

原

[uboot] （第二章）uboot流程——uboot-spl编译流程

2016年10月27日 20:40:47

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）

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、uboot-spl编译和生成文件

spl的编译是编译uboot的一部分，和uboot.bin走的是两条编译流程，这个要重点注意。  
正常来说，会先编译主体uboot，也就是uboot.bin.再编译uboot-spl，也就是uboot-spl.bin,虽然编译命令是一起的，但是编译流程是分开的。

1、编译方法

在project X项目中，所有镜像，包括uboot、kernel、rootfs都是放在build目录下进行编译的。具体去参考该项目build的Makefile的实现。  
假设config已经配置完成，在build编译命令如下：

```
1 make uboot
```

Makefile中对应的命令如下：

```
1 BUILD_DIR=$(shell pwd)
2 OUT_DIR=$(BUILD_DIR)/out
3 UBOOT_OUT_DIR=$(OUT_DIR)/u-boot
4 UBOOT_DIR=$(BUILD_DIR)/../u-boot
5 uboot:
6     mkdir -p $(UBOOT_OUT_DIR)
7     make -C $(UBOOT_DIR) CROSS_COMPILE=$(CROSS_COMPILE) KBUILD_OUTPUT=$(UBOOT_OUT_DIR) $(BOARD_NAME)_defconfig
8     make -C $(UBOOT_DIR) CROSS_COMPILE=$(CROSS_COMPILE) KBUILD_OUTPUT=$(UBOOT_OUT_DIR)
9     ## -C $(UBOOT_DIR) 指定了要在../uboot，也就是uboot的代码根目录下执行make
10    ## CROSS_COMPILE=$(CROSS_COMPILE) 指定了交叉编译器
11    ## KBUILD_OUTPUT=$(UBOOT_OUT_DIR) 指定了最终编译的输出目录是build/out/u-boot.
```

最终，相当于进入了uboot目录执行了make动作。  
也就是说spl的编译是编译uboot的一部分，和uboot.bin走的是两条编译流程，这个要重点注意。  
正常来说，会先编译主体uboot，也就是uboot.bin.再编译uboot-spl，也就是uboot-spl.bin,虽然编译命令是一起的，但是编译流程是分开的。

2、生成文件

最终编译完成之后，会在project-x/build/out/u-boot/spl下生成如下文件：

```
1 arch common dts include u-boot-spl u-boot-spl.cfg u-boot-spl.map
2 board drivers fs tiny210-spl.bin u-boot-spl.bin u-boot-spl.lds u-boot-spl-nodtb.bin
```

其中，arch、common、dts、include、board、drivers、fs是对应代码的编译目录，各个目录下都会生成相应的built.o，是由同目录下的目标文件连接而  
重点说一下以下几个文件：

文件	说明
u-boot-spl	初步链接后得到的spl文件
u-boot-spl-nodtb.bin	在u-boot-spl的基础上，经过objcopy去除符号表信息之后的可执行程序
u-boot-spl.bin	在不需要dtb的情况下，直接由u-boot-spl-nodtb.bin复制而来，也就是编译spl的最终目标
tiny210-spl.bin	由s5pv210平台决定，需要在u-boot-spl.bin的基础上加上16B的header用作校验
u-boot-spl.lds	spl的连接脚本
u-boot-spl.map	连接之后的符号表文件
u-boot-spl.cfg	由spl配置生成的文件

## 二、uboot-spl编译流程

### 1、编译整体流程

根据零、2生成的文件说明可知简

（1）各目录下built-in.o的生成

对应二、2（5）的实现

（2）由所有built-in.o以u-boot-spl.lds为连接脚本通过连接来生成u-boot-spl

对应二、2（4）的实现

（3）由u-boot-spl生成u-boot-spl-nodtb.bin

对应二、2（3）的实现

（4）由u-boot-spl-nodtb.bin生成u-boot-spl.bin，也就是spl的bin文件

对应二、2（2）的实现

后续的编译的核心过程就是按照上述的四个编译流程就是按照上述四个步骤来的。

## 2、具体编译流程分析

我们直接从make uboot命令分析，也就是从uboot下的Makefile的依赖关系来分析整个编译流程。注意，这个分析顺序和上述的整体编译流程的顺序是反着的。

### （1）入口分析

在project-x/u-boot/Makefile中

```
1 all: $(ALL-y)
2
3 ALL-$(CONFIG_SPL) += spl/u-boot-spl.bin
4 ## 当配置了CONFIG_SPL，make的时候就会执行spl/u-boot-spl.bin这个目标
5
6 spl/u-boot-spl.bin: spl/u-boot-spl
7 @:
8 spl/u-boot-spl: tools prepare $(if $(CONFIG_OF_SEPARATE),dts/dt.dtb)
```

```

9      $(Q)$(MAKE) obj=spl -f $(srctree)/scripts/Makefile.spl all
10  ## obj=spl 会在out/u-boot目录下生成spl目录
11  ## -f $(srctree)/scripts/Makefile.spl 说明执行的Makefile文件是scripts/Makefile.spl
12  ## $(MAKE) all 相当于make -j 2 就是all

```

综上，由CONFIG\_SPL来决定是否编译出spl文件，也就是BL1。

后续相当于执行了“`make -f u-boot/scripts/Makefile.spl obj=spl all`”命令。

在project-x/u-boot/scripts/Makefile.spl中，

```

1  SPL_BIN := u-boot-spl
2  ALL-y += $(obj)/$(SPL_BIN).bin $(obj)/$(SPL_BIN).cfg
3  ## 所以最终目标是spl/u-boot-spl.bin和spl/u-boot-spl.cfg

```

在project-x/u-boot/scripts/Makefile.spl中建立了spl/u-boot-spl.bin的依赖关系，后续make过程的主体都是在Makefile.spl中。

#### • (2) spl/u-boot-spl.bin的依赖关系

在project-x/u-boot/scripts/Makefile.spl中

```

1  $(obj)/$(SPL_BIN).bin: $(obj)/$(SPL_BIN)-nodtb.bin FORCE
2      $(call if_changed,copy)
3  ## $(obj)/$(SPL_BIN).bin依赖于$(obj)/$(SPL_BIN)-nodtb.bin。
4  ## $(call if_changed,copy)表示当依赖文件发生变化时，直接把依赖文件复制为目标文件，即直接把$(obj)/$(SPL_BIN)-nodtb.bin复制为$(obj)/$(SPL_BIN).bin

```

如上述Makefile代码spl/u-boot-spl.bin依赖于spl/u-boot-spl-nodtb.bin，并且由spl/u-boot-spl-nodtb.bin复制而成。

对应于上述二、1 (4) 流程。

#### • (3) spl/u-boot-spl-nodtb.bin的依赖关系

在project-x/u-boot/scripts/Makefile.spl中

```

1  $(obj)/$(SPL_BIN)-nodtb.bin: $(obj)/$(SPL_BIN) FORCE
2      $(call if_changed,objcopy)
3  $(obj)/$(SPL_BIN)-nodtb.bin依赖于$(obj)/$(SPL_BIN)。也就是spl/u-boot-spl-nodtb.bin依赖于spl/u-boot-spl。
4  ## $(call if_changed,objcopy)表示当依赖文件发生变化时，将依赖文件经过objcopy处理之后得到目标文件。
5  ## 也就是通过objcopy把spl/u-boot-spl的符号信息以及一些无用信息去掉之后，得到了spl/u-boot-spl-nodtb.bin。

```

如上述Makefile代码spl/u-boot-spl-nodtb.bin依赖于spl/u-boot-spl，并且由spl/u-boot-spl经过objcopy操作之后得到。

对应于上述二、1 (3) 流程。

#### • (4) spl/u-boot-spl的依赖关系

在project-x/u-boot/scripts/Makefile.spl中

```

1  $(obj)/$(SPL_BIN): $(u-boot-spl-init) $(u-boot-spl-main) $(obj)/u-boot-spl.lds FORCE
2      $(call if_changed,u-boot-spl)
3  ## $(call if_changed,u-boot-spl)来生成目标
4  ## $(call if_changed,u-boot-spl)对应cmd_u-boot-spl命令

```

如上，spl/u-boot-spl依赖于\$(u-boot-spl-init)、\$(u-boot-spl-main)和spl/u-boot-spl.lds，并且最终会调用cmd\_u-boot-spl来生成spl/u-boot-spl。

cmd\_u-boot-spl实现如下：

```

1  quiet_cmd_u-boot-spl = LD      $@
2      cmd_u-boot-spl = (cd $(obj) && $(LD) $(LDFLAGS) $(LDFLAGS_$(@F)) \
3                      $(patsubst $(obj)/%,%, $(u-boot-spl-init)) --start-group \
4                      $(patsubst $(obj)/%,%, $(u-boot-spl-main)) --end-group \
5                      $(PLATFORM_LIBS) -Map $(SPL_BIN).map -o $(SPL_BIN))

```

将cmd\_u-boot-spl通过echo命令打印出来之后得到如下（拆分出来看的）：

```

1 cmd_u-boot-spl=(
2 cd spl &&
3 /build/arm-none-linux-gnueabi-4.8/bin/arm-none-linux-gnueabi-ld
4 -T u-boot-spl.lds --gc-sections -Bstatic --gc-sections
5 arch/arm/cpu/armv7/start.o
6 --start-group
7 arch/arm/mach-s5pc1xx/built-in.o arch/arm/cpu/armv7/built-in.o arch/arm/cpu/built-in.o arch/arm/lib/built-in.o board/samsung/tiny21
8 --end-group
9 arch/arm/lib/eabi_compat.o
10 -L /home/disk3/xys/temp/project-x/build/arm-none-linux-gnueabi-4.8/bin/../lib/gcc/arm-none-linux-gnueabi/4.8.3 -lgcc
11 -Map u-boot-spl.map
12 -o u-boot-spl)

```

可以看出上述是一条连接命令，以 `u-boot-spl.lds` 为链接脚本，把 `$(u-boot-spl-init)`、`$(u-boot-spl-main)` 的指定的目标文件连接到 `u-boot-spl` 中。连接很重要的东西就是连接标识，它是 `$(LD) $(LDFLAGS) $(LDFLAGS_$(@F))` 的定义。

尝试把 `$(LD) \$(LDFLAGS) \$(LDFLAGS_$(@F))` 打印出来，结果如下：

```

1 LD=/home/disk3/xys/temp/project-x/build/arm-none-linux-gnueabi-4.8/bin/arm-none-linux-gnueabi-ld
2 LDFLAGS=
3 LDFLAGS_u-boot-spl=-T u-boot-spl.lds --gc-sections -Bstatic --gc-sections
4 ## $(LDFLAGS_$(@F)) 对应于 LDFLAGS_u-boot-spl

```

也就是说在 `LDFLAGS_u-boot-spl` 中指定了链接脚本。

重点关注 `$(LDFLAGS_$(@F))` 的由来

```

1 ## @F 是一个自动化变量，提取目标的文件名，比如目标是 $(obj)/$(SPL_BIN)，也就是 spl/u-boot-spl，那么 @F 就是 u-boot-spl。
2 ## 所以 LDFLAGS_$(@F) 就是 LDFLAGS_u-boot-spl
3 ## 定义如下
4 LDFLAGS_$(SPL_BIN) += -T u-boot-spl.lds $(LDFLAGS_FINAL)
5 ifneq ($(CONFIG_SPL_TEXT_BASE),)
6 LDFLAGS_$(SPL_BIN) += -Ttext $(CONFIG_SPL_TEXT_BASE)
7 endif
8 ## 当指定 CONFIG_SPL_TEXT_BASE 时，会配置连接地址。在 tiny210 项目中，因为 spl 是地址无关代码设计，故没有设置连接地址。
9
10 ## $(LDFLAGS_FINAL) 在如下几个地方定义了
11 ## ./config.mk:19:LDFLAGS_FINAL :=
12 ## ./config.mk:80:LDFLAGS_FINAL += -Bstatic
13 ## ./arch/arm/config.mk:16:LDFLAGS_FINAL += --gc-sections
14 ## ./scripts/Makefile.spl:43:LDFLAGS_FINAL += --gc-sections
15 ## 综上：最后 LDFLAGS_u-boot-spl=-T u-boot-spl.lds --gc-sections -Bstatic --gc-sections 就可以理解了。
16 ## 对应于上述二、1（2）流程。

```

#### • (5) u-boot-spl-init & u-boot-spl-main 依赖关系 (代码是如何被编译的)

先看一下这两个值打印出来的

```

1 u-boot-spl-init=spl/arch/arm/cpu/armv7/start.o
2
3 u-boot-spl-main= spl/arch/arm/mach-s5pc1xx/built-in.o spl/arch/arm/cpu/armv7/built-in.o spl/arch/arm/cpu/built-in.o spl/arch/arm/li

```

可以观察到是一堆目标文件的路径。这些目标文件最终都要被连接到 `u-boot-spl` 中。

`u-boot-spl-init` & `u-boot-spl-main` 的定义如下代码：

`project-x/u-boot/scripts/Makefile.spl`

```

1 u-boot-spl-init := $(head-y)
2 head-y := $(addprefix $(obj)/,$(head-y))
3 ## 加 spl 路径
4 ## ./arch/arm/Makefile 定义了如下:
5 ## head-y := arch/arm/cpu/$(CPU)/start.o
6
7 u-boot-spl-main := $(libs-y)
8 libs-y += $(if $(BOARD_DIR),board/$(BOARD_DIR)/)
9 libs-y += $(HAVE_VENDOR_COMMON_LIB) += board/$(VENDOR)/common/

```

```

10  libs-$(CONFIG_SPL_FRAMEWORK) += common/spl/
11  libs-y += common/init/
12  libs-$(CONFIG_SPL_LIBCOMMON_SUPPORT) += common/cmd/
13  libs-$(CONFIG_SPL_LIBDISK_SUPPORT) += disk/
14  libs-y += drivers/
15  libs-y += dts/
16  libs-y += fs/
17  libs-$(CONFIG_SPL_LIBGEM_SUPPORT) += lib/
18  libs-$(CONFIG_SPL_POST_IMAGE_SUPPORT) += post/drivers/
19  libs-$(CONFIG_SPL_NETWORK_SUPPORT) += net/
20  libs-y := $(addprefix $(obj)/,$(libs-y))
21  ## 加spl路径
22  u-boot-spl-dirs := $(patsubst %/,,$(filter %/, $(libs-y)))
23  ## 这里注意一下, u-boot-spl-dirs是libs-y没有加built-in.o后缀的时候被定义的。
24  libs-y := $(patsubst %/,,$(filter %/, $(libs-y)))
25  ## 加built-in.o文件后缀

```

那么u-boot-spl-init & u-boot-spl-main是如何生成的呢？

需要看一下对应的依赖如下：

```

1  $(sort $(u-boot-spl-init) $(u-boot-spl-main)): $(u-boot-spl-dirs) ;
2  ## 也就是说$(u-boot-spl-init) $(u-boot-spl-main)依赖于$(u-boot-spl-dirs)
3  ## sort函数根据首字母进行排序并去除掉重复的。
4  ## u-boot-spl-dirs := $(patsubst %/,,$(filter %/, $(libs-y)))
5  ## $(filter %/, $(libs-y))过滤出 '/' 结尾的字符串，注意，此时$(libs-y)的内容还没有加上built-in.o文件后缀
6  ## patsubst去掉字符串中最后的 '/' 的字符。
7  ## 最后u-boot-spl-dirs打印出来如下：
8  ## u-boot-spl-dirs=spl/arch/arm/mach-s5pc1xx spl/arch/arm/cpu/armv7 spl/arch/arm/cpu spl/arch/arm/lib spl/board/samsung/tiny210 spl
9  ## 也就是从libs-y改造而来的。
10
11  ## $(u-boot-spl-dirs) 的依赖规则如下：
12  $(u-boot-spl-dirs):
13      $(Q)$(MAKE) $(build)=$@

```

也就是会对每一个目标文件依次执行make \$(build)=目标文件

\$(build)定义如下：

project-x/u-boot/scripts/Kbuild.include

```
1  build := -f $(srctree)/scripts/Makefile.build obj
```

以arch/arm/mach-s5pc1xx为例

“\$(MAKE) \$(build)=\$@”展开后格式如下

make -f ~/code/temp/project-x/u-boot/scripts/Makefile.build obj=spl/arch/arm/mach-s5pc1xx。

Makefile.build定义built-in.o、.lib以及目标文件.o的生成规则。这个Makefile文件生成了子目录的.lib、built-in.o以及目标文件.o。

Makefile.build第一个编译目标是\_\_build，如下

```

1  PHONY := __build
2  __build:
3  ## 所以会直接编译执行__build这个目标，其依赖如下
4  __build: $(if $(KBUILD_BUILTIN),$(builtin-target) $(lib-target) $(extra-y)) \
5           $(if $(KBUILD_MODULES),$(obj-m) $(modorder-target)) \
6           $(subdir-ym) $(always)
7  @:
8  ## 和built-in.o相关的是依赖builtin-target。下面来看这个依赖。
9
10 builtin-target := $(obj)/built-in.o
11 ## 以obj=spl/arch/arm/mach-s5pc1xx为例，那么builtin-target就是spl/arch/arm/mach-s5pc1xx/built-in.o.
12
13 ## 依赖关系如下：
14 $(builtin-target): $(obj-y) FORCE
15     $(call if_changed,link_o_target)
16 ## $(call if_changed,link_o_target)将所有依赖连接到$(builtin-target)，也就是相应的built-in.o中了。
17 ## 具体实现可以查看cmd_link_o_target的实现，这里不详细说明了。

```

```
18
19
20 ## 那么$(obj-y)是从哪里来      是从相应目录下的Makefile中include得到的。
21 # The filename Kbuild hi 2 cedence over Makefile
22 kbuild-dir := $(if $(filter-out %, $(src)), $(src), $(srctree)/$(src))
23 kbuild-file := $(if $(wildcard $(kbuild-dir)/Kbuild), $(kbuild-dir)/Kbuild, $(kbuild-dir)/Makefile)
24 include $(kbuild-file) 写评论
25 ## 当obj=spl/arch/arm/mach-s5pc1xx时,得到对应的kbuild-file=u-boot/arch/arm/mach-s5pc1xx/Makefile
26 ## 而在u-boot/arch/arm/mach-s5pc1xx/Makefile中定义了obj-y如下:
27 ## obj-y = cache.o 目录
28 ## obj-y += reset.o
29 ## obj-y += clock.o 收藏
30 ## 对应obj-y对应一些目标文件,由C文件编译而来,这里就不说明了。
```

对应于上述二、1（1）流程。

- （6）spl/u-boot-spl.lds依赖关系  
这里主要是为了找到一个匹配的链接脚本。

```
1 $(obj)/u-boot-spl.lds
2 $(obj)/u-boot-spl.lds: $(LDSCRIPT) FORCE
3     $(call if_changed_dep, cpp_lds)
4 ## 依赖于$(LDSCRIPT), $(LDSCRIPT)定义了连接脚本所在的位置,
5 ## 然后把链接脚本经过cpp_lds处理之后复制到$(obj)/u-boot-spl.lds中,也就是spl/u-boot-spl.lds中。
6 ## cpp_lds处理具体实现看cmd_cpu_lds定义, 具体是对应连接脚本里面的宏定义进行展开。
7
8 ## $(LDSCRIPT)定义如下
9 ifeq ($(wildcard $(LDSCRIPT)),)
10     LDSCRIPT := $(srctree)/board/$(BOARDDIR)/u-boot-spl.lds
11 endif
12 ifeq ($(wildcard $(LDSCRIPT)),)
13     LDSCRIPT := $(srctree)/$(CPUDIR)/u-boot-spl.lds
14 endif
15 ifeq ($(wildcard $(LDSCRIPT)),)
16     LDSCRIPT := $(srctree)/arch/$(ARCH)/cpu/u-boot-spl.lds
17 endif
18 ifeq ($(wildcard $(LDSCRIPT)),)
19 $(error could not find linker script)
20 endif
21 ## 也就是说依次从board/板级目录、cpudir目录、arch/架构/cpu/目录下去搜索u-boot-spl.lds文件。
22 ## 例如, tiny210(s5vp210 armv7)最终会在./arch/arm/cpu/下搜索到u-boot-spl.lds
```

综上，最终指定了project-X/u-boot/arch/arm/cpu/u-boot-spl.lds作为连接脚本。

### 三、一些重点定义

- 1、CONFIG\_SPL  
在二、2（1）中说明。  
用于指定是否需要编译SPL，也就是是否需要编译出uboot-spl.bin文件
- 2、连接脚本的位置  
在二、2（6）中说明。  
对于tiny210(s5vp210 armv7)来说，连接脚本的位置在  
project-x/u-boot/arch/arm/cpu/u-boot-spl.lds
- 3、CONFIG\_SPL\_TEXT\_BASE  
在二、2（4）中说明。  
用于指定SPL的连接地址，可以定义在板子对应的config文件中。
- 4、CONFIG\_SPL\_BUILD  
在编译spl过程中，会配置

project-x/scripts/Makefile.spl中定义了如下

```
1 KBUILD_CPPFLAGS += -DCONFIG_SPL_BUILD
2
```

也就是说在编译uboot-spl.bin的过程中CONFIG\_SPL\_BUILD这个宏是被定义的。

## 四、uboot-spl链接脚本分析

### 1、连接脚本整体分析

相对比较简单，直接看连接脚本的 [project-x/u-boot/arch/arm/cpu/u-boot-spl.lds](#) 前面有一篇分析连接脚本的文章了 [《\[uboot\]启动流程 前篇——vmlinux.lds分析》](#)，可以参考一下。

```
1 ENTRY(_start)
2 //定义了地址为_start的地址，我们分析代码就是从这个函数开始分析的!!!
3 SECTIONS
4 {
5     . = 0x00000000;
6
7     //以下定义文本段
8     . = ALIGN(4);
9     .text :
10    {
11        __image_copy_start = .;
12        //定义__image_copy_start这个标号地址为当前地址
13        *(.vectors)
14        //所有目标文件的vectors段，也就是中断向量表链接到这里来
15        CPUDIR/start.o (.text*)
16        //start.o文件的.text段链接到这里来
17        *(.text*)
18        //所有目标文件的.text段链接到这里来
19    }
20
21    //以下定义只读数据段
22    . = ALIGN(4);
23    .rodata : { *(SORT_BY_ALIGNMENT(SORT_BY_NAME(.rodata*))) }
24
25    //以下定义数据段
26    . = ALIGN(4);
27    .data : {
28        *(.data*)
29        //所有目标文件的.data段链接到这里来
30    }
31
32    //以下定义u_boot_list段，具体功能未知
33    . = ALIGN(4);
34    .u_boot_list : {
35        KEEP(*(SORT(.u_boot_list*)));
36    }
37
38    . = ALIGN(4);
39
40    __image_copy_end = .;
41    //定义__image_copy_end符号的地址为当前地址
42    //从__image_copy_start 到 __image_copy_end的区间，包含了代码段和数据段。
43
44    //以下定义rel.dyn段，这个段用于uboot资源重定向的时候使用，后面会说明
45    .rel.dyn : {
46        __rel_dyn_start = .;
47        //定义__rel_dyn_start 符号的地址为当前地址，后续在代码中会使用到
48        *(.rel*)
49        __rel_dyn_end = .;
50        //定义__rel_dyn_end 符号的地址为当前地址，后续在代码中会使用到
51        //从__rel_dyn_start 到 __rel_dyn_end 的区间，应该是在代码重定向的过程中会使用到，后续遇到再说明。
52    }
```

```
53
54     .end :
55     {
56         *(__end)
57     }
58
59     __image_binary_end =
60 //定义__image_binary_end 地址为当前地址
61
62 //以下定义bss段
63     .bss __rel_dyn_start : {
64         __bss_start = .;
65         *(__bss*)
66         . = ALIGN(4);
67         __bss_end = .;
68     }
69     __bss_size = __bss_end - __bss_start;
70     .dynsym __image_binary_end : { *(__dynsym) }
71     .dynbss : { *(__dynbss) }
72     .dynstr : { *(__dynstr) }
73     .dynamic : { *(__dynamic) }
74     .hash : { *(__hash) }
75     .plt : { *(__plt) }
76     .interp : { *(__interp) }
77     .gnu : { *(__gnu) }
78     .ARM.exidx : { *(__ARM.exidx) }
79 }
```

2、符号表中需要注意的符号

project-x/build/out/u-boot/spl/u-boot-spl.map

```
1  Linker script and memory map
2  .text          0x00000000      0xd10
3
4  *(.vectors)
5
6  0x00000000      _start
7  0x00000020      _undefined_instruction
8  0x00000024      _software_interrupt
9  0x00000028      _prefetch_abort
10 0x0000002c      _data_abort
11 0x00000030      _not_used
12 0x00000034      _irq
13 0x00000038      _fiq
14 ...
15 0x00000d10      __image_copy_end = .
16 .rel.dyn        0x00000d10      0x0
17 *(.rel*)
18 .rel.plt        0x00000000      0x0 arch/arm/cpu/armv7/start.o
19 0x00000d10      __rel_dyn_end = .
20 .end
21 *(__end)
22 0x00000d10      __image_binary_end = .
23
24 .bss            0x00000d10      0x0
25 0x00000d10      __bss_start = .
26 *(__bss*)
27 0x00000d10      . = ALIGN (0x4)
28 0x00000d10      __bss_end = .
```

重点关注

- \* \_\_image\_copy\_start & \_\_image\_copy\_end  
界定了代码的位置，用于重定向代码的时候使用，后面碰到了再分析。
- \* \_start



在u-boot-spl.lds中ENTRY(\_start)，也就规定了代码的入口函数是\_start。所以后续分析代码的时候就是从这里开始分析。

```
* __rel_dyn_start & __rel_dyn_end
*_image_binary_end
```

五、tiny210 ( s5pv210 ) 的额外操作

1、为什么tiny210的spl需要额外操作？需要什么额外操作？

建议先参考《[project X] tiny210(s5pv210)上电启动流程（BL0-BL2）》一文。  
tiny210只支持SD启动的方式和NAND FLASH启动的方式。  
从《[project X] tiny210(s5pv210)上电启动流程（BL0-BL2）》一文中，我们已经得知了当使用SD启动的方式和NAND FLASH启动的方式，也就是BL1执行文件，并没有加上特别的header。  
因此，我们需要在生成uboot-spl.bin后，再为其加上16B的header后生成tiny210-spl.bin。  
16B的header格式如下：

地址	数据
0xD002_0000	BL1镜像包括header的长度
0xD002_0004	保留，设置为0
0xD002_0008	BL1镜像除去header的校验和
0xD002_000c	保留，设置为0

2、如何生成header？（如何生成tiny210-spl.bin）

project-x/u-boot/scripts/Makefile.spl

```
1  ifdef CONFIG_SAMSUNG
2  ALL-y += $(obj)/$(BOARD)-spl.bin
3  endif
4  ## 当平台是SAMSUNG平台的时候，也就是CONFIG_SAMSUNG被定义的时候，就需要生成对应的板级spl.bin文件，例如tiny210的话，就应该生成对应的spl/tiny2
5
6  ifdef CONFIG_S5PC110
7  $(obj)/$(BOARD)-spl.bin: $(obj)/u-boot-spl.bin
8      $(objtree)/tools/mks5pc1xxspl $< $@
9  ## 如果是S5PC110系列的cpu的话，则使用如上方法打上header。tiny210的cpu是s5pv210的，属于S5PC110系列，所以走的是这路。
10 ## $(objtree)/tools/mks5pc1xxspl对应于编译uboot时生成的build/out/u-boot/tools/mks5pc1xxspl
11 ## 其代码路径位于u-boot/tools/mks5pc1xxspl.c，会根据s5pc1xx系列的header规则为输入bin文件加上16B的header，具体参考代码。
12 ## 这里就构成了u-boot-spl.bin到tiny210-spl.bin的过程了。
13 else
14 $(obj)/$(BOARD)-spl.bin: $(obj)/u-boot-spl.bin
15     $(if $(wildcard $(objtree)/spl/board/samsung/$(BOARD)/tools/mk$(BOARD)spl),\
16         $(objtree)/spl/board/samsung/$(BOARD)/tools/mk$(BOARD)spl,\
17         $(objtree)/tools/mkexynosspl) $(VAR_SIZE_PARAM) $< $@
18 endif
19 endif
```

这里就构成了u-boot-spl.bin到tiny210-spl.bin的过程了。

综上，spl的编译就完成了。

文章标签： u-boot spl 编译

个人分类： uboot

相关热词： uboot的功能 uboot下命令 uboot命令行 uboot剪裁 uboot组成

- 上一篇
- [uboot] （第一章）uboot流程——概述
- 下一篇
- [uboot] （第三章）uboot流程——uboot-spl代码流程

uboot 2016.05编译uboot.bin和spl

1： Makefile中的all目标编译出相应的文件...们来看看这个all目标 all: \$(ALL-y) 2： # Always append ALL so that...

2

写评论

 小楼豆腐 2016-12-23 16:08:55 # 目录

点个赞，之前Makefile看的很烦，这篇文章给我很大的启发，谢谢！！

收藏

微信

u-boot-2016.11移植uboot bin

一、时钟初始化 1、修改clock.h cd arch/arm/mach-s5pv210/include/mach/vim clock.h /\* add by Sourcelink \*/ struct...

 Config\_init 2016-11-28 08:10:46 阅读量：1335

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人阅读 ...

 chuhongcai 2016-06-28 22:31:35 阅读量：1449

...S3c2440在最新版本U-boot-2015.10移植(支持SPL模式...\_CSDN博客

boot-2015.10.tar.bz2 后解压,进入u-boot目录,...来控制MAKE编译产生uboot.bin或uboot-spl.bin.我... 选择选项 boot images --> 发现...

2018-6-14

uboot 2016.05编译uboot.bin和spl - CSDN博客

2:scripts/Makefile.spl,drivers/Makefile 3:编译过程: Build U-Boot on the...uboot2015第一阶段---SPL uboot-spl u010243305 2017-01-21 13:50:34...

2018-6-9

u-boot-2016.09移植(3)-u-boot-spl.bin

从本节开始，就正式进入移植过程，首先进行u-boot-spl.bin的移植。

 keyue123 2016-11-07 21:39:39 阅读量：1035

移植u-boot-2015.07-rc3之修改代码支持SDRAM和SPL启动(二) - doc...

make menuconfig 选择SPL make 编译出错: arch/arm/lib/built-in.o: In function `clr\_gd': /home/uboot/u-boot-2015.07-rc3/arch/arm/lib/crt0.S:10...

2017-6-23

u-boot SPL的理解 - CSDN博客

uboot 2016.05编译uboot.bin和spl 1: Makefile中的all目标编译出相应的文件...uboot2015第一阶段---SPL uboot-spl u010243305 2017-01-21 13:50:34...

2018-6-21

uboot-spl编译流程

[uboot] （第二章）uboot流程——uboot-spl编译流程2016年10月27日 20:40:47 阅读数：2675以下例子都以project X项目tiny210（s5pv210平台,ar...

 u013165704 2018-05-19 15:12:19 阅读量：10

亮仔移植u-boot系列之-- S3c2440在最新版本U-boot-2015.10移植(支持SPL模式启动) -- 1

首先在u-boot官网上下载最新的U-boot-2015.10版本，通过命令 sudo tar xvf u-boot-2015.10.tar.bz2 后解压，进入u-boot目录，执行: mak...

 yangxiao Zhu\_xiaoyu 2015-12-04 21:23:39 阅读量：2394

uboot2015第一阶段---SPL - CSDN博客

uboot-spl

2018-5-22

2

...S3c2440在最新版本U-boot中移植U-Boot 2015.10移植(支持SPL模式启动) -- 1 ...

u-boot-2015.10版本通过CONFIG\_SPL\_BUILD宏来控制MAKE编译产生uboot.bin或uboot.bin.gz。选择选项 boot images ----> 发现并没有Enable SPL的选项,退出menu后编辑uboot.config,添加CONFIG\_SPL\_BUILD=y,再编译。目录

2016-10-22

收藏

U-BOOT移植过程详解: SPL模式启动

U-BOOT移植过程详解: SPL模式启动

U-BOOT移植

2014-01-21 18:06

641人...

linuxarmsummary

2015-04-02 14:00

微信

0

阅读数 : 14460

微博

uboot的编译问题 - CSDN博客

2015年10月27日 18:49:37 阅读数:180

QQ

的编译开始是从别处拿来代码,...#include <config\_uncmd\_spl.h> 其中红色的头文件是在boards.cfg的options域里面...

2018-6-8

[uboot] (第三章)uboot流程——uboot-spl代码流程 - CSDN博客

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。...

2018-6-15

u-boot之SPL分析

SPL SPL是uboot第一阶段执行的代码.主要负责搬移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nand或flash启动。tuwenqi2013 2017-01-10 11:13:00 阅读数 : 1383

uboot移植（五）移植 u-boot-spl.bin

首先大致分析一下 u-boot 的代码走向， 通过看链接脚本 u-boot-2014.04/arch/arm/cpu/u-boot.lds 和u-boot-2014.04/arch/arm/cpu/u-boot-spl.lds。JerryGou 2018-05-18 21:15:03 阅读数 : 49

Atqiao - CSDN博客

当然,我们也可以选择使用SPL和不使用。在编译的过程中,是先编译第二阶段Uboot,...2015年5月 2篇 展开 热门文章 Uboot 2017.01 启动流程分析 阅读量:6310 ...

2018-7-9

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

新版UBOOT启动流程

转载请注明地址 : http://blog.csdn.net/zsy2020314/article/details/9824035 1.关于启动流程 1.1 启动阶段分为3个, bl0 , bl1 , ...

gujintong1110 2015-10-04 22:49:15 阅读数 : 2395

[uboot] （第三章）uboot流程——uboot-spl代码流程

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...

ooonebook 2016-10-28 16:24:14 阅读数 : 2505


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 EXTRAV...

qq\_25827755 2016-12-15 16:30:24 阅读数 : 294

### u-boot-2016.09 make编译过程分析（一）


上一篇文章《》详尽分析了u-boot执行命令make xxx\_defconfig的整个流程，本文着眼于编译流程，即配置完成后执行make命令生成二进制文件的过程。由于涉及的依赖文件太多，这里只分析编译流程。

 guyongqiangx 2016-09-17 21:10:00 阅读量：3618

[写评论](#)

### uboot流程——uboot-spl编译流程


来自：<https://www.cnblogs.com/leaven/entry/2018-03-28/46160.html> 公告昵称：海王园龄：8年3个月粉丝：241关注：0+加关注日历&lt;2018年3月&gt;g...

 wlf\_go 2018-03-28 16:42:44 阅读量：160

[收藏](#)

### ARM U-Boot SPL过程浅析

原文地址：<http://bbs.chinaunix.net/thread48378-1-1.html> 【1】SPL简介 SPL（Secondary programloader）是...

 jxgz\_leo 2016-08-02 22:45:10 阅读量：1433

[微博](#)

### uboot配置和编译过程详解

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

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

[QQ](#)

### uboot 2015-01版本启动linux简易流程

uboot 2015-01版

 litao31415 2016-10-27 00:39:05 阅读量：493

### [IMX6Q]uboot\_v2015.04编译流程分析

执行生成.config文件 #make mx6qecovacsandroid\_configMakefile: %config: scripts\_basic outputmakefile FORCE ...

 kris\_fei 2016-01-18 11:36:02 阅读量：3889

### [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 22:30:45 阅读量：2946

### 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SPL，再编译Uboot。

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

### 三、uboot的编译链接过程 (2011-03-10 20:36)

分类：uboot2010.09移植 配置完之后，执行make即可编译，从makefile中可以了解uboot使用了哪些文件、哪个文件先执行，可执行文件占用内存的情况。 下...

 mirkerson 2012-08-06 11:38:15 阅读量：3479

### uboot编译架构

2015年12月04日 31KB [下载](#)

### [uboot] （第四章）uboot流程——uboot编译流程

uboot编译流程简单介绍

 oonebook 2016-11-01 21:23:36 阅读量：3275

### uboot学习心得（uboot流程分析）max32590芯片

1、代码框架源码解压以后，我们可以看到以下的文件和文件夹：cpu与处理器相关的文件。每个子目录中都包括cpu.c和interrupt.c、start.S、u-boot.lds。cpu.c：初始化CP..

 guoyiyan1987

2018-05-08 10:56:22

阅读数：28

### 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

评论数：3725

### u-boot配置和编译过程详解

备注：分析的是OK210开发板自带的uboot，可能有些部分和其他版本不太一样，但是原理都类似。编译u-boot的步骤make forlinux\_linux\_config make首先...


 Kevin\_Mr

2016-05-16 21:43:21

收藏数：13836

### uboot - 启动内核过程分析

我们都知道u-boot被缔造出来的使命是启动Linux内核。那么，他是如何完成他的使命的涅！请看下面↓↓↓...（1）我们先来分析下Linux内核镜像这个概念吧。我们编译内核完（集..

 KayChanGEEK

2015-11-29 22:22:22

阅读数：2380

### 【u-boot】u-boot-2017.05启动过程分析（一）

u-boot发展至今，版本已经很多，随着版本的升级，框架越来越复杂，不过其启动流程的核心过程都是一样的，本博文以当前最新u-boot-2017.05为例分析其启动过程，主要I..


 qq\_38144425

2017-06-19 12:00:54

阅读数：2464

### 移植uboot-2015-10(一)

移植uboot-2015.10.rc1(一) 开发板： friendly arm 2440 工具： Win7 + VMware + ubuntu U-boot版本： u-...

 lee244868149

2015-09-30 14:00:33

阅读数：10406

### 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

### openwrt的uboot

http://wiki.openwrt.org/doc/techref/bootloader/uboot

 xiaoxiaozhu2010

2014-05-08 17:20:46

阅读数：4563

### #嵌入式Linux最小系统移植# 对uboot移植和裁剪的一点点个人思考和总结

思路: 1.分析启动流程 2.移植config文件(smdk440\_config) 3.移植包含控制条件编译宏的.h文件(configs/s3c2440.h) 4.移植板级初始化.c文件(s3c24...

 sinat\_26551021

2018-02-09 20:55:06

阅读数：174

### UBoot命令解析与执行流程

本文为CP根据网上内容以及U-Boot 2014.4版本进行整理而成。原文地址为http://blog.chinaunix.net/uid-8867796-id-358806.html —————...

 jiujiaobusinia

2016-11-25 16:09:27

阅读数：1539

### UBOOT 学习心得 ( UBOOT流程分析 )

网上找到的UBOOT研究文章，结合自己这几天看的。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..

 windsun0800

2013-12-14 01:25:34

阅读数：5661

### IMX6Solo启动流程-从Uboot到kernel 中

Uboot的C函数入口以及命令的处理流程

 baicaiaichibaicai

2015-08-28 10:26:43

阅读数：2102

### 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

2

### U-boot引导内核流程分析

当U-boot完成重定位和初始化外设后，便以裸机方式进入工作状态，可以加载内核镜像到DDR的链接地址中了...

qq\_28992301

    2016-07-10 20:30:00    

写评论

目录

    阅读数：3222

### 深入理解uboot 2016 - 基础篇（一）处理器启动流程分析（一）

最近一段时间一直在做uboot相关的移植工作，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。之前

kernel\_yx

    2016-11-05 14:52:35    

收藏

微信

    阅读数：8436

### 2017.1.7 \_u-boot的初步认识

今天结束了C语言部分和裸机部分，现在学习UBOOT，学习完毕UBOOT后以后就是在操作系统的基础上来进行操作裸机了。UBOOT是用来启动操作系统的。当我们学

u012204121

    2017-01-18 09:53:06    

QQ

阅读数：1070

### uboot移植-从uboot官方源码开始移植过程总结

1 选取源码 下载源码 解压源码 自行登录uboot官网，下载uboot源码，我选取的是2013.10的，因为之后的源码采用类似新的配置模式（能用即可）。我的源码是uboot-2013.

KayChanGEEK

    2016-01-16 16:08:40    

微信

阅读数：7072

### u-boot-2016.09 make编译过程分析（二）

上一篇文章详尽分析了u-boot执行配置命令make xxx\_defconfig的整个流程，本文着眼于编译流程，即配置完成后执行make命令生成二进制文件的过程。由于涉及的依赖和命

guyongqiangx

    2016-10-08 23:25:08    

微博

阅读数：2863

### 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    

微信

阅读数：955

### 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

dghfj

    2016-06-05 00:50:11    

微信

阅读数：640

### u-boot编译学习--uboot编译链接过程

参考博客：http://blog.chinaunix.net/uid-18921523-id-165078.html U - BOOT是一个LINUX下的工程，在编译之前必须已经安装...

zhaolushandong

    2016-01-16 11:12:16    

微信

阅读数：1986

### 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    

微信

阅读数：925

### 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    

微信

阅读数：827

### u-boot显示logo

经过个人实践一下两种方法都可以实现：首先lcd驱动必须能正常运行，想确定是不是正常，就看uboot启动时能不能显示默认的logo。 #define CONFIG\_LCD 其次想在u-boc

 sddsighhz

2015-03-12 18:37:33

阅读数：1959

### uboot启动流程分析之一

最开始的就是start.S 一个可执行的二进制文件，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 的 \_start...

 Stars\_Moon\_Sky

2015-04-22 14:00:00

写评论

阅读数：1312

### uboot的编译及配置浅析 和 uboot启动，内核启动详细讲解

uboot的编译及配置浅析 网址：http://blog.csdn.net/evenness/article/details/7395535 U-Boot的源码是通过Git管理的...

 a746742897

2016-11-12 16:30:00

阅读数：1340

2

目录

收藏

微信

微博

QQ

#### 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：

#### 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

#### 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

2

写评论

目录

收藏

微信

微博

QQ

展开

热门文章

[uboot] uboot启动kernel篇（二）——bootm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]woshidahuaidan2011[reply] CPU0运行，其他核心睡眠等待...

[project X] tiny2...

woshidahuaidan2011：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...

chenxiaozhen22：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。