A10/A20 Bootloader加载过程分析

注：由于全志A10和A20在加载Bootloader过程方面基本一致，下面仅以A20叙述，但同时也适用于A10。另外在不需要区分Cubieboard1和Cubieboard2的情况下，统称为Cubieboard；另现在市面上一般所说的SD卡即为Micro SD Card，也就是TF卡，为区别于一般传统的SD卡，本文一般使用TF卡描述，但同于平时所说的SD卡。

A20的启动过程大概可分为5步：Boot ROM，SPL，Uboot，Kernel，RootFileSystem。本文关注的是镜像的加载和启动过程，分析Boot ROM→SPL→Uboot→Kernel的启动流程。

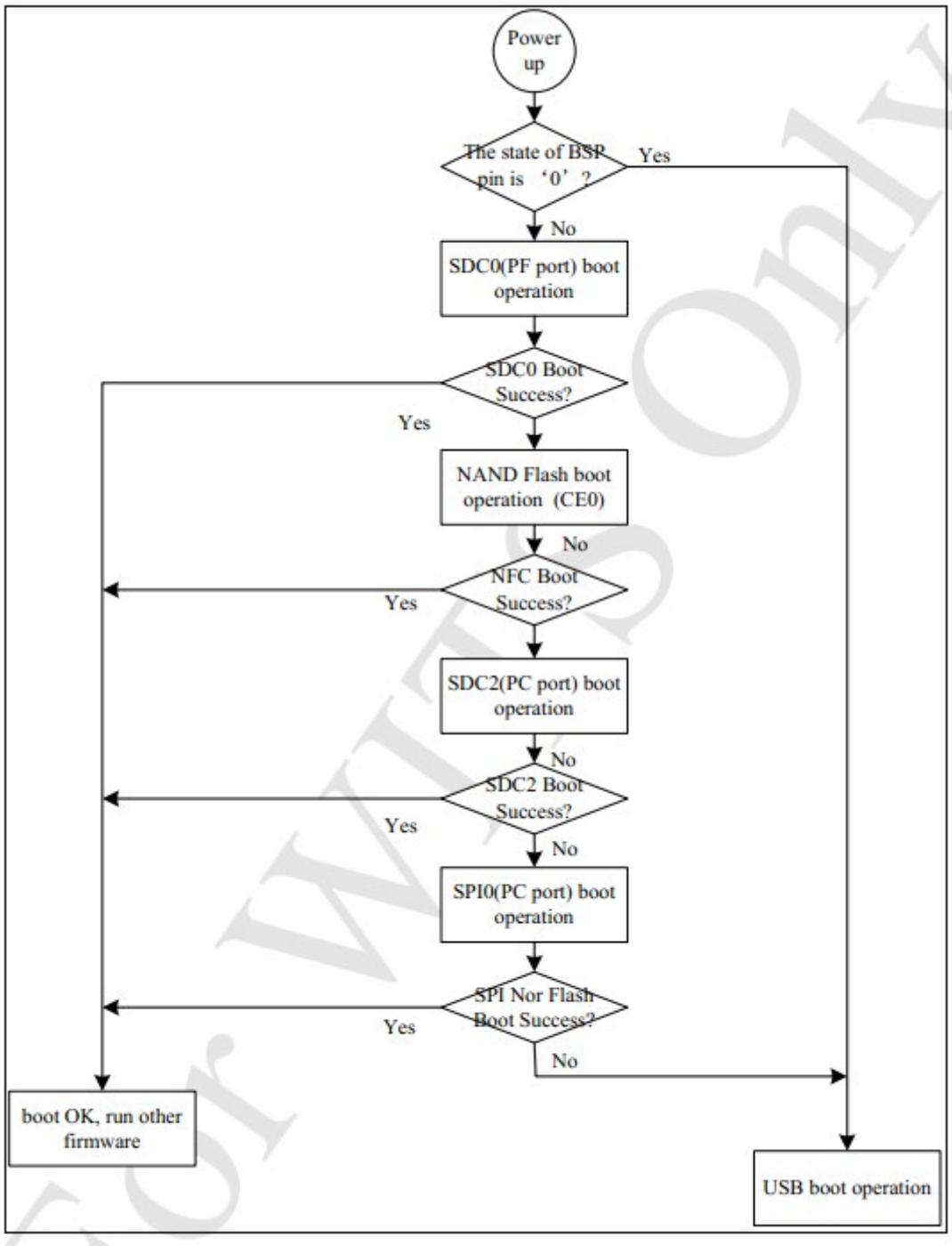
系统上电后，ARM处理器在复位时从地址0x000000开始执行指令，把板上ROM或Flash映射到这一地址。A20将启动设备选择程序固化在CPU内部的一个32KB ROM中，默认的启动时序为SD Card0，NAND FLASH，SD Card2，SPI NOR FLASH。另外通过外部的一个启动选择引脚可以使其跳转到USB启动模式。通常情况下，启动选择引脚状态连接50K内部上拉电阻。在上电后，执行存储在Boot ROM中的启动代码，将自动检测启动选择引脚状态。只有当该引脚状态为低电平时才选择USB启动模式。

在选择启动设备后将加载并执行bootloader程序，CPU通过拷贝或映射bootloader程序到内存，然后执行bootloader的第一条指令。通过阅读官方的uboot烧写方法，发现A20通过uboot引导系统之前先载入了uboot SPL。什么是SPL？通过查阅uboot的官网资料得知，SPL是一个迷你版的uboot，全拼为Second Program Loader。适用于SOC的内部SRAM<64K的情况，用它来加载完整的uboot程序到SDRAM，并通过完整uboot加载内核来启动系统。其中SRAM一般指CPU内部的L1/L2或外部的L2高速缓存，这里即为Boot ROM，而SDRAM一般指内存。

SPL程序流程如下：

1. 初始化ARM处理器
2. 初始化串口控制台
3. 配置时钟和最基础的分频
4. 初始化SDRAM
5. 配置引脚多路复用功能
6. 启动设备初始化（即上面选择的启动设备）
7. 加载完整的uboot程序并转交控制权

启动设备选择程序的流程图：



搞清楚了上面的概念，可以知道Cubieboard出厂已经烧写了NandFlash中的程序，即在启动时使用的是NandFlash。现在根据全志A20上述步骤，我们尝试用SD Card0（即Cubieboard上卡槽中的TF卡）来启动系统。

环境准备

本文所使用的主机环境为kubuntu 12.10，然而一般情况下，下面涉及到的命令对基于Debian的(X)ubuntu系列都应该适用。

为不引起混淆，我们作如下约定：

- 工作目录为 $WORK_DIR
- 命令均以root用户执行

笔者的设定如下:

```
WORK_DIR=/home/itviewer/src
```

下载必须的工具软件

```
apt-get install build-essential libncurses5-dev u-boot-tools
qemu-user-static debootstrap git binfmt-support libusb-1.0-0-dev pkg-config
apt-get install gcc-arm-linux-gnueabihf
```

下载源码

从 github 下载 SPL,U-BOOT,Linux 内核源码。注意 linux-sunxi 超过 3.8G ,耗时最长。如果您曾经下载过这些 代码,记得分别用 git pull 更新后再进行后续操作,因为代码仓库每天都有变化。

```
cd $WORK_DIR
git clone git://github.com/linux-sunxi/u-boot-sunxi.git
git clone git://github.com/cubieboard2/linux-sunxi
git clone git://github.com/linux-sunxi/sunxi-tools.git
git clone git://github.com/linux-sunxi/sunxi-boards.git
```

编译uboot

```
cd $WORK_DIR/u-boot-sunxi
make distclean CROSS_COMPILE=arm-linux-gnueabihf-
make cubieboard2 CROSS_COMPILE=arm-linux-gnueabihf-
```

得到 u-boot-sunxi-with-spl.bin（同时生成的还有其它几个文件，这里我们只用该文件）

编译kernel

```
cd $WORK_DIR/linux-sunxi
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- cubieboard2_defconfig
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j4 uImage modules
```

得到内核文件 arch/arm/boot/uImage（还有其它内核模块，这里暂时没有用到）

生成 boot.scr和script.bin

在将uboot和kernel烧写到TF卡之前，我们需要先编译生成两个启动参数文件：boot.scr和script.bin。

生成 boot.scr

boot.scr是什么？

根据资料描述（https://github.com/linux-sunxi/u-boot-sunxi/wiki#bootscr-support），u-boot在启动的时候会在第一个分区（FAT/extX格式）寻找/boot.scr或者/boot/boot.scr文件，boot.scr中可以包含用于载入script.bin，kernel，initrd（可选）以及设置内核启动参数的uboot命令。

boot.scr如何生成？

在$WORK_DIR目录新建 boot.cmd 文件，添加以下内容：

```
setenv bootargs console=ttyS0,115200 noinitrd disp.screen0_output_mode=EDID:
1280x1024p60 init=/init root=/dev/mmcblk0p2 rootfstype=ext4 rootwait panic=
10 ${extra}
fatload mmc 0 0x43000000 boot/script.bin
fatload mmc 0 0x48000000 boot/uImage
bootm 0x48000000
```

详细解释：

1.上述第一行设置uboot的bootargs启动参数，格式为 参数=值，不同参数使用空格分开，其中

- console=ttyS0,115200 含义为使用特定的串口ttyS0，波特率为 115200
- noinitrd 含义为不使用ramdisk（内存磁盘）
- init=/init 含义为内核启起来后，进入系统中运行的第一个脚本
- root=/dev/mmcblk0p2 含义为指定rootfs的位置为TF卡第二个分区
- rootfstype=ext4 含义为根文件系统类型
- rootwait 含义为等待设备/dev/mmcblk0p2就绪后才尝试挂载rootfs
- panic=10 传递内核参数，当遇到panic（内核严重错误）时等待10秒后重启
- screen0_output_mode 设置合适的屏幕显示分辨率

更多的参数可以通过查看Linux内核源码目录下Documentation/kernel-parameters.txt文件了解

2.第二行和第三行为将script.bin和内核uImage加载到指定内存地址。fatload是U-Boot中装载linux kernel到内存的指令。

基本用法：fatload <interface> <dev[:part]> <addr> <filename> <bytes>

- interface：所用到接口，如：MMC、USB

- dev [:part]: 文件存放的设备 如：ide 0:1
- addr: 装载到内存的开始地址。
- filename: 装载的文件名称。
- bytes: copy的字节数.

3.第四行bootm 用于将内核映像加载到指定的地址

保存文件后，执行以下命令生成boot.scr：

```
mkimage -C none -A arm -T script -d boot.cmd boot.scr
```

生成 script.bin

script.bin是什么？

script.bin是被全志SOC内核驱动或LiveSuit使用的针对特定目标板的二进制配置文件，包含如何设置基于A10/A20目标版的各种外设，端口，I/O针脚信息。

其对应的可读文本文件格式为FEX，可以利用 Sunxi-tools在二进制和文本文件之间进行转换。更多关于FEX配置的信息可以参考http://linux-sunxi.org/Fex_Guide

script.bin如何生成？

首先需要编译sunxi-tools：

```
cd $WORK_DIR/sunxi-tools
make
```

得到fex2bin、bin2fex等文件，其中fex2bin能把 *.fex 文件生成 *.bin文件。反之bin2fex可以将得到的*.bin文件生成可读的*.fex文件。

然后编译生成script.bin：

```
cd $WORK_DIR/sunxi-boards/sys_config/a20
$WORK_DIR/sunxi-tools/fex2bin cubieboard2.fex script.bin
```

烧写uboot和kernel到TF卡

上面罗嗦了这么多，其实就是为了将uboot和kernel烧写到TF卡上并能够启动，OK，让我们先从分区开始：

A20 芯片上电启动的时候，会读取SD卡最前面的 1M 内容，从而得到 bootloader，所以我们需要把 u-boot 写到SD卡的前1M区间。

其中详细的SD卡布局如下：

起始	大小	用途
0	8KB	存放分区表等内容
8	24KB	SPL loader
32	512KB	u-boot
544	128KB	environment
672	352KB	保留
1024	-	用于剩余分区

接下来，我们开始使用fdisk进行分区（由于sfdisk对部分TF不兼容，故除非你真的知道怎么用sfdisk，否则不要使用）：

将TF卡插到电脑上并确认设备名，为不至于混淆，我们使用sdX代替，您需要根据自己的情况修改，如sdb：

```
card=/dev/sdX
dd if=/dev/zero of=${card} bs=1M count=1        # 把SD卡前1M的区域填充为0，预留给u-boot
sfdisk -R ${card}                               # 重新读取${card}
fdisk ${card}                                   #使用fdisk进行分区
```

具体分区步骤如下：

建立第一个分区

```
root@kubuntu:~/src/u-boot-sunxi# fdisk ${card}
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0x911332e8.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help): n #键入n然后回车
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p):             #直接回车
Using default response p
Partition number (1-4, default 1):          #直接回车
Using default value 1
First sector (2048-15278079, default 2048):      #直接回车
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-15278079, default 15278079):
+64M #键入+64M后回车，即分区大小为64M
```

建立第二个分区

```
Command (m for help): n #键入n然后回车
Partition type:
```

```
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p):              #直接回车
Using default response p
Partition number (1-4, default 2):          #直接回车
Using default value 2
First sector (133120-15278079, default 133120):     #直接回车
Using default value 133120
Last sector, +sectors or +size{K,M,G} (133120-15278079, default 15278079):  #直接回车，即第二个分区使用全部剩余空间
Using default value 15278079
```

接下来指定分区类型：

```
Command (m for help): t #键入t然后回车
Partition number (1-4): 1            #键入1然后回车，即指定第一个分区
Hex code (type L to list codes): c #键入c然后回车，即指定第一个分区为vfat
Changed system type of partition 1 to c (W95 FAT32 (LBA))

Command (m for help): w #键入w然后回车,保存分区表
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

格式化分区：

```
mkfs.vfat ${card}1
mkfs.ext4 ${card}2                                          #需要稍等片刻
```

然后写入bootloader：

```
cd $WORK_DIR/u-boot-sunxi
dd if=u-boot-sunxi-with-spl.bin of=$card bs=1024 seek=8
```

最后安装内核 uImage，设置启动参数：

```
mount ${card}1 /mnt
mkdir /mnt/boot
cp $WORK_DIR/linux-sunxi/arch/arm/boot/uImage /mnt/boot
cp $WORK_DIR/sunxi-boards/sys_config/a20/script.bin /mnt/boot
cp $WORK_DIR/boot.scr /mnt/
sync && umount /mnt
```

至此，启动到linux内核的工作已经完成，接下来我们就可以观看linux内核启动过程、进行内核调试了。

打印串口调试信息

通过查看串口输出信息，可以方便我们判断问题所在，进而找到解决问题的方法。

在linux和windows下有多种串口调试工具可供使用，这里仅给出两篇参考文章：

- http://linux-sunxi.org/Cubieboard/TTL
- http://cn.cubiebook.org/index.php?title=Cubieboard/串口调试

这里附上笔者的串口输出信息：http://mer.jolladev.net/data/media/cutecom.txt

信息结尾由于无法挂载根文件系统、启动init而等待10秒后重启。

总结

本文参考了众多网络内容，在后面列出了主要文章地址，在此一并感谢。

之所以写这篇内容，是因为从本人入手cubieboard2以来，通过苦寻资料，在各种痛苦与错误的尝试中不断走上了"正道"，深知对于一个没有嵌入式开发基础或开发经验的普通玩家而言，想把各种环节搞清楚是一件很难的事情，故通过大量引用、参考网络内容加上自己的摸索，总结出该文，以期望对新的玩家能够有所帮助；就嵌入式linux而言，整个 加电——启动bootloader——启动内核——加载rootfs流程对于新手会感到非常的模糊，而不知如何下手。本篇内容尽可能详细的描述了利用cubieboard从加电到启动linux内核的整个操作过程，为进一步学习如何构建一个可运行的linux系统打下了基础。后面，将会在此基础上继续介绍如何进一步挂载跟文件系统，启动到shell甚至GUI图形界面，从而构建一个完整、可用的linux系统。

参考

- http://linux-sunxi.org/Bootable_SD_card
- http://linux-sunxi.org/Boot
- https://github.com/linux-sunxi/u-boot-sunxi/wiki
- http://cn.cubieboard.org/forum.php?mod=viewthread&tid=1108&extra=page%3D1
- http://blog.csdn.net/yichunjie/article/details/8661176
- http://blog.csdn.net/abc47bca/article/details/6306005