Flash 研究报告

SWD 聂勇

版 本 历 史

版本/状态	责任人	起止日期	备注
V1.0/草稿	聂勇	4Jan2011	创建文档

目　录

1. 说明

本来打算将 Flash 研究方面的资料放到 redmine 上，不单独建立文档。但是考虑到 redmine 的不连贯性和不容易查找，特撰写本文档。

2. Flash Wiki

关于 Flash 的基本概念和知识，请参考 Flash wiki。

下面就几个用到的比较重要的概念作出解释。

block：块，在 NOR Flash 中进行擦除的单位就是 block，也就是 Erase 操作一次必须擦除一个 block，而不能够是一个字节，几个字节等。block 的大小根据芯片的不同会有变化，常见的例如 128Kbytes，64Kbytes。

3. Flash 接口电路分析

NOR Flash 接口电路，是 MCU 和 NOR Flash 之间的电路连接。通过了解实际电路，能够深入了解很多相关的概念，有助于底层驱动的编写。可以深入了解的内容有：NOR Flash 的随机存储，NOR Flash 的芯片内执行（XIP，eXecute In Place），NOR Flash 何时不需要驱动，NOR Flash 的驱动需要怎么设计，双字节/4 字节地址对齐，NAND Flash 采用 I/O 口设计等概念。

具体的 Flash 接口电路参考硬件部门的文档《TP-Link 硬件工程师培训教材 2010 版.pdf》。

4. Flash 的操作

Flash 作为存储设备，基本上只需要两个操作，读操作和写操作。但是由于 Flash 的特殊性，写操作又有两个：Erase 和 Program。

在开始具体的操作之前，首先要了解的就是一张 Flash Command 表。这张表规定了在开始某一种操作之前，首先要发给 Flash 芯片的命令。例如，要读取 Flash 芯片中某一个地址的一个字节的数据，首先得告诉 Flash 芯片，下面要进行 Flash 的读取了。那怎么做到呢，那么就向 Flash 芯片的基地址处写入一个规定的字节，例如规定 0XFF 表示读操作。只有发送了这样一个命令之后，Flash 芯片才会切换到读的操作。

下面这个表格就是 Intel® Embedded Flash Memory (J3 v. D) device 的命令表。各个不同的芯片型号对应的命令表可能会有区别。

Table 31. Command Bus Operations for

Command		Setup Write Cycle		Confirm Write Cycle	
		Address Bus	Data Bus	Address Bus	Data Bus
Registers	Program Enhanced Configuration Register	Register Data	0060h	Register Data	0004h
	Program OTP Register	Device Address	00C0h	Register Offset	Register Data
	Clear Status Register	Device Address	0050h	---	---
	Program STS Configuration Register	Device Address	00B8h	Device Address	Register Data
Read Modes	Read Array	Device Address	00FFh	---	---
	Read Status Register	Device Address	0070h	---	---
	Read Identifier Codes (Read Device Information)	Device Address	0090h	---	---
	CFI Query	Device Address	0098h	---	---
Program and Erase	Word/Byte Program	Device Address	0040h/ 0010h	Device Address	Array Data
	Buffered Program	Word Address	00E8h	Device Address	00D0h
	Block Erase	Block Address	0020h	Block Address	00D0h
	Program/Erase Suspend	Device Address	00B0h	---	---
	Program/Erase Resume	Device Address	00D0h	---	---
Security	Lock Block	Block Address	0060h	Block Address	0001h
	Unlock Block	Device Address	0060h	Device Address	00D0h

4.1 Read Operation

读操作在很多文档中也叫做读模式，read mode。一般来说，读操作获得的数据有 4 种。分别是 Read Array，Read Status Register，Read ID，CFI Query。其中，Read Array 就是获得存储在 Flash 中的用户数据，例如代码，文件等等。

下面以 Read Array 操作，来说明对 Flash 中数据的读取过程。代码如下：

flash28f640DrvLib.c

```
LOCAL int flashRead(int sectorNum, char *buff, unsigned int offset, unsigned int count)
{
    flashReadReset();
    if (sectorNum < 0 || sectorNum >= flashSectorCount)
    {
        printf("flashRead(): Illegal sector %d\n", sectorNum);
        return (ERROR);
    }

    bcopy((char *)(FLASH_SECTOR_ADDRESS(sectorNum) + offset), buff, count);

    return (0);
}
```

flashReadReset()函数：Read Array Command，就是告诉 Flash 芯片，下面进入了对存储数据的读操作。

bcopy(……)函数：真正的对 Flash 中数据的读操作。可以看出，其操作方式和对内存中（SDRAM）数据的操作时一样的，都是使用的 bcopy 函数。(char *)(FLASH_SECTOR_ADDRESS(sectorNum) + offset)参数对应的就是数据在 Flash 中的位置，或者说是地址。

4.2 Write Operation

这是因为 Flash 的存储单元中，将比特 1 变为比特 0 和将比特 0 变为比特 1 是两种不同的操作机制。其中，Erase 操作就是将存储单元的比特 0 变为比特 1，并且，一次只能够以一个 block 来操作（一个 block 为 128K 或者 64K，器件不同会有差别）。而 Program 操作就是将存储单元的比特 1 变成比特 0，反之就不行。例如，可以将 FF 变成任意的 8bit 值，也可以将 0xA3 变成 0xA2，再变成 0xA0，但不能变成 0xA1 哦。

5. Flash 安全和保护

对 Flash 中数据的安全和保护，主要有软件和硬件两种方式，每种芯片支持的可能有些不同，或者甚至不支持某一些操作。下面以 Intel® Embedded Flash Memory (J3 v. D) device 为例说明其保护的机制。

5.1 lock&unlock

Flash 支持对每一个 block 上锁，被上锁的 block 不能够进行擦除操作和编程操作。并且，在 Flash 芯片复位或者断电重启之后，上锁状态不会丢失。

Flash 支持对整片芯片进行 unlock 解锁。

这个功能有什么作用呢？这个可以在防止程序修改某些 block 的内容。例如，存放代码的 block，在每一次烧写成功之后，要进行上锁操作。这样就可以防止这些代码被修改。又例如，某些只读的配置文件，可以对其所在的 block 进行上锁，这样就可以防止程序修改其中的内容。

存在的疑惑：解锁时整片芯片，上锁是针对一个 block。如果一个需要上锁的 block 更新，必须先解锁（整片 Flash 芯片），在更新该 block 之后再上锁（该 block 上锁）。这样其他的原来上锁的 block 不就处在 unlock 状态了吗？

5.2 OTP Protection Registers

该寄存器为 128 比特，其中前 64 比特为芯片唯一序列号，只读，后 64 比特为用户可读写。可以利用此来进行 Flash ID 加密。

5.3 VPP/VPEN 引脚

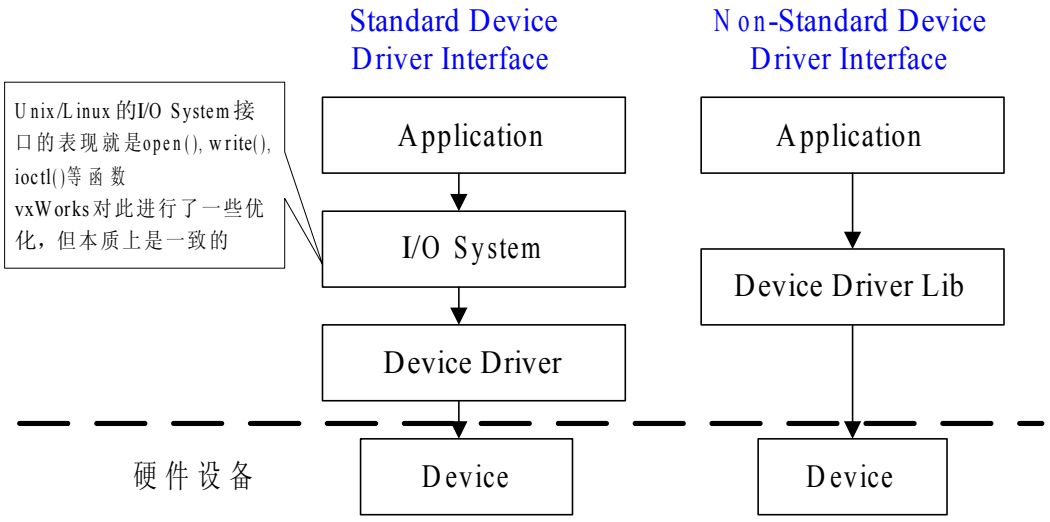
可以在硬件设计时利用这些引脚电平对整片 Flash 芯片来进行保护。

6. NOR Flash 驱动

6.1 Flash 驱动与文件系统

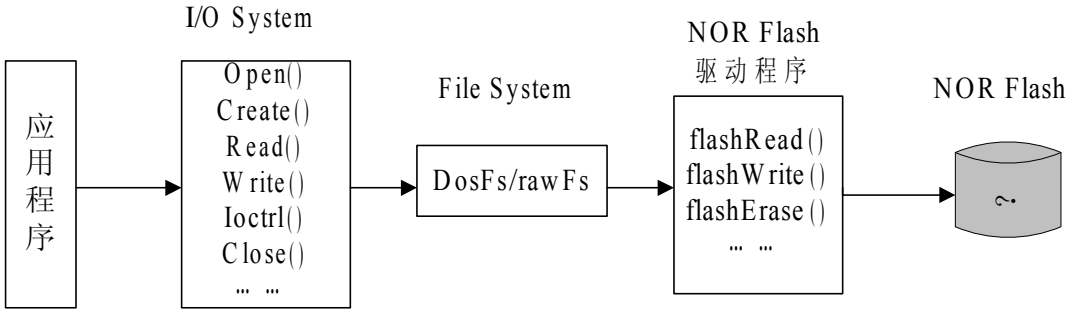
首先，简要的介绍一下嵌入式系统中设备驱动接口的概念。我们将从整体上来把握，就像从飞机上来俯瞰一样，这样能够让我们更加明白自己所在的位置和角色。

设备驱动接口分为两种，一种是标准的设备驱动接口，另外一种为非标准的，如下图所示。区别在于，标准的设备驱动接口的符合操作系统中队设备进行操作的相关流程，而非标准接口由于应用程序直接和驱动程序联系，绕过 OS 中的 I/O 系统，在效率，实时性等方面更强。



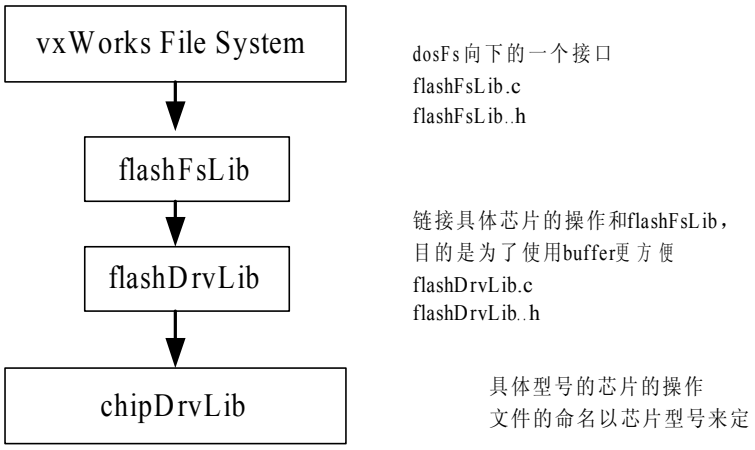
以上是一个总体的设备驱动接口的框图。本文档中需要分析的是 Device Driver 或者 Device Driver Lib 在 NOR Flash 这个设备是如何实现的。

对于 NOR Flash 这样的设备，由于其中还加入了文件系统一层，其总体的框图又有所变化。



通过上图可以看出，文件系统就是接在了 I/O 系统和 NOR Flash 驱动程序之间。那么，如果使用了文件系统，那么 NOR Flash 中就是存放的文件，目录等。如果没文件系统，设计的 NOR Flash 驱动程序也能够对 NOR Flash 中的任何地址，数据进行操作。例如在 bootload 阶段没有使用文件系统，也需要读取 NOR Flash 中的配置信息等等。

在 MIPS BSP 中，Flash 驱动采用模块化设计，各模块的关系如下图所示。



6.2 chipDrvLib 和 flashDrvLib

chipDrvLib 是实际操作 NOR Flash 的寄存器。在这部分的实现过程中，需要反复参考第四节-Flash 的操作中的内容。你暂时可以将 I/O 系统，文件系统的知识都抛掷脑后，它们和我们现在要做的一点关系都没有。

下面为对 Flash 地址的操作宏。在 chipDrvLib 中反复使用到该宏。

flash28f640DrvLib.c

```
#define FLASH_ADDR(dev, addr) \
    ((volatile UINT8 *) (((int)(dev) + (int)(addr)) ^ xor_val))
```

```
#define FLASH_WRITE(dev, addr, value) \
    (*FLASH_ADDR(dev, addr) = (value))

#define FLASH_READ(dev, addr) \
    (*FLASH_ADDR(dev, addr))
```

6.3 flashFsLib

本节涉及到文件系统的相关知识点。要注意总结。

7. upgrade 的实现

交换机软件版本的升级是本节研究的内容，具体包括交换机软件升级的过程，升级软件包的制作两部分。

7.1 upgrade 过程

首先，明确两个概念

启动型 BOOTROM：也就是最终发布的软件中带的 BOOTROM，该 BOOTROM 的命令不是原来 BCM 公司提供的那一套命令，而是自己实现的。

下载型 BOOTROM：BCM 提供的 BOOTROM，最重要的差别在于命令行是使用的原来的一套。

例如，需要改变 FTP 的参数，启动型 BOOTROM 使用的是命令#ftp，而下载型 BOOTROM 使用的是#c 命令来修改，#p 命令来打印。

在启动型的 BOOTROM 中，添加了升级系统软件的命令 upgrade。所以，如果要使用升级软件包升级软件，则需要运行的是启动型的 BOOTROM。在出厂的产品中，都是使用的启动型的 BOOTROM，而在开发调试过程中，大部分使用的是下载型的 BOOTROM，至于这两者的切换，请参考下面的说明。

文件说明：

hawkeye_bootrom_20101008.bin	第一版启动型 BOOTROM
hawkeye_bootrom_20101029.bin	第二版启动型 BOOTROM
hawkeye_bootrom_20101029_vxWorks	启动型 BOOTROM 的 VXWORKS
hawkeye_bootrom_download.bin	下载型 BOOTROOM 镜像
hawkeye_bootrom_20101230_release.bin	发本版本启动型 BOOTROM

由启动项 BOOTROM 升级为下载型 BOOTROM 方法：

在 BOOTROM 菜单中输入 bootrom hawkeye_bootrom_download.bin 命令后就可以将其写入到 FLASH 中，重启后就可以正常下载 vxwoks 了。

由下载型 bootrom 升级为启动型 bootrom 方法：

将 hawkeye_bootrom_20101029_vxWorks 下载到内存中，然后进入 BOOTROM 菜单中，输入 bootrom hawkeye_bootrom_20101029.bin 命令就可以升级为启动型 BOOTROM 了。

在进入 BOOTROM 的命令行下，首先指定 FTP 下的 IP 地址，用户，密码和升级文件，例如：

#ftp host 192.168.0.79 user hawkeye pwd 123 file up.bin

这就是表示，从主机 192.168.0.79 上通过 FTP 下载升级文件 up.bin 对交换机进行升级。其中，FTP 的用户名为 hawkeye，密码为 123。

然后，输入系统升级命令 upgrade

#upgrade

该命令就会通过交换机的端口 1 链接主机的 FTP，下载升级文件，解析升级文件，然后将更新的内容烧入 FLASH 中。下载升级文件在 ET 网络驱动部分有详细的讲解，将更新内容写入 FLASH 是本文上面的 FLASH 驱动部分实现的。这里最重要的是对升级文件的解析。

7.2 up.bin 解析

在开始 up.bin 文件解析之前，首先要了解 up.bin 文件中内容的组织结构。有关内容的组织结构，请参考韦晟敢编写的文档《Flash 升级文件布局.doc》。

7.3 up.bin 制作

制作升级文件步骤：

1、如果机器为从来没有升级过的话，那么请先修改机器的产品类型，具体可以通过启动型 BOOTROM 菜单中的 product 命令进行修改

2、创建相应的机型配置文件***.tpcfg 和相应的文件夹，具体可以参考已有的机型例子。.tpcfg 文件格式含义可以参考下面部分解析。升级文件中的 profile 文件，switchTp 文件，webImage 文件和 vxworks_compress_bin 中的任意一个或几个都可以包含在升级文件中，即可以对任意一个进行升级。这里有两个注意点：

1)item 项中的 producutId 项中的 includeUp 必须置为 1

2)如果为初次升级的话，那么请将 profile 文件，switchTp 文件，webImage 文件和 vxworks_compress_bin 项中的 includeUp 都置为 1

8. Problems

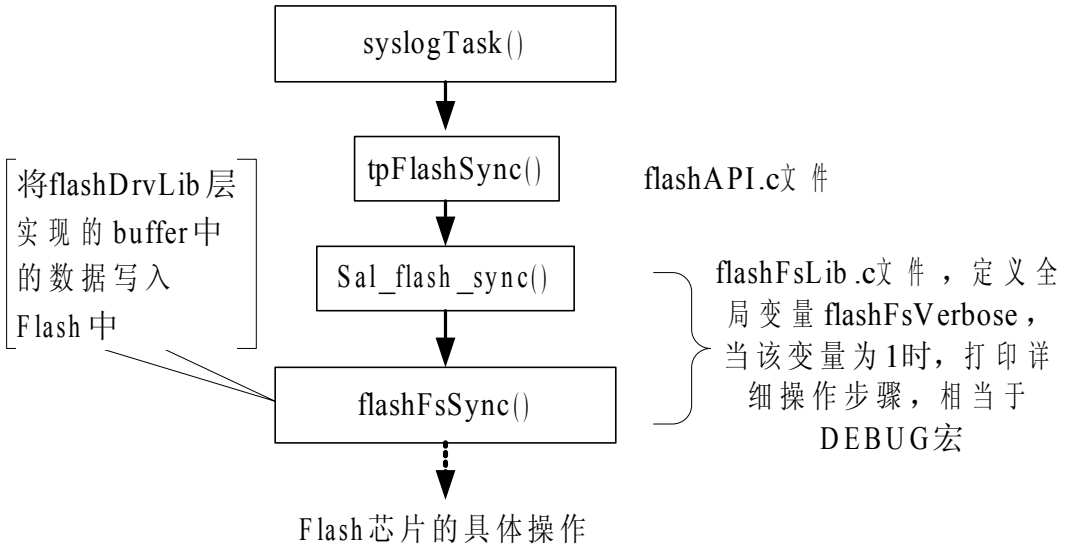
8.1 反复打印 flashFsSync()问题

2011.1.17

xCat 平台 SDK：CPSS-3.4P1

问题描述：打开交换机的网页之后，在终端（串口）反复打印信息：flashFsSync()…… done。

问题原因：syslogTask 在反复的写 Flash，其函数的调用关系如下：



出现打印信息的原因是在 flashFsLib 中的调试用全局变量被置位 1。当然，syslogTask 任务反复擦写 Flash 也是不正常的，但是不在我们这里讨论的范围之内。