

原

[uboot] （第三章）uboot流程——uboot-spl代码流程

2016年10月28日 16:24:14

阅

以下例子都以project X项目tiny210(s5pv210平台,armv7架构)为例。

- [uboot] uboot流程系列：
- [project X] tiny210(s5pv210)上电启动流程（BL0-BL2）
 - [uboot] （第一章）uboot流程——uboot-spl编译流程
 - [uboot] （第二章）uboot流程——uboot-spl编译流程

建议参考文章

- [kernel 启动流程] （第二章）第一阶段之——设置SVC、关闭中断
- [kernel 启动流程] （第六章）第一阶段之——打开MMU
- ARM的CP15协处理器的寄存器

建议先看《[project X] tiny210(s5pv210)上电启动流程（BL0-BL2）》，根据例子了解一下上电之后的BL0\BL1\BL2阶段，以及各个阶段的运行位置，

一、说明

1、uboot-spl入口说明

通过uboot-spl编译脚本project-X/u-boot/arch/arm/cpu/u-boot-spl.lds

```
1 ENTRY(_start)
```

所以uboot-spl的代码入口函数是_start
对应于路径project-X/u-boot/arch/arm/lib/vector.S的_start，后续就是从这个函数开始分析。

2、CONFIG_SPL_BUILD说明

前面说过，在编译SPL的时候，编译参数会有如下语句：
project-X/u-boot/scripts/Makefile.spl

```
1 KBUILD_CPPFLAGS += -DCONFIG_SPL_BUILD
```

所以说在编译SPL的代码的过程中，CONFIG_SPL_BUILD这个宏是打开的。
uboot-spl和uboot的代码是通用的，其区别就是通过CONFIG_SPL_BUILD宏来进行区分的。

二、uboot-spl需要做的事情

CPU初始刚上电的状态。需要小心的设置好很多状态，包括cpu状态、中断状态、MMU状态等等。
在armv7架构的uboot-spl，主要需要做如下事情

- 关闭中断，svc模式
- 禁用MMU、TLB
- 芯片级、板级的一些初始化操作
 - IO初始化
 - 时钟

- 内存
 - 选项，串口初始化
 - 选项，nand flash初始化
 - 其他额外的操作
- 加载BL2，跳转到BL2

上述工作，也就是uboot-spl代码流程核心。

三、代码流程

1、代码整体流程

代码整体流程如下，以下列出来的都是spl核心函数。



2、_start

上述已经说明了_start是整个spl的入口，其代码如下：

arch/arm/lib/vector.S

```
1 _start:
2 #ifdef CONFIG_SYS_DV_NOR_BOOT_CFG
3     .word    CONFIG_SYS_DV_NOR_BOOT_CFG
4 #endif
5     b       reset
```

会跳转到reset中。
注意，spl的流程在reset中就应该被结束，也就是说在reset中，就应该转到到BL2，也就是uboot中了。
后面看reset的实现。

3、reset

建议先参考[kernel 启动流程]（第二章）第一阶段之——设置SVC、关闭中断，了解一下为什么要设置SVC、关闭中断以及如何操作。

代码如下：

arch/arm/cpu/armv7/start.S

```
1     .globl  reset
2     .globl  save_boot_params_ret
3
4 reset:
5     /* Allow the board to save important registers */
6     b       save_boot_params
7 save_boot_params_ret:
8     /*
9      * disable interrupts (FIQ and IRQ), also set the cpu to SVC32 mode,
10     * except if in HYP mode already
11     */
12     mrs r0, cpsr
13     and r1, r0, #0x1f @ mask mode bits
14     teq r1, #0x1a @ test for HYP mode
15     bicne r0, r0, #0x1f @ clear all mode bits
16     orrne r0, r0, #0x13 @ set SVC mode
17     orr r0, r0, #0xc0 @ disable FIQ and IRQ
```

```

18     msr cpsr,r0
19 @@ 以上通过设置CPSR寄存器里设置CPU为SVC模式，禁止中断
20 @@ 具体操作可以参考《[kernel启动流程] (第二章) 第一阶段——设置SVC、关闭中断》的分析
21
22     /* the mask ROM code should have PLL and others stable */
23 #ifndef CONFIG_SKIP_LOWLEVEL_INIT
24     bl cpu_init_cp15    写评论
25 @@ 调用cpu_init_cp15，初始化协处理器CP15，从而禁用MMU和TLB。
26 @@ 后面会有一小节进行分析
27
28     bl cpu_init_crit
29 @@ 调用cpu_init_crit，进行关键的初始化动作，也就是平台级和板级的初始化
30 @@ 后面会有一小节进行分析
31 #endif
32
33     bl _main
34 @@ 跳转到主函数，也就是要执行的BL2以及跳转到BL2的主体部分

```

目录

收藏

微信

微博

QQ

4、cpu_init_cp15

建议先参考[kernel启动流程] (第六章) 第一阶段之——打开MMU两篇文章的分析。

cpu_init_cp15主要用于对cp15协处理器进行初始化，其主要目的就是关闭其MMU和TLB。

代码如下(去掉无关部分的代码)：

arch/arm/cpu/armv7/start.S

```

1 ENTRY(cpu_init_cp15)
2     /*
3      * Invalidate L1 I/D
4      */
5     mov r0, #0        @ set up for MCR
6     mcr p15, 0, r0, c8, c7, 0 @ invalidate TLBs
7     mcr p15, 0, r0, c7, c5, 0 @ invalidate icache
8     mcr p15, 0, r0, c7, c5, 6 @ invalidate BP array
9     mcr p15, 0, r0, c7, c10, 4 @ DSB
10    mcr p15, 0, r0, c7, c5, 4 @ ISB
11 @@ 这里只需要知道是对CP15处理器的部分寄存器清零即可。
12 @@ 将协处理器的c7\c8清零等等，各个寄存器的含义请参考《ARM的CP15协处理器的寄存器》
13
14     /*
15      * disable MMU stuff and caches
16      */
17     mrc p15, 0, r0, c1, c0, 0
18     bic r0, r0, #0x00002000 @ clear bits 13 (--V-)
19     bic r0, r0, #0x00000007 @ clear bits 2:0 (-CAM)
20     orr r0, r0, #0x00000002 @ set bit 1 (--A-) Align
21     orr r0, r0, #0x00000800 @ set bit 11 (Z---) BTB
22 #ifdef CONFIG_SYS_ICACHE_OFF
23     bic r0, r0, #0x0001000 @ clear bit 12 (I) I-cache
24 #else
25     orr r0, r0, #0x0001000 @ set bit 12 (I) I-cache
26 #endif
27     mcr p15, 0, r0, c1, c0, 0
28 @@ 通过上述的文章的介绍，我们可以知道cp15的c1寄存器就是MMU控制器
29 @@ 上述对MMU的一些位进行清零和置位，达到关闭MMU和cache的目的，具体的话去看一下上述文章吧。
30
31 ENDPROC(cpu_init_cp15)

```

5、cpu_init_crit

cpu_init_crit，进行一些关键的初始化动作，也就是平台级和板级的初始化。其代码核心就是lowlevel_init，如下

arch/arm/cpu/armv7/start.S

```

1 ENTRY(cpu_init_crit)
2     /*

```

```

3      * Jump to board specific initialization...
4      * The Mask ROM will have already initialized
5      * basic memory. Go to bump up clock rate and handle
6      * wake up condition
7      */
8      b lowlevel_init go setup pll,mux,memory
9  ENDPROC(cpu_init_crit) 写评论

```

所以说lowlevel_init就是这个函数。

lowlevel_init一般是由板级代码自己实现的。但是对于某些平台来说，也可以使用通用的lowlevel_init，其定义在arch/arm/cpu/lowlevel_init.S中以tiny210为例，在移植tiny210的时候，就需要在board/samsung/tiny210下，也就是板级目录下面创建lowlevel_init.S，在内部实现lowlevel_init。（其实lowlevel_init了就好，没必要说它是实现，但是通常规范都是创建了lowlevel_init.S来专门实现lowlevel_init函数）。

在lowlevel_init中，我们要实现如下：

- * 检查一些复位状态
- * 关闭看门狗
- * 系统时钟的初始化
- * 内存、DDR的初始化
- * 串口初始化（可选）
- * Nand flash的初始化

下面以tiny210的lowlevel_init为例（这里说明一下，当时移植tiny210的时候，是直接把kangear的这个lowlevel_init.S文件拿过来用的）

这部分代码和平台相关性很强，简单介绍一下即可

board/samsung/tiny210/lowlevel_init.S

```

1  lowlevel_init:
2      push    {lr}
3
4      /* check reset status */
5
6      ldr r0, =(ELFIN_CLOCK_POWER_BASE+RST_STAT_OFFSET)
7      ldr r1, [r0]
8      bic r1, r1, #0xffff6ffff
9      cmp r1, #0x10000
10     beq wakeup_reset_pre
11     cmp r1, #0x80000
12     beq wakeup_reset_from_didle
13     @@ 读取复位状态寄存器0xE010_a000的值，判断复位状态。
14
15     /* IO Retention release */
16     ldr r0, =(ELFIN_CLOCK_POWER_BASE + OTHERS_OFFSET)
17     ldr r1, [r0]
18     ldr r2, =IO_RET_REL
19     orr r1, r1, r2
20     str r1, [r0]
21     @@ 读取混合状态寄存器E010_e000的值，对其中的某些位进行置位，复位后需要对某些wakeup位置1，具体我也没搞懂。
22
23     /* Disable Watchdog */
24     ldr r0, =ELFIN_WATCHDOG_BASE /* 0xE2700000 */
25     mov r1, #0
26     str r1, [r0]
27     @@ 关闭看门狗
28
29     @@ 这里忽略掉一部分对外部SRAM操作的代码
30
31     /* when we already run in ram, we don't need to relocate U-Boot.
32      * and actually, memory controller must be configured before U-Boot
33      * is running in ram.
34      */
35     ldr r0, =0x00ffffff
36     bic r1, pc, r0 /* r0 <- current base addr of code */
37     ldr r2, _TEXT_BASE /* r1 <- original base addr in ram */
38     bic r2, r2, r0 /* r0 <- current base addr of code */
39     cmp r1, r2 /* compare r0, r1 */

```

```

40     beq     1f                /* r0 == r1 then skip sdram init */
41 @@ 判断是否已经在SDRAM上运行了，如果是的话，就跳过以下两个对ddr初始化的步骤
42 @@ 判断方法如下：
43 @@ 1、获取当前pc指针的地址，其低24bit，存放与r1中
44 @@ 2、获取_TEXT_BASE（CONFIG_SYS_TEXT_BASE）地址，也就是uboot代码段的链接地址，后续在uboot篇的时候会说明，并屏蔽其低24bit
45 @@ 3、如果相等的话，就跳过初始化的部分
46
47     /* init system clock */
48     bl system_clock_init
49 @@ 初始化系统时钟，后续有研究一下具体怎么配置的
50
51     /* Memory initialization */
52     bl mem_ctrl_asm_init
53 @@ 重点注意：在这里初始化！！后续会写一篇文章说明一下s5pv210平台如何初始化DDR。
54
55 1:
56     /* for UART */
57     bl uart_asm_init
58 @@ 串口初始化，到这里串口输出一个'O'字符，后续通过写字符到UTXH_OFFSET寄存器中，就可以在串口上输出相应的字符。
59
60     bl tzpc_init
61
62     #if defined(CONFIG_NAND)
63     /* simple init for NAND */
64     bl nand_asm_init
65 @@ 简单地初始化一下NAND flash，有可能BL2的镜像是在nand flash上面的。
66 #endif
67
68     /* Print 'K' */
69     ldr r0, =ELFIN_UART_CONSOLE_BASE
70     ldr r1, =0x4b4b4b4b
71     str r1, [r0, #UTXH_OFFSET]
72 @@ 再串口上打印'K'字符，表示lowlevel_init已经完成
73
74     pop {pc}
75 @@ 弹出PC指针，即返回。

```

当串口中打印出'OK'的字符的时候，说明lowlevel_init已经执行完成。

system_clock_init是初始化时钟的地方。mem_ctrl_asm_init这个函数是初始化DDR的地方。后续应该有研究一下这两个函数。这里先有个印象。

6、_main

spl的main的主要目标是调用board_init_f进行先前的板级初始化动作，在tiny210中，主要设计为，加载BL2到DDR上并且跳转到BL2中。DDR在上述lo已经初始化好了。

由于board_init_f是以C语言的方式实现，所以需要先构造C语言环境。

注意：uboot-spl和uboot的代码是通用的，其区别就是通过CONFIG_SPL_BUILD宏来进行区分的。

所以以下代码中，我们只列出spl相关的部分，也就是被CONFIG_SPL_BUILD包含的部分。

arch/arm/lib/crt0.S

```

1  ENTRY(_main)
2
3  /*
4   * Set up initial C runtime environment and call board_init_f(0).
5   */
6  @ 注意看这里的注释，也说明了以下代码的主要目的是，初始化C运行环境，调用board_init_f。
7     ldr sp, =(CONFIG_SPL_STACK)
8     bic sp, sp, #7 /* 8-byte alignment for ABI compliance */
9     mov r0, sp
10    bl board_init_f_alloc_reserve
11    mov sp, r0
12    /* set up gd here, outside any C code */
13    mov r9, r0
14    bl board_init_f_init_reserve
15
16    mov r0, #0

```

```

17     bl    board_init_f
18
19  ENDPROC(_main)

```

1

代码拆分如下：

(1) 因为后面是C语言环境，首先设置堆栈

写评论

```

1     ldr sp, =(CONFIG_SPL_STACK)
2  @@ 设置堆栈为CONFIG_SPL_STACK
3
4     bic sp, sp, #7 /* {alignment for ABI compliance */
5  @@ 堆栈是8字节对齐, 2^7bit 收藏 byte=8byte
6
7     mov r0, sp
8  @@ 把堆栈地址存放到r0寄存器中, 微信

```

关于CONFIG_SPL_STACK，我们前面的文章《[project X] tiny210(s5pv210)上电启动流程 (BL0-BL2)》

我们已经知道s5pv210的BL1(spl)在IRAM的，并且IRAM的地址空间是0xD002_0000-0xD003_7FFF，IRAM前面的部分放的是BL1的代码部分，最后的空间用来当作堆栈。

QQ

所以CONFIG_SPL_STACK定义如下：

include/configs/tiny210.h

```

1  #define CONFIG_SPL_STACK 0xD0037FFF

```

注意：上述还不是最终的堆栈地址，只是暂时的堆栈地址！！

(2) 为GD分配空间

```

1     bl    board_init_f_alloc_reserve
2  @@ 把堆栈的前面一部分空间分配给GD使用
3
4     mov sp, r0
5  @@ 重新设置堆栈指针SP
6
7     /* set up gd here, outside any C code */
8     mov r9, r0
9  @@ 保存GD的地址到r9寄存器中

```

注意：虽然sp的地址和GD的地址是一样的，但是堆栈是向下增长的，而GD则是占用该地址后面的部分，所以不会有冲突的问题。

关于GD，也就是struct global_data，可以简单的理解为uboot的全局变量都放在了这里，比较重要，所以后续会有会写篇文章说明一下global_data。这道在开始C语言环境的时候需要先为这个结构体分配空间。

board_init_f_alloc_reserve实现如下

common/init/board_init.c

```

1  ulong board_init_f_alloc_reserve(ulong top)
2  {
3     /* Reserve early malloc arena */
4     /* LAST : reserve GD (rounded up to a multiple of 16 bytes) */
5     top = rounddown(top-sizeof(struct global_data), 16);
6     // 现将top (也就是r0寄存器, 前面说过存放了暂时的指针地址), 减去sizeof(struct global_data), 也就是预留出一部分空间给sizeof(struct global_data)
7     // rounddown表示向下16个字节对其
8
9     return top;
10 // 到这里, top就存放了GD的地址, 也是SP的地址
11 //把top返回, 注意, 返回后, 其实还是存放在了r0寄存器中。
12 }

```

还有一点，其实GD在spl中没什么使用，主要是用在uboot中，但在uboot中的时候还需要另外分配空间，在讲述uboot流程的时候会说明。

(3) 初始化GD空间

前面说了，此时r0寄存器存放了GD的地址。

```
1 bl board_init_f_init_reserve
```

board_init_f_init_reserve实现如下
common/init/board_init.c

编译SPL的时候_USE_MEMCPY5 打开，所以我们去掉了_USE_MEMCPY的无关部分。

```
1 void board_init_f_init_reserve(uulong base)
2 {
3     struct global_data *gd_ptr;
4     int *ptr;
5     /*
6      * clear GD entirely and set it up.
7      * Use gd_ptr, as gd_ptr not be properly set yet.
8      */
9
10    gd_ptr = (struct global_data *)base;
11    // 从r0获取GD的地址
12    /* zero the area */
13    for (ptr = (int *)gd_ptr; ptr < (int *) (gd_ptr + 1); )
14        *ptr++ = 0;
15    // 对GD的空间进行清零
16 }
```

(4) 跳转到板级前期的初始化函数中
如下代码

```
1 bl board_init_f
```

board_init_f需要由板级代码自己实现。

在这个函数中，tiny210主要是实现了从SD卡上加载了BL2到ddr上，然后跳转到BL2的相应位置上
tiny210的实现如下：

board/samsung/tiny210/board.c

```
1 #ifdef CONFIG_SPL_BUILD
2 void board_init_f(uulong bootflag)
3 {
4     __attribute__((noreturn)) void (*uboot)(void);
5     int val;
6     #define DDR_TEST_ADDR 0x30000000
7     #define DDR_TEST_CODE 0xaa
8     tiny210_early_debug(0x1);
9     writel(DDR_TEST_CODE, DDR_TEST_ADDR);
10    val = readl(DDR_TEST_ADDR);
11    if(val == DDR_TEST_CODE)
12        tiny210_early_debug(0x3);
13    else
14    {
15        tiny210_early_debug(0x2);
16        while(1);
17    }
18    // 先测试DDR是否完成
19
20    copy_bl2_to_ddr();
21    // 加载BL2的代码到ddr上
22
23    uboot = (void *)CONFIG_SYS_TEXT_BASE;
24    // uboot函数设置为BL2的加载地址上
25    (*uboot)();
26    // 调用uboot函数，也就跳转到BL2的代码中
27 }
28 #endif
```

关于copy_bl2_to_ddr的实现，也就是如何从SD卡或者nand flash上加载BL2到DDR上的问题，请参考后续文章《[project X] tiny210(s5pv210)代码加载访

到此，SPL的任务就完成了，也已经跳到了BL2也就是uboot里面去了。

文章标签：

u-boot

spl

个人分类：

uboot

相关热词：

uboot的功能

uboot下命令

写评论

命令行

uboot剪裁

uboot组成

上一篇

[uboot]（第二章）uboot流程

目录

uboot-spl编译流程

下一篇

[project X] tiny210(s5pv210)

设备加载代码到DDR

收藏

想对作者说点什么？

我来说两句

u-boot之SPL分析

SPL SPL是uboot第一阶段执行的代码。负责搬移uboot第二阶段的代码到内存中运行。SPL是由固化在芯片内部的ROM引导的。我们知道很多芯片厂商固化的ROM支持从n...
tuwenqi2013 2017-01-10 11:13:uu 阅读数：1383

U-BOOT移植过程详解: SPL

U-BOOT移植过程详解: SPL 分类： U-BOOT移植 2014-01-21 18:06 641人...
linuxarmsummary 2015-04-02 19:43:10 阅读数：14460

uboot启动流程 - CSDN博客

在前面的文章中虽然已经说明了,在spl的阶段中已经对arch级进行了初始化了,为什么uboot里面还要对arch再初始化一遍? 回答:spl对于启动uboot来说并不是必须的,在...
2018-6-6

[uboot] (第二章)uboot流程——uboot-spl编译流程 - CSDN博客

建议先看《[project X] tiny210(s5pv210)上电启动流程(BL0-BL2)》,根据例子...最终编译完成之后,会在project-x/build/out/u-boot/spl下生成如下文件: arch...
2018-5-17

ARM U-Boot SPL过程浅析

原文地址：http://bbs.chinaunix.net/thread-4248378-1-1.html 【1】SPL简介 SPL (Secondary programloader) 是...
jxgz_leo 2016-08-02 22:45:10 阅读数：1433

新版UBOOT启动流程 - CSDN博客

1.3 启动代码相关的几个文件在u-boot中的路径 start.S: /arch/arm/cpu/arm...* (OMAP4 spl TEXT_BASE is not 32 byte aligned. * Continue to use ROM...
2018-6-5

AM335x启动流程(BootRom->MLO->Uboot) - CSDN博客

http://blog.chinaunix.net/uid-28458801-id-3486399.html 参考文件: 1,AM335x ARM Cortex-A8 Microprocessors (MPUs) Technical Reference Manual.pdf; 2,am...
2018-5-22

uboot 2016.05编译uboot.bin和spl

1：Makefile中的all目标编译出相应的文件。我们来看看这个all目标 all: \$(ALL-y) 2：# Always append ALL so that...
michaelcao1980 2016-10-25 10:54:40 阅读数：1059

u-boot中SPL源代码分析

u-boot SPL源代码分析。使用Atmel sama5d3xek做为例子进行分析。源代码：https://github.com/voiceshen/u-boot/tree/sama5d3xek...
voice_shen 2013-12-17 11:59:21 阅读数：15389

https://blog.csdn.net/ooonebook/article/details/52957395

8/13

关于uboot 中nand_spl 启动 - CSDN博客

最近有碰到uboot 中nand_spl启动,说说 1 解! 写在前面。 1.从名字上来看,他肯定和nand有关系的。spl是什么,我网查了一下,没找到,姑且理解为second ...

2018-6-2

写评论

ARM U-Boot SPL过程浅析 - CSDN博客


在最新版本的uboot中,可以看到SPL也有 nandflash,SDCARD等多种启动方式。当SPL本身被搬移到内部RAM中运行时,它将从nandflash、 SDCARD等外部介质中搬移uboot

2018-6-5

收藏

uboot之CONFIG_SPL_BOARD

首先进行第一步,下载工作:输入 U-Boot 的地址,找到自己要下载的 U-Boot 版本, 点击开始下载,下载完成之后开始解压。 U-Boot 下载之后压缩包的压缩方式是.tar.

 michaelcao1980 2015-03-18 10:00 阅读数: 4443

微博

u-boot SPL的理解 - CSDN博客

uboot分为uboot-spl和uboot两个组成部分。SPL是Secondary Program Loader的简称...是相对于SOC中的BROM来说的,之前的文章已经有所介绍,SOC启动最先执行的是BRC

2018-6-17


uboot启动过程 - CSDN博客

编译u-boot 后,会生成三个 u-boot-spl.bin、 MLO、 u-boot.img 镜像。 u-boot-spl.bin 和 MLO 都是第二级 bootloader,不同在于 u-boot-spl.bin 用于串...

2018-6-7


uboot spl分析

芯片到uboot启动流程 ROM → SPL→ uboot.img 简介 在IC中ROM code是第一级的bootlader。mpu上电后将会自动执行这里的代码,完成部分初始化和...

 u010402372 2013-10-17 16:44:48 阅读数: 1286

u-boot SPL的理解

uboot分为uboot-spl和uboot两个组成部分。SPL是Secondary Program Loader的简称,第二阶段程序加载器,这里所谓的第二阶段是相对于SOC中的BROM来说的,之前的..

 rikeyone 2016-06-12 14:49:31 阅读数: 3725


芯片到uboot启动流程 :ROM → MLO(SPL)→ uboot.img - CSDN博客

第三级 bootloader:uboot.img,C代码的入口。 其中第一级 bootloader 是板子固化的,第二级和第三级是通过编译 uboot 所得的。 2,第二级 bootloader:MLO(SPL)做...

2018-6-9

uboot2015第一阶段---SPL

uboot-spl

 u010243305 2017-01-21 13:50:34 阅读数: 363


u-boot-2014.10移植第29天----nand flash的SPL启动 (一)

前面在移植nand flash启动时做了很多探索性的工作,但是后来发现在relocate.S文件中调用的函数中有调用大部分的库函数,牵扯到的文件较多,很难将它们——包含到前面去。正在想其他方...

 sonbai 2015-03-29 16:27:24 阅读数: 4157

uboot SPL

点击打开链接

 linkedin_35878439 2016-10-25 16:01:49 阅读数: 123

uboot SPL Overview

了解下新uboot的SPL 原文地址：http://blog.csdn.net/abc47bca/article/details/6306005 Introduction: ===== ...

 lizhiguo0532 2012-05-10 20:25:44 阅读数：4424

芯片到uboot启动流程：ROM → MLO(SPL)→ uboot.img

AM335x 中bootloader被分成了 3 个部分：一级 bootloader：引导加载程序，板子上电后会自动执行这些代码，如选择哪种方式启动（NAND，SDcard，UART。。。）...

 hushup 2014-03-07 12:25:13 阅读数：3180

3-uboot-spl代码流程

[uboot] （第三章）uboot流程——uboot-spl代码流程2016年10月28日 16:24:14阅读数：2077以下例子都以project X项目tiny210（s5pv210平台,armv7架构）为例

 u013165704 2018-05-19 15:15:00 阅读数：14

Uboot 下board_init_f调用集合

前面我们讨论了board_init_f调用的调用过程，此函数主要是对init_sequence_f中的函数进行回调。common/board_f.c static const init_fnc_t ...

 sunlei0625 2017-03-16 08:34:30 阅读数：1170

U-BOOT的两个阶段启动过程与第二阶段的board_init_f和board_init_r

U-BOOT的两个阶段启动过程：（2010.06经典版来说）第一阶段：start.S的路径位于arch/arm/cpu/arm920t这段汇编代码一般被称作第一阶段初始化代码。主要作用是初始化和设置CPU寄存器。

 pugu12 2015-07-22 22:29:22 阅读数：2979

ARM-Linux嵌入式系统启动流程

学习嵌入式

 zy812248258 2014-09-03 12:58:05 阅读数：3832

Uboot 2017.01 启动流程分析

前言2017.01 UBoot包含两个阶段的启动，一个是SPL启动，一个是正常的启动我们称为第二阶段Uboot。当然，我们也可以选择使用SPL和不使用。在编译的过程中，是先编译boot，然后编译uboot。

 kl1125290220 2017-12-01 10:29:30 阅读数：6810

uboot启动过程详解

在android启动过程中，首先启动的便是uboot，uboot是负责引导内核装入内存启动或者是引导recovery模式的启动。现在在很多android的uboot的启动过程中，都需要对内核进行烧录。

 hongbochen1223 2017-04-15 17:50:05 阅读数：2752

u-boot怎样生成spl

u-boot怎样生成spl u-boot版本:2016.05 顶层Makefile定义生成spl: 生成spl需要的BOARD,CPU等变量的由来: 顶层Makefile包含了....

 luoqindong 2017-01-16 19:35:33 阅读数：970

[uboot] （第二章）uboot流程——uboot-spl编译流程

以下例子都以project X项目tiny210（s5pv210平台,armv7架构）为例[uboot] uboot流程系列：[project X] tiny210(s5pv210)上电启动流程（...）

 oonebook 2016-10-27 20:40:47 阅读数：3114

uboot配置和编译过程详解

uboot主Makefile分析1 1、uboot version确定（Makefile的24-29行）(1)uboot的版本号分3个级别：VERSION：主版本号 PATCHLEVEL：次版本号... SUBLEVEL：修订版本号

 czg13548930186 2016-12-03 14:27:05 阅读数：7628

uboot流程——uboot-spl代码流程

来自：https://www.cnblogs.com/leaven/p/6296160.html公告呢称：海王园龄：8年3个月粉丝：241关注：0+加关注日历⁢2018年3月&g...</p>
</div>
<div data-bbox="43 969 957 981" data-label="Page-Footer"><div>https://blog.csdn.net/oonebook/article/details/52957395</div><div>10/13</div>
</div>

 wlf_go

2018-03-28 16:42:44

阅读数：160

UBOOT 学习心得 (UBOOT启动流程分析)

网上找到的UBOOT研究文章，结合自己写的。目前是明白了UBOOT主干程序流程了。开始分析细节部分了。下面是别人写的UBOOT分析。参考了fzb和赵春江两位大牛了2010.06版本的...

 windsun0800

2013-12-14 01:25

阅读数：5661

[uboot] （第一章）uboot流程——概述

[uboot] uboot流程系列： [project X] tiny210(s5pv210)上电启动流程 (BL0-BL2) 建议先看 《[project X] tiny210(s5pv210)上电启动...

 oonebook

2016-10-26 22:30:4

阅读数：2946

UBoot命令解析与执行流程

本文为CP根据网上内容以及U-Boot 2014.07版本进行整理而成。原文地址为<http://blog.chinaunix.net/uid-8867796-id-358806.html> ——...

 jiujiaobusinao

2016-11-25 16:0

阅读数：1539

uboot启动流程详解(2)-reset

1、cpsr寄存器介绍 通过向模式位M[4:0]里写入相应的数据切换到不同的模式，在对CPSR，SPSR寄存器进行操作不能使用mov，ldr等通用指令，只能使用特权指令msr

 silent123go

2016-11-12 18:58:01

阅读数：925

uboot的eMMC初始化代码流程分析

源码参考九鼎科技移植的X210开发板捆绑BSP中的uboot, 版本为1.3.4mmc初始化函数int mmc_initialize(bd_t *bis)在uboot/lib_arm/board.c中...

 harrydotter_wh

2017-09-14 14:36:44

阅读数：955

uboot启动流程和架构

概述： 本文将从两个方面来阐述uboot: 1、启动流程 2、架构一、uboot流程图： 从上图中看到红色1,2,3,4,5,7,8,9的标号，下面分别说明每个过程： ...

 eZiMu

2017-02-01 15:52:04

阅读数：911

史上最详细最全面的uboot启动过程分析

2015年04月30日

324KB

下载

uboot之Makefile编译过程详解

1.主Makefile分析： uboot的version（版本信息）： VERSION = 1 PATCHLEVEL = 3 SUBLEVEL = 4 EXTRAVERSION =

 qq_25827755

2016-12-15 16:30:24

阅读数：294

uboot学习笔记之uboot1.3.4一移植

自己学习嵌入式学习已经有一段时间了，感觉不懂的太多，在这里想把每一步学习的只是一步记录下来，以供自己以后查看使用，也希望看到这篇文章的朋友多给点意见记录一下，uboot1.3.4部分， ...

 dghfj

2016-06-05 00:50:11

阅读数：640

uboot到kernel启动过程中内存布局变化、初始化

内核编译（make）之后会生成两个文件，一个是Image，一个是zImage，其中Image为内核映像文件，而zImage为内核的一种映像压缩文件。ulImage是uboot专用的映像文件...

 taochao90

2018-04-18 11:26:15

阅读数：73

Uboot启动流程和Kernel启动流程

/******Uboot启动流程（分为两部分）*****/ 第一部分（放在start.s中，汇编）1).定义入口（通过链接器脚本...

 zhangsan_3

2016-11-27 16:31:47

阅读数：827

uboot分析.pdf

1

2014年03月03日

194KB

下载

写评论

uboot启动流程分析之一

目录

最开始的就是start.S 一个可执行的

必须有一个入口点并且只能有一个唯一的全局入口，通常这个入口放在Rom(flash)的0x0地址。 start.S · _st...

 Stars_Moon_Sky

2015-04-22 1

阅读数：1312

收藏

U-boot初始化阶段流程分析

微信

U-boot的初始化主要分为两个阶段 第一

主要是SOC内部的初始化，板级的初始化比较少，所以移植的修改量比较小。此阶段由汇编语言编写，代码主体分布在start.S和t...

 qq_28992301

2016-07-06 21:3

微博 阅读数：2105


深入理解uboot 2016 - 基础

QQ

处理器启动流程分析)

最近一段时间一直在做uboot相关的移植的工作，需要将uboot-2016-7移植到ARMv7的处理器上。正好元旦放假三天闲来无事，有段完整的时间来整理下最近的工作成果。之

时，在网上...


 kernel_yx

2016-11-05 14:52:35

阅读数：8436

[RK3288][Android6.0] U-boot显示模块部分流程小结

Platform: RK3288 OS: Android 6.0 Version: v2014.10 //mipi dsi接口为例: drv_lcd_init -> lcd.c lc...

 kris_fei

2016-10-09 17:36:39

阅读数：1889

LCD 在uboot和Kernel中的基本流程


转自http://blog.csdn.net/li_heaven1/article/details/9786431 一、LCD显示的基本原理 通过framebuffer，应用程序用mmap把显...

 qwaszx523

2016-09-01 14:45:24

阅读数：2900

个人资料

 ooonebook

关注

原创	粉丝	喜欢	评论
48	145	26	17

等级：

博客

4

访问：11万+

积分：1644

排名：3万+

勋章：

恒

最新文章

- [sd card] mmc硬件总线扫描流程（以sd card为例）
- [emmc] emmc总线设置
- [sd card] mmc_blk层为sd card创建块设备流程
- [sd card] sd card块设备(mmc_blk)读写流程学习笔记
- [sd card] sd card初始化流程

1

写评论

目录

收藏

微信

微博

QQ

个人分类

console	2篇
kernel启动流程	8篇
project-X	5篇
uboot	15篇
mmc	17篇

归档

2017年3月	6篇
2017年2月	12篇
2016年12月	3篇
2016年11月	9篇
2016年10月	12篇

展开

热门文章

[uboot] uboot启动kernel篇（二）——boot m跳转到kernel的流程
阅读量：6901

[uboot]（番外篇）uboot 驱动模型
阅读量：6830

[uboot]（第六章）uboot流程——命令行模式以及命令处理介绍
阅读量：4720

[emmc] emmc总线设置
阅读量：4600

[uboot] uboot启动kernel篇（一）——Legacy-ulmage & FIT-ulmage
阅读量：4433

最新评论

[project X] tiny2...
baidu_26866585：[reply]woshidahuidan2011[/reply] CPU0运行，其他核心睡眠等待...

[project X] tiny2...
woshidahuidan2011：写的不错， 有没有想过：“s5pv210上电之后，CPU会直接从0x0地址取指令，也就是直接执行...

[emmc] emmc总线设置
weixin_41711181：楼主，您好，请问你的EMMC数据资料是从哪下载的？

[emmc] emmc总线设置
kingman_0717：你好，请问在52MHz的DDR模式下，能否把数据总线电压设置为3v呢？我的VCCQ是3.3v，使用...

[uboot]（番外篇）uboo...
chenxiaozen22：感谢博主的讲解，作为一个应届生都能大概跟上博主的讲解节奏。学到很多东西了。