# SDIO设备驱动

## 1. MMC子系统

### 1.1 MMC子系统结构

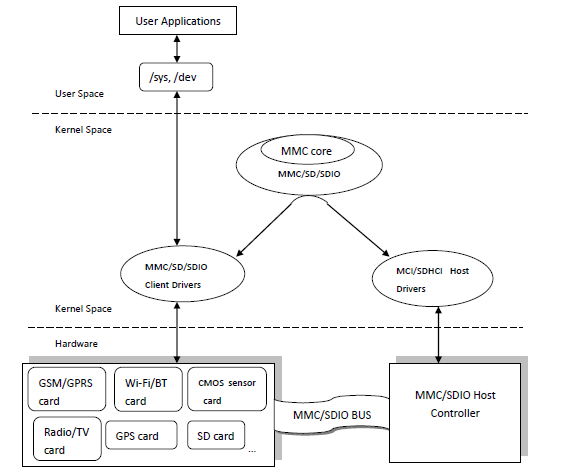
MMC子系统代码主要在drivers/mmc目录下，共有三个目录：

**Card：**存放闪存卡(块设备)的相关驱动，如MMC/SD卡设备驱动，SDIOUART；

**Host：**针对不同主机端的SDHC、MMC控制器的驱动，这部分需要由驱动工程师来完成；

**Core：**整个MMC的核心层，这部分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子系统框架:**



[**Linux**](http://lib.csdn.net/base/linux) MMC子系统主要分成三个部分：

**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**：针对不同主机端的SDHC、MMC控制器的驱动；

**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。

### 1.1.1 Core层初始化

core层以subsys\_initcall(mmc\_init)做初始化

static int \_\_init mmc\_init(void)

{

ret = mmc\_register\_bus();

…..

ret = sdio\_register\_bus();

}

### 1.1.2 Host层初始化

选一个SD类的Host驱动为例

module\_platform\_driver(sdhci\_s3c\_driver);



mmc\_attach\_sdio函数中主要做如下事情:

1. 发送命令探测SDIO设备

2. 配置工作电压

3. 探测和初始化card,获取card的类型,操作cccr,配置card

4. 配置并初始化card的各个function

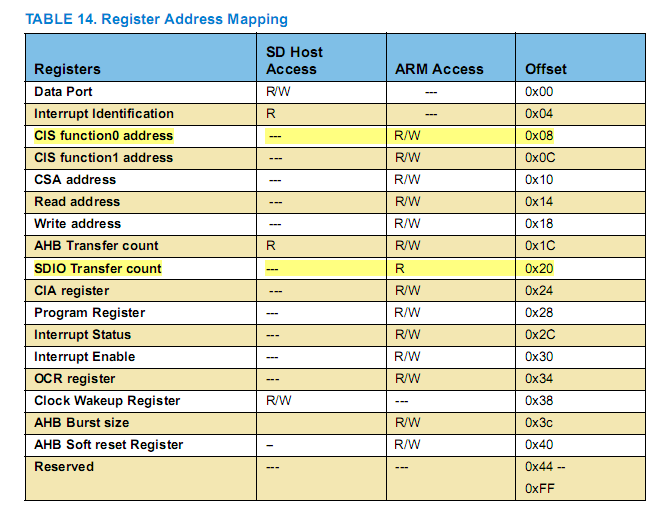
5. 注册卡设备到驱动模型中去(mmc\_bus\_type),注册卡各个function到设备驱动模型中(sdio\_bus\_type)

关于function 补充一下:

function0:



function1:



## 2 sdio client设备驱动结构

要完成一个sdio接口下的网口设备驱动开发,首先它是一个sdio驱动,然后再是一个网口设备驱动,在sdio驱动的包装下完成网口的功能

module\_init(if\_sdio\_init\_module);

模块加载下调用if\_sdio\_probe,该设备驱动匹配id\_table具体的设备(如果已经挂载在总线上的话),以marval公司WIFI芯片8686系列驱动为例说明在if\_sdio\_probe做了哪些事情,下面绘图基本上描述了本驱动的结构.



## 3. 设备驱动与device之间数据收发

### 3.2 Host端中断数据接收和数据发送

### 3.1 Host端对device的初始化操作

Host端对device端进行初始化操作,包括如下:

1. 查询和设备工作电压(CMD5)

2. 获取RCA值(card address)(CMD3)

3. CMD7进入传输模式

在进入传输模式前,一般需要对CIA寄存器组进行设置

⑴使能I/O： 对地址0x02的寄存器写0x02

⑵ 检查功能是否ready：读地址0x03的寄存器的bit1

⑶使能中断：对地址0x04的寄存器写0x03,只有这样，device的中断才能发送到host

#### 3.2.1 Host端数据接收

在前面申明一个 sdio\_claim\_irq(func, if\_sdio\_interrupt),sdio\_claim\_irq函数创建了一个内核线程专门用来不断读取中断PEND寄存器,根据PEND寄存器哪位设置了标记,然后调用回调函数if\_sdio\_interrupt.而这个PEND寄存器需要ARM配置设备function1中的Program Register中的fun1 read dataready位.ARM还需要设置数据长度寄存器告知Host应该读取多长的数据

#### 3.2.1 Host端数据发送

Host端数据发送比较绕, 网卡数据函数为 .ndo\_start\_xmit = lbs\_hard\_start\_xmit, 然后将skb网络数据保存在priv结构中去唤醒之前创建的内核线程处理,该内核线程主要处理固件发来的事情,命令,以及ndo\_start\_xmit函数发送的数据. 由于我们自己处理没有固件,故没有事件处理和命令处理,故打算将数据发送直接放到ndo\_start\_xmit回调函数中.

下面来看看数据是如何发送的:



### 3.3 ARM对device操作

#### 3.3.1 ARM发送数据到SD Host

(1)向AHB Transfer count寄存器写入将要传输的字节数

(2)向Program Register寄存器位fun1 read dataready写入,当然两个寄存器Read address Write address 需要提前配置OK

(3)AHB bridge作为代理host读取完数据后向ARM发送中断,SD host通过CMD52,CMD53从rd\_fifo读取数据

#### 3.3.1 ARM从SD Host读取数据

(1)SD Host通过CMD52,CMD53命令发送数据.

(2)AHB bridge作为代理Host向write addr写数据,完成后向interrupt Status的write\_tran\_over bit 告知写完

(3)ARM读取interrupt Status及SDIO transfer count register获取数据大小