

CSDN新首页上线啦，邀请你来立即体验！(http://blog.csdn.net/)

立即体

验

CSDN

博客 (http://blog.csdn.net/?ref=toolbar) 学院 (http://edu.csdn.net/?ref=toolbar)

下载 (http://download.csdn.net/?ref=toolbar) 更多 ▾

[搜索](#) [登录](#) [注册](#)

登录 (https://passport.csdn.net/account/login?ref=toolbar) 注册 (https://passport.csdn.net/account/mobile/register?ref=toolbar&action=mobileRegister)

4412 UBOOT移植

原创

2014年09月19日 14:23:23

932

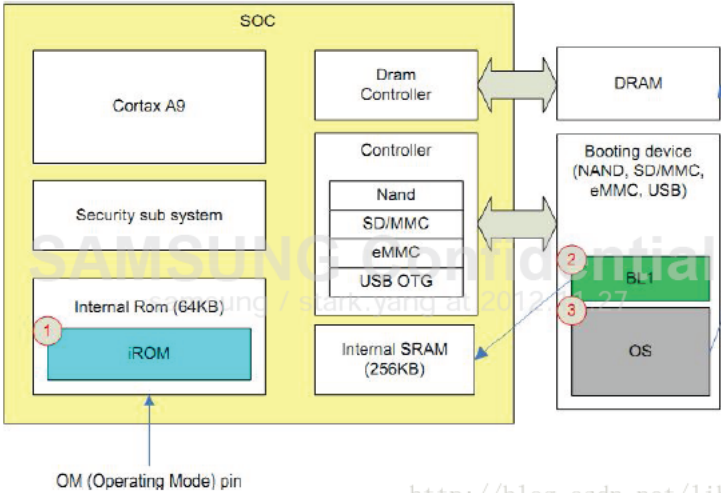
Exynos 4412 android 4.0版本系统使用的uboot为uboot2010.12, uboot的作用在这里简单来描述一下, uboot在整个系统来说就是一个启动引导代码, 就象我们PC中的BIOS,把系统复制到内存中运行, 然后跳转到内存中运行, 它的使命就到此结束, 以后就没它什么事了.

在早期ARM启动方式都和单片机一样, 都是从x00000000开始, 一般 0 地址接的是norflash,但后来随着技术的发展norflash已经不能满足用户的要求, 主要因为nor的容量, 价格, 速度等多方面的因素, 后面启动的方式增加了多种, 如:nandflash,usb,emmc等等. 但是这些的存储设备都不是接到x00000000地址, 所以就要在ARM芯片内部固化一个小程序, 来对启动方式 (OM0~OM5) 选来做一个判断,从而确定从哪来读取启动代码.

下面我们还是重点来介绍一下EXYNOS4412 bootloader吧, 首先我们从我们要烧写的文件来说起, 在烧写镜像中我们会看到下面一些文件:

E4412_bl1.bin
E4412_bl2.bin
E4412_tzsw.bin
u-boot.bin

要了解这些文件是什么作用, 我们就要从4412启动流程说起. 我们先看一下三星4412芯片启动框图:



我们可以从图中可以看到4412内部有64K的ROM和254K SRAM,在ROM中已经固化好了一段代码, 当硬件上电后首先运行的就是这部分代码. 这段代码三星其名为BL0(IROM BOOT 代码), 从图中我们很清楚看到一个运行过程.

1. 在芯片的IROM中已经固化一个代码,当硬件上电后就读取OM电平从而确定硬件设置的启动模式,
2. 把已经设置启动存储单元代码复制到内部RAM中并跳转到RAM运行.
3. 第三运行OS

在图中大概是这样一个描述启运过种, 这好象不太好理解, 我们接合u-boot启动方式来给大家再解释一下.

1. 这一部和上面介绍的一样, 运行IROM已经固化的代码, 读取硬件启动模式 (如: NOR,NAND.EMMC等启动模式), 并把UBOOT第一阶段代码复制到内部RAM中运行.

libh_2012 (http://blog.c...)

+ 关注

(http://blog.csdn.net/libh_2012)

码云

未开通 (https://gi

utm_sourc

他的最新文章
更多文章 (http://blog.csdn.net/libh_2012)

4412 UBOOT移植 (http://blog.csdn.net/libh_2012/article/details/39397605)

去查看 我是皇1115

去查看 我是皇1115

去查看 我是皇1115

去查看 我是皇1115

在线课程

数据科学家 从入门到精通

数据科学家 从入门到精通

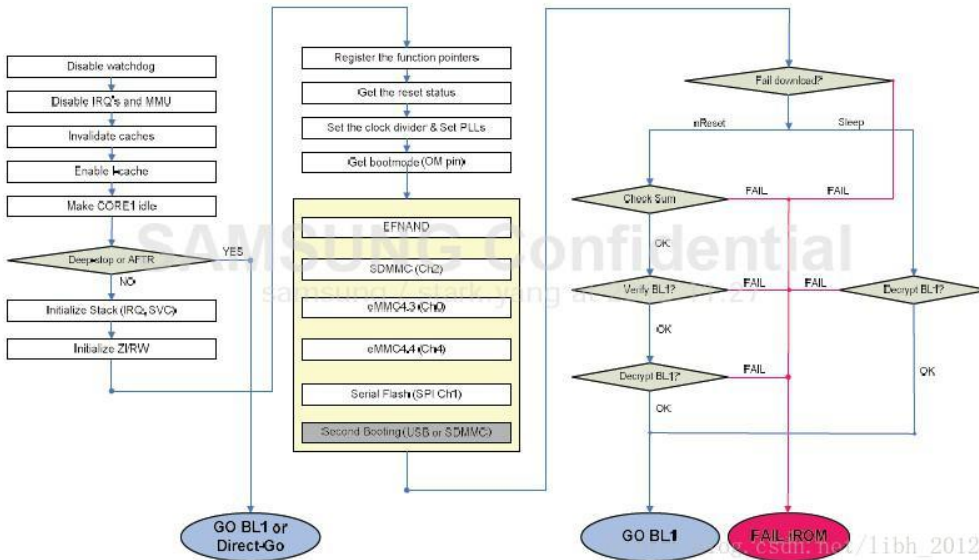
SDCC 2017 前端技术实战线上峰会

SDCC 2017 前端技术实战线上峰会

热门文章

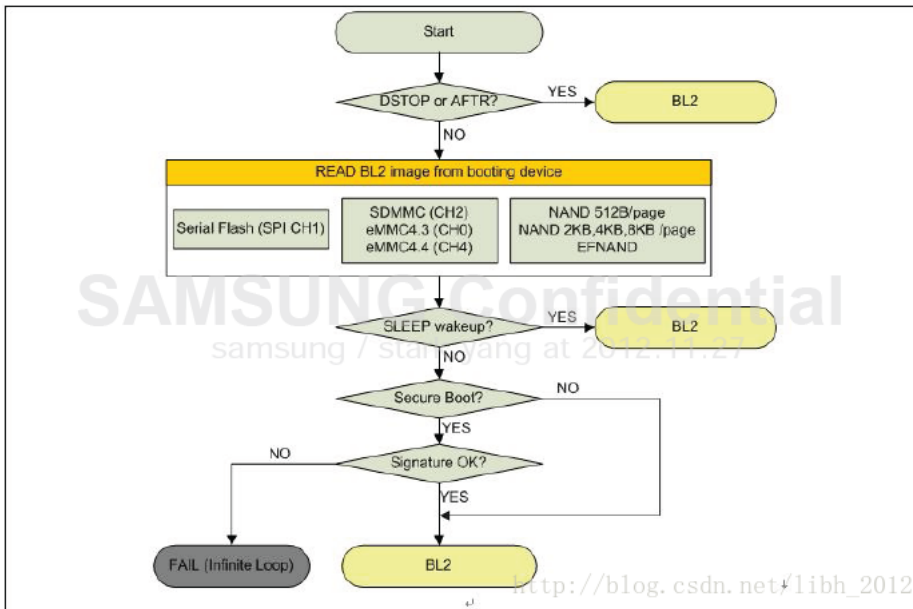
4412 UBOOT移植 (http://blog.csdn.net/libh_2012/article/details/39397605)
930

2. 运行uboot第一部分代码，复制uboot第二部代码到外接内存中（DDR3中）并跳转到DDR3运行，uboot/
 3. 运行uboot.加载kernel和android.
- 下面我们来分别介绍一下irom代码是怎么一个运行过程。



IROM流程图

在上图中可以看到，IROM中代码做了一些关watch dog \ MMU,中断,配制时钟，在写启动配制。然后去运行BL1.



BL1流程图

BL1也只做了一些简单的工作，把BL2从启动设备中放到内部IRAM中并运行。

好了，上面已经简单介绍一下启动代码运行的过程。现在我们介绍一下刚才四个文件是什么。

- E4412_bl1.bin：这是上面BL1镜像，主要作用加载BL2。
 E4412_bl2.bin: BL2代码镜像，也就是u-boot第一部分代码。
 E4412_tzsw.bin：trustzone镜像
 u-boot.bin：uboot第二部分代码镜像。

下面我们来讲一下uboot怎么移植到自己硬件上。

在网上有很多文章都有介绍uboot启动代码分析，我们在这里就不多讲这个，我们重点要讲的是怎么来移植一个uboot到自己的硬件板上，移植一个uboot到硬件板上大概有两种可能。

1. 到官网下载一下原始的uboot源代码，自己来添加平台代码。
2. 使用平台商提供的bsp，在这个基础上做uboot移植。

关于这两种方法，我们通常情况是选用第二种方式，因为第一种方式是芯片厂商做的事情，我们没有必要花那么多的时间来做平台移植。当然如果你只是想来学习学习有足够的时间的话可以尝试去移植。

为什么我们会选用第二种方式呢？

- 第一．芯片厂商会把这个移植相对成熟，我们自己来移植花要时间不说可能一时还不会做的比芯片厂商更稳定，
- 第二．在BSP的基础上移植花的时间会更少。

我们直接来说怎么来移植哪些文件。我们现在要讲的是基于三星4412芯片，X4412开发板。

下面我们来看一下X4412开发板uboot源代码目录。

在这些目录和我们关系最密切的就是arch和board两个目录。我们所有重点工作就是在这两个文件中。Arch是和cpu相关的代码目录。在X4412的开发板上这个目录中的代码基础本没有变更。三星已经根据4412芯片移植好，我们要做的就是对自己板端代码做修改，主要是board目录中主要有下面的一些文件。

要移植好一个板端的代码，就事先要认真对比一下自己的硬件板和三星smdk板有什么区别，这才会有针对情做一些代码修改。下面我们对比一个smdk4412和X4412有什么区别：

- 1．三星SMDK4412内在接了四片DDR3,分别接在两个通道中，这和X4412是一样所以内部分不要做修改，
- 2．三星使用的是PMIC 8767,而X4412使用的是DC-DC来直接搭建的电源系统。

对比了两个硬件最小系统基本也就是这两个地方不太一样。

知道了现个硬件板不区别修改起代码就方便多了。下面我们来整体地来看一下代码。

在看代码之前我们要知道代码是从哪里开始运行的，从哪个文件来始的，我们先来看一下

```
uboot/board/Samsung/x4412/u-boot.lds
OUTPUT_FORMAT("elf32-littlearm", "elf32-littlearm", "elf32-littlearm")
OUTPUT_ARCH(arm)
ENTRY(_start)
SECTIONS
{
    . = 0x00000000;

    . = ALIGN(4);
    .text:
    {
        arch/arm/cpu/armv7/start.o(.text)
        board/samsung/x4212/libx4212.o(.text)
        arch/arm/cpu/armv7/exynos/libexynos.o(.text)
        *(.text)
    }
}
```

从这里可以清楚看到平台为ARM平台也就是4412平台，入口地址为0x00000000。

从arch/arm/cpu/armv7/start.s开始运行。

```
.globl _start
_start: b reset
可见程序的开始就是跳到标号为reset的地方执行
reset:
    /*
    * set the cpu to SVC32 mode
    */
    mrs r0, cpsr
    bic r0, r0, #0x1f
    orr r0, r0, #0xd3
    msr cpsr,r0
    .....
    bl cpu_init_crit
从程序的注释就可以看出以上语句是让cpu进入SVC模式,然后跳转到
cpu_init_crit:
bl cache_init
```

```

/*
 * Invalidate L1 I/D
 */
movr0, #0@ set up for MCR
mcrp15, 0, r0, c8, c7, 0@ invalidate TLBs
mcrp15, 0, r0, c7, c5, 0@ invalidate icache

/*
 * disable MMU stuff and caches
 */
mrcp15, 0, r0, c1, c0, 0
bicr0, r0, #0x00002000@ clear bits 13 (--V-)
bicr0, r0, #0x00000007@ clear bits 2:0 (-CAM)
orr0, r0, #0x00001000@ set bit 12 (---I) Icache
orr0, r0, #0x00000002@ set bit 1 (--A-) Align
orr0, r0, #0x00000800@ set bit 11 (Z---) BTB
mcrp15, 0, r0, c1, c0, 0

/*
 * Jump to board specific initialization...
 * The Mask ROM will have already initialized
 * basic memory. Go here to bump up clock rate and handle
 * wake up conditions.
 */
movip, lr@ persevere link reg across call
bllowlevel_init@ go setup pll,mux,memory
movlr, ip@ restore link
movpc, lr@ back to my caller
cpu_init_cirt主要是初始化cache，关闭MMU和中断，然后进入到lowlevel_init，lowlevel_init位于
board/samsung/x4212/lowlevel_init.s文件中
.globllowlevel_init
lowlevel_init:
    ldr    r0,=(INF_REG_BASE+INF_REG1_OFFSET)
    ldr    r1,[r0]
    /*Sleepwakeup reset*/
    ldr    r2,=S5P_CHECK_SLEEP
    cmp    r1,r2
    beq    wakeup_reset

判断CPU是怎样进入复位的，如果是从睡眠唤醒进入复位的，就跳转到wakeup_reset，否则继续往下执行
/* BL pmic_init */ X4412没有PMIC所以这里注释掉
    bl    uart_asm_init
初始化电源管理芯片和串口，Uboot运行时打印出来的信息都是通过这个初始化的串口打印出来的
    bl    read_om
    .....
    read_om:
        /*Readbooting information*/
        ldr    r0,=S5PV310_POWER_BASE
        ldr    r1,[r0,#OMR_OFFSET]
        bic    r2,r1,#0cffffffc1
S5PV310_POWER_BASE是OM_STAT寄存器，存储OM管脚的信息，整个语句下来，就把OM管脚信息
（启动信息）存储在寄存器r2中，接下来的程序就是判断从何处启动了
    .....
/* eMMC 4.4 BOOT*/
    cmp    r2, #0x8
    moeq    r2,#BOOT_EMMC_4_4
    cmp    r2,#0x28
    moveq    r3,#BOOT_EMMC_4_4
    ldr    r0,=INF_RGE_BASE
    str    r3,[r0,#INF_REG3_OFFSET]

```

```
ldr pc,lr
```

我这个开发板是从emmc4.4启动所以满足上述语句，同时将启动信息保存在INFORM3寄存器中，然后回到程序跳转的地方，接着bl read_om往下执行

```
ldr r0, =0xff000fff
bic r1,pc,r0 /*r0<-current base addr of code*/
ldr r2,_TEXT_BASE /*r1<-original base addr in ram*/
bic r2,r2,r0 /*r0<-current base addr of code*/
cmp r1,r2 /*compare r0,r1*/
beq after_copy
```

判断当前程序指针是否在SDRAM中，如果是，则跳过SDRAM的初始化，也不需要装载u-boot.bin了，直接执行after_copy，我这个开发板是从emmc启动的，所以程序此时还不在于SDRAM中，还要接着往下执行

```
/*Memory initialize */
bl mem_ctrl_asm_init
/*initsystem clock*/
bl system_clock_init
b 1f
```

内存初始化以及系统时钟初始化，然后跳转到前面的标号为1处

```
1:
bl tzpc_init
b load_uboot
```

初始化trustzoneprotection controller，并跳转到load_uboot

```
load_uboot:
ldr r0,=INF_REG_BASE
ldr r1,[r0,#INF_REG3_OFFSET]
.....
cmp r1,#BOOT_NAND
.....

cmp r1,#BOOT_EMMC_4_4
beq emmc_boot_4_4
.....
```

通过判断从何处启动，跳转到相应的标号处执行，我这块开发板将跳转到emmc_boot_4_4,该标号处的语句作用就是从emmc处拷贝uboot到SDRAM中，然后跳转到after_copy处

```
after_copy:
#ifdef CONFIG_SMDKC220
/*set up C2C*/
ldr r0,=S5PV310_SYSREG_BASE
ldr r2,=GENERAL_CTRL_C2C_OFFSET
ldr r1,[r0,r2]
ldr r3,=0x4000
orr r1,e1,r3
str r1,[r0,f2]
#endif
#ifdef CONFIG_ENABLE_MMU
bl enable_mmu
#endif
/*store second boot information in u-boot C lerve variable*/
ldr r0,=CONFIG_PHY_UBOOT_BASE
sub r0,r0,#8
ldr r1,[r0]
ldr r0,_second_boot_info
str r1,[r0]
.....
ldr r0,_board_init_f
mov pc,r0
```

after_copy所做的工作主要就是初始化设置C2C，打开MMU，并跳到board_init_f执行板级初始化。Board_init_f函数位于arch/arm/lib/board.c中

```

void board_init_f (ulong bootflag)
{
    bd_t *bd;
    init_fnc_t **init_fnc_ptr;
    gd_t *gd;
    ulong addr, addr_sp;

    /* Pointer is writable since we allocated a register for it */
    gd = (gd_t *) ((CONFIG_SYS_INIT_SP_ADDR) & ~0x07);
    /* compiler optimization barrier needed for GCC >= 3.4 */
    __asm__ __volatile__ ("": : : "memory");

    memset ((void*)gd, 0, sizeof (gd_t));

    gd->mon_len = _bss_end_ofs;
    /* 执行init_sequence函数 */
    for (init_fnc_ptr = init_sequence; *init_fnc_ptr; ++init_fnc_ptr) {
        if ((*init_fnc_ptr)() != 0) {
            hang ();
        }
    }

    init_fnc_t *init_sequence[] = {
        #if defined(CONFIG_ARCH_CPU_INIT)
        arch_cpu_init, /* CPU相关的设置 */
        #endif
        #if defined(CONFIG_BOARD_EARLY_INIT_F)
        board_early_init_f,
        #endif
        timer_init, /* 初始化时钟 */
        #ifdef CONFIG_FSL_ESDHC
        get_clocks,
        #endif
        env_init, /* initialize environment */
        #if defined(CONFIG_S5P6450) && !defined(CONFIG_S5P6460_IP_TEST)
        init_baudrate, /* initialize baudrate settings */
        serial_init, /* 串口初始化 */
        #endif
        console_init_f, /* stage 1 init of console */
        display_banner, /* say that we are here */
        #if defined(CONFIG_DISPLAY_CPUINFO)
        print_cpuinfo, /* 打印CPU相关信息 */
        #endif
        #if defined(CONFIG_DISPLAY_BOARDINFO)
        checkboard, /* 显示硬件板信息 */
        #endif
        #if defined(CONFIG_HARD_I2C) || defined(CONFIG_SOFT_I2C)
        init_func_i2c,
        #endif
        dram_init, /* 配制SDRAM banks个数 */
        #if defined(CONFIG_CMD_PCI) || defined (CONFIG_PCI)
        arm_pci_init,
        #endif
        NULL,
    };

```

上面就是uboot第一部分代码（BL2）的执行过程。这一部分代码基本都不用做太多的修改。在X4412要做修改的基本集中在四个文件上。lowlevel_init.S，mem_init_x4212.S，x4212.c，X4412.h。

我们先来看一下X4412.h文件要改的地方。

```

#define CONFIG_SERIAL31    / / 定义打印输出串口
#define CONFIG_SERIAL_MULTI1

```

```

#define CONFIG_USB_OHCI
#undef CONFIG_USB_STORAGE
#define CONFIG_S3C_USBD
#undef CONFIG_USB_CPUMODE
. . . . .

#ifdef CONFIG_EVT0_STABLE
#define CONFIG_NR_DRAM_BANKS2
#else
#ifdef USE_2G_DRAM
#define CONFIG_NR_DRAM_BANKS8
#else
#define CONFIG_NR_DRAM_BANKS4
#endif
#endif
#define SDRAM_BANK_SIZE      0x10000000 /* 256 MB */
#define PHYS_SDRAM_1         CONFIG_SYS_SDRAM_BASE /* SDRAM Bank #1 */
#define PHYS_SDRAM_1_SIZE    SDRAM_BANK_SIZE
#define PHYS_SDRAM_2         (CONFIG_SYS_SDRAM_BASE + SDRAM_BANK_SIZE) /* SDRAM Bank #2 */
#define PHYS_SDRAM_2_SIZE    SDRAM_BANK_SIZE
#define PHYS_SDRAM_3         (CONFIG_SYS_SDRAM_BASE + 2 * SDRAM_BANK_SIZE) /* SDRAM Bank #3 */
#define PHYS_SDRAM_3_SIZE    SDRAM_BANK_SIZE
#define PHYS_SDRAM_4         (CONFIG_SYS_SDRAM_BASE + 3 * SDRAM_BANK_SIZE) /* SDRAM Bank #4 */
#define PHYS_SDRAM_4_SIZE    SDRAM_BANK_SIZE
#define PHYS_SDRAM_5         (CONFIG_SYS_SDRAM_BASE + 4 * SDRAM_BANK_SIZE) /* SDRAM Bank #5 */
#define PHYS_SDRAM_5_SIZE    SDRAM_BANK_SIZE
#define PHYS_SDRAM_6         (CONFIG_SYS_SDRAM_BASE + 5 * SDRAM_BANK_SIZE) /* SDRAM Bank #6 */
#define PHYS_SDRAM_6_SIZE    SDRAM_BANK_SIZE
#define PHYS_SDRAM_7         (CONFIG_SYS_SDRAM_BASE + 6 * SDRAM_BANK_SIZE) /* SDRAM Bank #7 */
#define PHYS_SDRAM_7_SIZE    SDRAM_BANK_SIZE
#define PHYS_SDRAM_8         (CONFIG_SYS_SDRAM_BASE + 7 * SDRAM_BANK_SIZE) /* SDRAM Bank #8 */
#define PHYS_SDRAM_8_SIZE    SDRAM_BANK_SIZE

```

上面的代码中CONFIG_NR_DRAM_BANKS在这里要解释一下，这里和以前s5pv210里面有点不一样，在210平台上这个宏是定义接了几个SDRAM通道。

而在4412里面不是这样，这里只是以256MB为bank一个单位来计算内存容量，所以我们接的是1GB的话这里就要设置成4，如果是2GB的话应该设置成8。

在这里定义好了。我们就要在X4412.C里做相应的修改。

```

dram_init_banks(void)
{
    nr_dram_banks = CONFIG_NR_DRAM_BANKS;
    gd->bd->bi_dram[0].start = PHYS_SDRAM_1;
    gd->bd->bi_dram[0].size = PHYS_SDRAM_1_SIZE;
    gd->bd->bi_dram[1].start = PHYS_SDRAM_2;
    gd->bd->bi_dram[1].size = PHYS_SDRAM_2_SIZE;
    gd->bd->bi_dram[2].start = PHYS_SDRAM_3;
    gd->bd->bi_dram[2].size = PHYS_SDRAM_3_SIZE;
    gd->bd->bi_dram[3].start = PHYS_SDRAM_4;
    gd->bd->bi_dram[3].size = PHYS_SDRAM_4_SIZE;
#ifdef USE_2G_DRAM
    gd->bd->bi_dram[4].start = PHYS_SDRAM_5;
    gd->bd->bi_dram[4].size = PHYS_SDRAM_5_SIZE;
    gd->bd->bi_dram[5].start = PHYS_SDRAM_6;

```

```
gd->bd->bi_dram[5].size = PHYS_SDRAM_6_SIZE;
gd->bd->bi_dram[6].start = PHYS_SDRAM_7;
gd->bd->bi_dram[6].size = PHYS_SDRAM_7_SIZE;
gd->bd->bi_dram[7].start = PHYS_SDRAM_8;
gd->bd->bi_dram[7].size = PHYS_SDRAM_8_SIZE;
#endif
```

还有一个文件mem_init_x4212.S，这里主要是对MEMORY寄存器做设置，大家可以参数4412datasheet第 1 8 章.对下面寄存器做相应的设置，我们在这里就不细说了。

代码简单介绍了一下，下面我们来介绍怎么来烧写到开发板中，在 4 4 1 2 上有OM几个引脚，这几个就是来设置启动模式，通常当一块新的板子是没有uboot的，所以无法正常启动，传统的方法是通过一个仿真器来烧写，但是仿真器对于个人或一些小的公司来说实在是太贵了，所以三星想到了通过SD卡的方式来完成烧写功能。

严格来说SD卡在这里只是作为一个桥梁作用，就象我们PC机的启动盘，它是把uboot.bin文件先烧写到sd卡中，通过启动的uboot把uboot.bin再写到emmc或别的存储单元。

通过SD卡来烧写uboot时要注意设置，启动方式有分第一启动和第二启动，何为第一第二呢，就是说当第一启动单元没有程序时会自动跳到第二启动单元去运行程序，而烧写的程序启动是烧写到第一启动单元。

这一点一定要注意，否则你烧写失败都不知道什么原因。

下面介绍一下在ubuntu下制作一个启动盘

首先接入一张SD卡在PC 上，
进入uboot/sdfuse目录输入命令
sudo bash ./sd_fusing.sh /dev/sdc
通过SD_fusing.sh脚本把 . BL1.BL2.TZSW.uboot.bin分别写到了SD卡中。地址如下：
fusing images

signed_bl1_position=1 / / BL1地址
bl2_position=31 / / BL2地址
uboot_position=63 / / uboot.bin地址
tzsw_position=719 / / tszw地址

这样就可以制作一张启动盘，接到板子上就可以正常动行，就可通过fastboot使用烧写所有的image到EMMC中。

更多的内容可下载 4 4 1 2 开发手册。

链接：<http://pan.baidu.com/s/1gdGKEEnL> 密码：xc3h

内容举报

返回顶部





相关文章推荐



Exynos4412 Uboot 移植（二）—— Uboot 启动流程分析 (<http://blog.csdn.net/zqixiao...>




U-Boot 属于两阶段的Bootloader，第一阶段的文件为arch/arm/cpu/armv7 /start.S 和 board//lowlevel_init.S，前者是平台相关的，后者是开发板...

 zqixiao_09 (http://blog.csdn.net/zqixiao_09) 2016年03月04日 22:29  4982



exynos4412_uboot移植与分析 (<http://download.csdn.net/detail/sike...>



2015年06月08日 20:22

737KB

下载

都是前端，月薪20K和40k的开发到底差距在哪？

大学毕业后我成为前端开发者，从一开始的小白到现在的“高手”，我把一些感想记录下来...

(http://www.baidu.com/cb.php?c=Igf_pyfqHmknj0dP1f0IZ0qnfK9ujYzP1nYPH0k0Aw-5Hc3rHnYnHb0TAq15HfLPWRznjb0T1dWmHKBryRzPvDvryuhn1fY0AwY5HDdnHDLPWczrj60Igf_5y9YIZ0IQzq-uZR8mLPbUB48ugfEIAqspynElvNBnHqdIAdxTvqdThP-5yF_UvTkn0KzujY1n0KBUHYs0ZKz5H00Iy-b5HDdP1f1PWD0Uv-b5HDZrH63nHf0mv-b5HTzPWb1n6KEIv3qn0KsXHYznjm0mLFW5HfzrH0s)



exynos4412-uboot移植笔记 (http://download.csdn.net/detail/sea11...


2016年08月05日 22:19

114KB

下载

Exynos4412 Uboot 移植（六）—— 相关知识补充 (http://blog.csdn.net/zqixiao_09/art...


一、gd结构体的定义与使用 gd_t和bd_t是u-boot中两个重要的数据结构，在初始化操作很多都要靠这两个数据结构来保存或传递。分别定义在./include/asm-armgd_t和bd_t是u-...




zqixiao_09 (http://blog.csdn.net/zqixiao_09) 2016年03月07日 15:46 1966

【Tiny4412--2】Uboot移植 (http://blog.csdn.net/flappy_boy/article/details/737290...

阅读说明该系列文章，基于的平台均为Tiny4412SDK 1312；4G EMMC; 1G DDR3带@lamar: 前缀的表示终端里面的命令前言此处略准备 tiny4412 开发板 ubuntu 1...



flappy_boy (http://blog.csdn.net/flappy_boy) 2017年06月25日 20:03 106




人人都能看懂的 AI 入门课

本课程将讲述人工智能的现状、应用场景和入门方法，并通过运用 TensorFlow，使得受众能清晰了解人工智能的运作方式。

(http://www.baidu.com/cb.php?c=Igf_pyfqHmknjfrjc0IZ0qnfK9ujYzP1f4Pjn10Aw-5Hc4nj6vPjm0TAq15Hf4rjn1n1b0T1YvPj01njbvmWnLPHTYrAFh0AwY5HDdnHDLPWczrj60Igf_5y9YIZ0IQzqMpgwBUvqoQhP8QvIGIAPCmgffmwa_Iyd:n16kPWKWrHnvnHRvvnNBuYD4PHqdIAdxTvqdThP-5HDknWFWmhkEusKzujY1n0KBUHYs0ZKz5H00Iy-b5HDdP1f1PWD0Uv-b5HD1nj0sPjn0mv-b5HTzPWb1n6KEIv3qn0KsXHYznjm0mLFW5HczPWfk)

关于 4412 uboot 移植与分析 (http://blog.csdn.net/livesflying/article/details/49074463)


Exynos 4412 android 4.0 版本系统使用的 uboot 为 uboot2010.12, uboot的作用在这里简单来描述一下，uboot 在整个系统来说就是一个启动引导代码，就象我...



livesflying (http://blog.csdn.net/livesflying) 2015年10月12日 15:33 121

【嵌入式开发学习笔记】Exynos4412 uboot移植笔记 (http://blog.csdn.net/zjq77700/ar...


嵌入式开发学习笔记-Exynos4412 uboot移植笔记



zjq77700 (http://blog.csdn.net/zjq77700) 2016年04月19日 16:31 2063

Exynos4412 Uboot 移植（六）—— 相关知识补充 (http://blog.csdn.net/u013491946/a...

Uboot版本：u-boot-2013.01 一、gd结构体的定义与使用 gd_t 和 bd_t 是u-boot中两个重要的数据结构，在初始化操作很多都要靠这两个数据结构来...



u013491946 (http://blog.csdn.net/u013491946) 2017年06月27日 20:46 122

FS4412系统移植实验手册-uboot移植 (http://download.csdn.net/detail/andylauren/95...




2016年05月21日 16:54 434KB [下载](#)



FS4412系统移植uboot移植实验代码 (<http://download.csdn.net/detail/...>)
2016年05月21日 17:03 34.62MB [下载](#)


Exynos4412 Uboot 移植（一）—— Uboot 编译流程分析 (<http://blog.csdn.net/zqixiao...>)

Uboot 所用版本 u-boot-2013.01 u-boot-2013.01 中有上千文件，要想了解对于某款开发板，使用哪些文件、哪些文件首先执行、可执行文件占用内存的情况，最好的方法...

 zqixiao_09 (http://blog.csdn.net/zqixiao_09) 2016年03月04日 21:44 [🔖4486](#)


Exynos4412 Uboot 移植（四）—— Uboot引导内核过程分析 (<http://blog.csdn.net/u01...>)

bootloader 要想启动内核，可以直接跳到内核的第一个指令处，即内核的起始地址，这样便可以完成内核的启动工作了。但是要想启动内核还需要满足一些条件，如下所示： 1、cpu 寄存器设置 ...

 u013491946 (<http://blog.csdn.net/u013491946>) 2017年06月27日 11:08 [🔖178](#)


Exynos4412 Uboot 移植（三）—— Uboot添加自定义命令 (<http://blog.csdn.net/sea11...>)

转载自http://blog.csdn.net/zqixiao_09/article/details/50805936 Uboot添加自定义命令：uboot中的命令使用U_BOOT_CMD这...

 sea1105 (<http://blog.csdn.net/sea1105>) 2016年08月07日 15:25 [🔖354](#)


Exynos4412 Uboot 移植（四）—— Uboot引导内核过程分析 (<http://blog.csdn.net/sea...>)

转载自http://blog.csdn.net/zqixiao_09/article/details/50817500 bootloader 要想启动内核，可以直接跳到内核的第一个指令处，即...

 sea1105 (<http://blog.csdn.net/sea1105>) 2016年08月07日 15:27 [🔖485](#)

Exynos4412 Uboot 移植（三）—— Uboot添加自定义命令 (<http://blog.csdn.net/zqixia...>)

Uboot添加自定义命令：uboot中的命令使用U_BOOT_CMD这个宏声明来注册进系统，链接脚本会把所有的cmd_tbl_t结构体放在相邻的地方。（占坑，后续添加。。。）...

 zqixiao_09 (http://blog.csdn.net/zqixiao_09) 2016年03月04日 22:36 [🔖2427](#)


Exynos4412 Uboot 移植（一）—— Uboot 编译流程分析 (<http://blog.csdn.net/huoho...>)

转载自http://blog.csdn.net/zqixiao_09/article/details/50805205 Uboot 所用版本 u-boot-2013.01 ...

 huohongpeng (<http://blog.csdn.net/huohongpeng>) 2017年05月12日 10:02 [🔖222](#)

Exynos4412 Uboot 移植（五）—— Uboot 移植过程 (http://blog.csdn.net/zqixiao_09/...)

Uboot 版本：u-boot-2013.01 开发板：FS_4412 平台（Exynos4412,可以根据自己的板子修改，只要是4412的过程都是一样的）一、建立自己的平台 1、下载源码 我们...

 zqixiao_09 (http://blog.csdn.net/zqixiao_09) 2016年03月07日 14:54 [🔖9233](#)

Exynos4412 Uboot 移植（五）—— Uboot 移植过程 (<http://blog.csdn.net/u01349194...>)

 内容举报

 返回顶部