# 绪论

本文属于作者原创，未经允许，不得传播。

联系方式：jeff-lv-1009080308@outlook.com

开始时间：2018-3-27

更新时间：2018-5-13

复习时间：

参考：《Tiny4412用户手册》

《Exynos 4412 SCP\_Users Manual\_Ver.0.10.00\_Preliminary0》

## 概述

本篇文章主要记录对友善之臂tiny4412开发板的使用心得。

目前本人手中的tiny4412底板是标准版，编号是1506，简略资源如下：

1. 1+4G

补充：git clone <https://github.com/jeff-lv-cn/tiny4412.git>

### dnw

dnw使用：

1. 进入dnw-linux，执行make编译，得到：src/driver/secbulk.o，src/dnw
2. 安装驱动文件src/driver/secbulk.o
3. uboot：dnw addr， PC：./dnw filename

# 刷机

## 概述

本章介绍tiny4412的刷机，目的有两个：1) 熟悉tiny4412的环境，为以后开发做准备；2) 验证官方提供源码以及image的正确性。故我的主要任务是烧写自己编译的image到开发板。需要准备linux操作系统，本人使用的是“ubuntu-14.04.5-server”。

说明：建立/home/jf/tiny4412目录用于存放tiny4412相关东西。该目录下创建了tools/，用于存放一些工具。

## 源码编译

将友善之臂光盘的如下内容拷贝到ubuntu下：

1. Disk-A\Linux\：

arm-linux-gcc-4.5.1-v6-vfp-20120301.tgz

linux-3.5-20170221.tgz

busybox-1.17.2-20101120.tgz

1. Disk-A\uboot：uboot\_tiny4412\_0929.tar.gz

注意：tiny4412(1G+4G)标准版在2015.7.26和2015.9.29生产的两批板子，其EMMC芯片不同，故uboot也有了两份。但光盘中仅提供uboot\_tiny4412-20130729.tgz，在启动开发版是会卡死。故使用uboot\_tiny4412\_0929.tar.gz版本。

### toolchains

解压arm-linux-gcc-4.5.1-v6-vfp-20120301.tgz，并导出toolchains到环境变量。使用的命令如下：

tar xvzf arm-linux-gcc-4.5.1-v6-vfp-20120301.tgz

cd opt/FriendlyARM/toolschain/;

chmod +w 4.5.1 –R; cp 4.5.1 /home/jf/tiny4412/tools/

export PATH=/home/jf/tiny4412/tools/4.5.1/bin:$PATH

### uboot

解压uboot\_tiny4412\_0929.tar.gz，修改Makefile后进行编译。使用的命令如下：

//vim Makefile ：CROSS\_COMPILE := arm-linux-

make tiny4412\_config O=./obj; make O=./obj // O=./obj是可选项，指定输出目录

make -C sd\_fuse //制作SD卡启动的工具

说明：make distclean O=./obj

### linux

解压linux-3.5-20170221.tgz，修改Makefile及.config后进行编译，使用命令如下：

//vim Makefile : ARCH := arm CROSS\_COMPILE := arm-linux-

cp tiny4412\_linux\_defconfig .config

mkdir obj; make zImage O=./obj

注意：若编译的内核由superboot启动，上面的步骤就可以完成。但本人需要由自己编译得而uboot来启动，故需要关闭trustzone选项，故在编译之前需要修改tiny4412\_linux\_defconfig：CONFIG\_ARM\_TRUSTZONE=n，并在编译过程中对该项选择为N。

### rootfs

## 烧写SD

本节介绍如何将系统烧写到SD卡，先准备一张SD卡，对其进行格式化。

### uboot

先将bootload烧写到SD卡，使用的命令如下：

cd /home/jf/tiny4412/uboot\_tiny4412\_0929/

make -C sd\_fuse

cd sd\_fuse/tiny4412

sudo ./sd\_fusing.sh /dev/sdb //可能需要cp obj/u-boot.bin ./

说明1：/dev/sdb是SD卡在ubuntu的设备节点，不同操作该节点可能不一样

说明2：如果uboot在编译时指定输出目录，需要复制u-boot.bin到uboot根目录

### linux

使用SD启动开发板，进入uboot命令行模式，对SD卡进行分区后使用使用fastboot烧写kernel、rootfs。烧写完上述image需要重启uboot设置启动参数，之后就可以启动linux。

uboot操作命令：

fdisk -c 0

fastboot //等待PC fastboot烧写

PC操作命令：

fastboot devices //查看设备是否连接上

fastboot flash bootloader tiny4412/linux/u-boot.bin

fastboot flash kernel tiny4412/linux/zImage

fastboot flash system tiny4412/linux/rootfs\_qtopia\_qt4.img

重启uboot，设置启动参数：

setenv bootargs root=/dev/mmcblk0p2 rootfstype=ext4 console=ttySAC0,115200 init=/linuxrc

说明：在uboot输入fastboot之后，此时PC才会识别到tiny4412。若是第一次使用fastboot，需要在此时安装驱动。成功可以在PC使用fastboot devices查看是否识别开发板。

### 补充

可以使用如下命令在fastboot下烧写其他内容：

fastboot flash fwbl1 tiny4412/linux/E4412\_N.bl1.bin

fastboot flash bl2 tiny4412/linux/bl2.bin

fastboot flash tzsw tiny4412/linux/E4412\_tzsw.bin

生成bl2方法：

cd /home/jf/tiny4412/uboot\_tiny4412\_0929/ sd\_fuse/

./mkbl2 ../u-boot.bin bl2.bin 14336

## x

mmcinfo 0 : 查看mmc卡信息。0 表示SD卡; 1表示emmc卡

fdisk -p 0 //查看SD卡分区信息

fatformat/ ext3format mmc 0:1 //格式化SD卡1号分区为fat/ext3

若内核启动卡死“Uncompressing Linux..... done, booting the kernel”

可以开启进行调试：

Kernel hacking -> Kernel low-level debugging functions

Kernel hacking -> Kernel low-level debugging messages via S3C UART

# 启动过程

参考: 《Android\_Exynos4412\_iROM\_Secure\_Booting\_Guide\_Ver.1.00.00》

<https://wenku.baidu.com/view/864da861cc17552706220813.html>

<http://www.techbulo.com/1335.html>

## 概述

本章从Exynos-4412内部iROM入手，分析整个系统的启动过程。

参考图3-1-1，Exynos-4412的启动过程分为如下几个个阶段：

1. iROM：系统复位或上电，CPU从iROM位置取出指令开始执行。iROM里的程序三星固化（无源码），主要任务为接下来程序执行提供最基本的环境初始化，然后加载外部存储设备中的BL1到iRAM执行。
2. BL1 & BL2：BL1和BL2在图3-1-1中可以看成是一个整体BL1，其作用是为了加载OS到外部DRAM，然后启动OS。但实际上BL1与BL2是分离的，对于开发人员来说也仅需要修改BL2

图3-1-2列举了Exynos-4412启动配置。

说明：为了描述方便，将Internal SRAM成为iRAM。

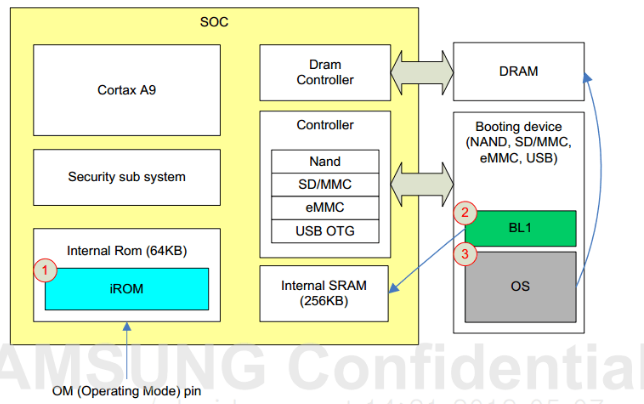


图3-1-1

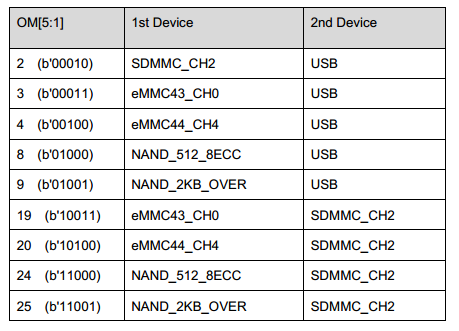


图3-1-2

## iROM

iROM内部固化程序的所做的工作参考图3-2-1:

1. 初始化内部环境，主要为BL1程序执行做准备：关闭看门狗、IRQ、MMU、D-cache、I-cache，设置clock divider、PLL….
2. 选择启动设备若“First Booting”没有成功则选择“Second Booting”，然后从启动设备中加载BL1到iRAM
3. 校核BL1的完整性，确定无误后解码BL1
4. 跳转到BL1进行执行。

说明：图3-2-1的“Direct-Go”用于从深度睡眠唤醒的动作，也就是我们常说的热启动。

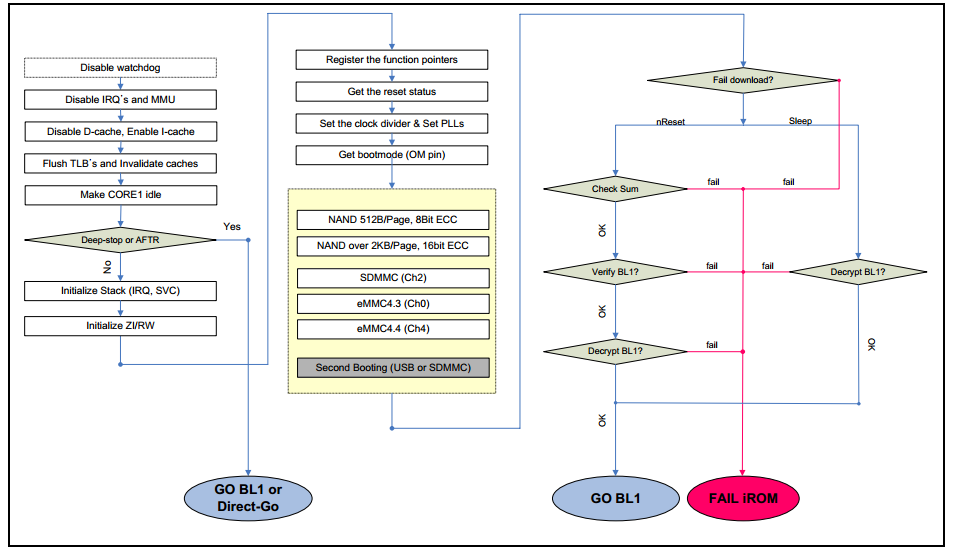


图3-2-1

## BL1 & BL2

BL1和BL2其实可以看作是一个东西，其目的是为了启动外部的OS。从友善之臂提供的资源来看，BL1仅提供image而不开放源码，BL2由uboot制作而来（仅仅拷贝了uboot前14K-4的数据，然后加上相应的校验和），基本也就可以看作uboot。故我们在本节仅仅介绍BL1的工作，BL2在uboot环节再来介绍。

先来看看为什么要分离BL1和BL2，三星公司的解释是为了实现“chip-dependent”和“plat-dependent”的分离。就我理解而言，这里的chip和plat可以看成是tiny4412的核心板和底板。核心板上主要是Exynos4412 SOC和DRAM，一般做产品开发是不用动的，但底板的资源就需要根据具体产品需求来设计了。故简单的理解BL1就是用来启动核心板，而BL2用来启动底板。所以使用tiny4412核心板的任何产品，无论底板具体硬件如何，都不需要仅需要修改BL2的代码就可以完成开发，这样也就实现了所谓“chip-dependent”和“plat-dependent”的分离思想。

猜测：关BL1和BL2分离的另一种猜测是BL1是为了保证软件的产权，即仅仅是三星公司认证的程序才能启动Exynos4412平台。而BL2则是为了突破iRAM仅有256K大小的限制，因为uboot一般大于256K。

## iRAM映射

iRAM一共有256K，参考图3-4-1，我们可以知道iRAM位于SoC寻址空间的0x0202\_0000~0x0206\_0000。iRAM的分配策略是是固定的，具体如下（从下到上）

1. 256B：产品ID，iROM版本号，内部函数指针…
2. 1.75KB（1792B）：iROM栈
3. 3KB（3072B）：iROM的ZI/RW（不太清楚）
4. 8KB（8192B）：BL1的执行区
5. 16K（16384）：BL2的执行区。合法的的BL2大小应该小于14KB-4B（14332B）。若BL2小于14332B，剩余部分应该填充为0.
6. 剩余：保留

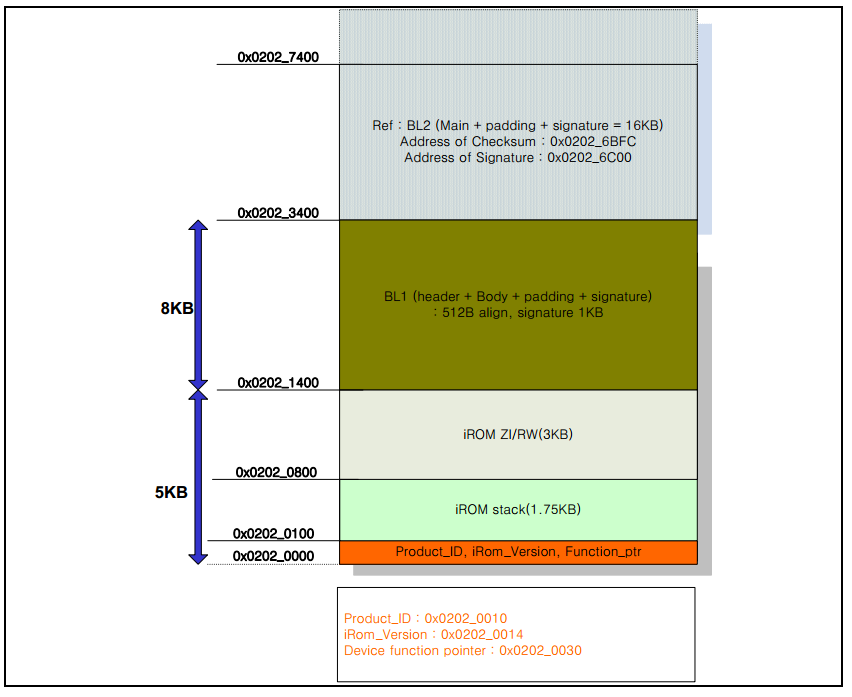


图3-4-1

### BL1

BL1所做的工作可以参考图3-3-1：

1. 初始化IRQ和SVC-stack，这也是内部环境的初始化，为BL2的运行做准备。
2. 选择启动设备。
3. 校核BL2的完整性，成功后解码。
4. 跳转到BL2处执行。

说明：BL1的工作和iROM有很大的相似性，内部环境准备、选择启动设备…同时，BL1这里也支持热启动。

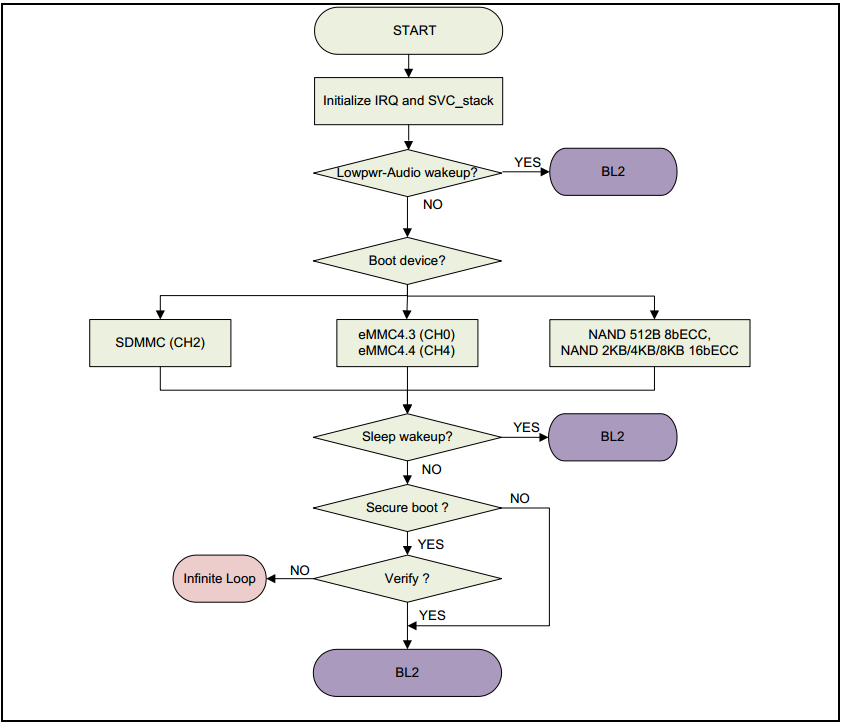


图3-3-1

## 启动设备分配

iROM会从启动设备拷贝8KB大小的数据到iRAM，而且对于不同的启动设备，其对于block的分配是不一样的。

### SD/MMC/MOVINAND

参考图3-5-1，SD卡等设备的block规划如下：

1. block1（512B）：保留
2. blcok1~16（8KB）：BL1存放区域。
3. blcok17~48（8KB）：BL2存放区域。

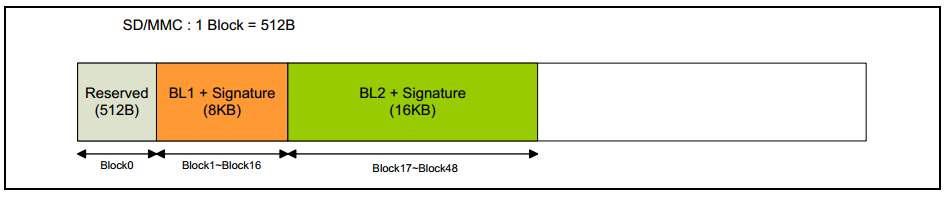


图3-5-1

### EMMC4.3/4.4

eMMC4.3/4.4的block分配是可以在启动时配置的，由CSSD寄存器配置。但三星提供的sample规划如下，参考图3-5-2：

1. 8KB：BL1的存放区域
2. 16KB：BL2的存放区域

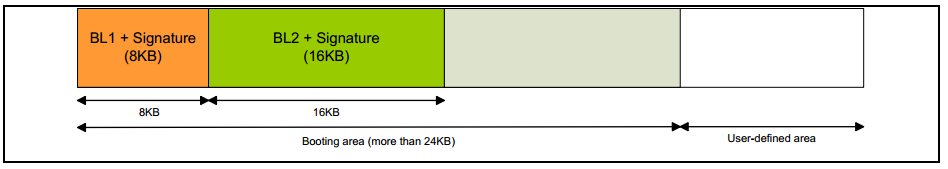


图3-5-2

# 资源

参考：《Exynos 4412 SCP\_Users Manual\_Ver.0.10.00\_Preliminary0》

## 概述

本章的主要目的，是结合官方的数据手册，对Exynos4412，做一个初步的认识。