 该 bit 在寄存器中的位移
 - mask: 该 bit 的掩码值
- 返回:
 - 0: fail
 - other: 获取到的实际 bit 信息

5 可测试性

5.1 sysfs 调试接口

SID 驱动本身没有注册为单独的模块，需要通过注册 sysinfo 字符驱动（实现代码见 drivers/char/sunxi-sysinfo/）来提供 sysfs 节点。

1. /sys/class/sunxi_info/sys_info

此节点文件可以打印出一些 SoC 信息，包括版本信息、ChipID 等：

```
# cat /sys/class/sunxi_info/sys_info
sunxi_platform : sun20iw1p1
sunxi_chipid   : 00000000000000000000000000000000
sunxi_chiptype : 00000400
sunxi_batchno  : 0x1
```

著作权声明

版权所有 © 2022 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明

、 全志科技、（不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。