u-boot 框架分析

内部文件：[1.0]

颁布时间：[2013.3.19]

目　　录

文件版本说明

表 1 版本说明

版本	发布时间	修订章节	作者
1.0	2013.03.19	初始版本	朱正晶

参考资料

1. U-boot source code
2. 网络资料：http://www.cnblogs.com/heaad/archive/2010/07/17/1779806.html
3. http://www.denx.de/wiki/U-Boot/

手册目的

分析 u-boot 整体框架结构，为之后 STM32 项目提供参考。

声明

无

名词定义和缩略语说明

表 2 名词定义及缩略语说明

序号	缩写	说明
1	U-boot	the Universal Boot Loader
2		

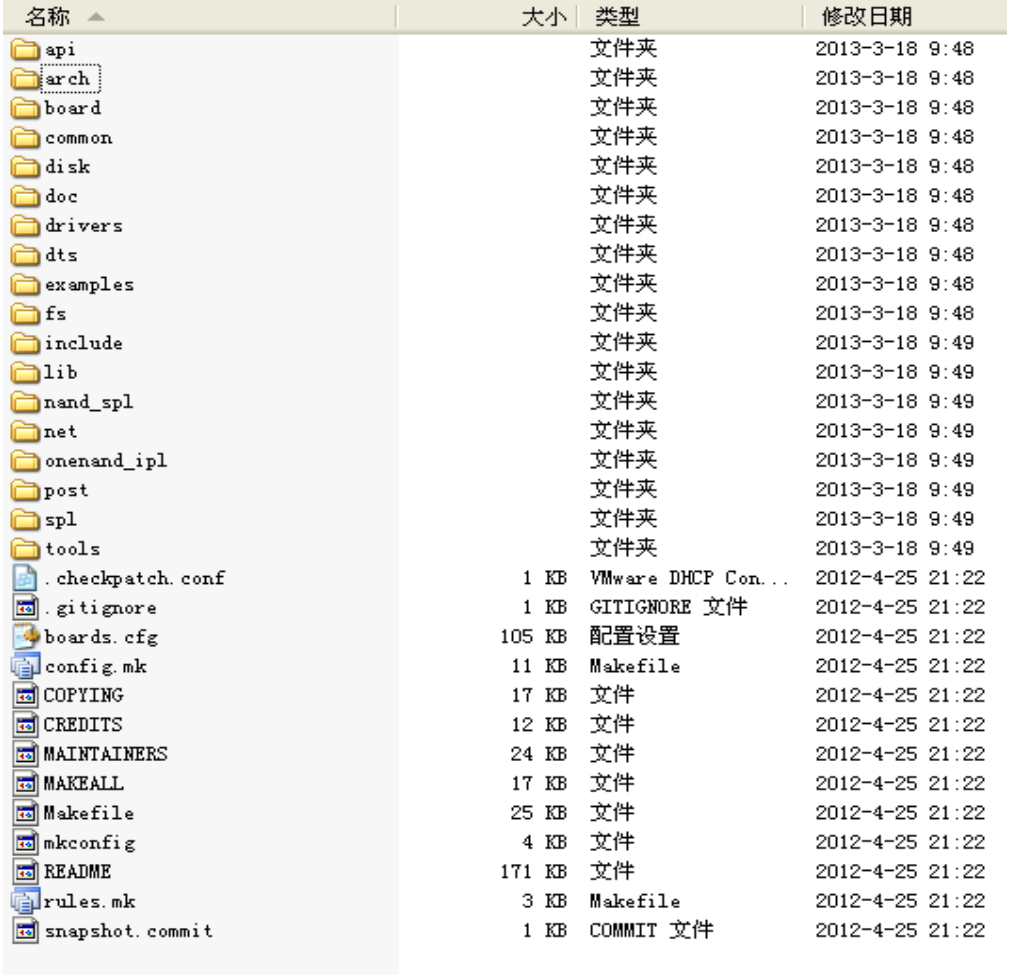
1 U-boot 分析

U-Boot，全称 Universal Boot Loader，是遵循 GPL 条款的开放源码项目。从 FADSROM、8xxROM、PPCBOOT 逐步发展演化而来。其源码目录、编译形式与 Linux 内核很相似，事实上，不少 U-Boot 源码就是相应的 Linux 内核源程序的简化，尤其是一些设备的驱动程序，这从 U-Boot 源码的注释中能体现这一点。

1.1 U-boot 目录结构

- board 目标板相关文件，主要包含 SDRAM、FLASH 驱动；
- common 独立于处理器体系结构的通用代码，如内存大小探测与故障检测；
- arch 包含各种 CPU 的架构代码，如 arm，avr 等等；
- driver 通用设备驱动，如 CFI FLASH 驱动（目前对 INTEL FLASH 支持较好）
- doc U-Boot 的说明文档；
- examples 可在 U-Boot 下运行的示例程序；如 hello_world.c,timer.c；
- include U-Boot 头文件，尤其 configs 子目录下与目标板相关的配置头文件是移植过程中经常要修改的文件；
- lib 一些常用的算法，如 CRC，MD5，zlib 等等；
- net 与网络功能相关的文件目录，如 bootp,nfs,tftp；
- post 上电自检文件目录。尚有待于进一步完善；
- rtc RTC 驱动程序；
- tools 用于创建 U-Boot S-RECORD 和 BIN 镜像文件的工具；

图 1-1 显示了 u-boot 2012.04.01 版本的目录结构



名称	大小	类型	修改日期
api		文件夹	2013-3-18 9:48
arch		文件夹	2013-3-18 9:48
board		文件夹	2013-3-18 9:48
common		文件夹	2013-3-18 9:48
disk		文件夹	2013-3-18 9:48
doc		文件夹	2013-3-18 9:48
drivers		文件夹	2013-3-18 9:48
dts		文件夹	2013-3-18 9:48
examples		文件夹	2013-3-18 9:48
fs		文件夹	2013-3-18 9:48
include		文件夹	2013-3-18 9:49
lib		文件夹	2013-3-18 9:49
nand_spl		文件夹	2013-3-18 9:49
net		文件夹	2013-3-18 9:49
onenand_ipl		文件夹	2013-3-18 9:49
post		文件夹	2013-3-18 9:49
spl		文件夹	2013-3-18 9:49
tools		文件夹	2013-3-18 9:49
.checkpatch.conf	1 KB	VMware DHCP Con...	2012-4-25 21:22
.gitignore	1 KB	GITIGNORE 文件	2012-4-25 21:22
boards.cfg	105 KB	配置设置	2012-4-25 21:22
config.mk	11 KB	Makefile	2012-4-25 21:22
COPYING	17 KB	文件	2012-4-25 21:22
CREDITS	12 KB	文件	2012-4-25 21:22
MAINTAINERS	24 KB	文件	2012-4-25 21:22
MAKEALL	17 KB	文件	2012-4-25 21:22
Makefile	25 KB	文件	2012-4-25 21:22
mkconfig	4 KB	文件	2012-4-25 21:22
README	171 KB	文件	2012-4-25 21:22
rules.mk	3 KB	Makefile	2012-4-25 21:22
snapshot.commit	1 KB	COMMIT 文件	2012-4-25 21:22

图 1-1 2012.04.01 U-Boot 目录结构

1.2 u-boot 编译流程

比如对于 mini2440 开发板，编译 U-Boot 需要执行如下的命令：

```
$ make mini2440_config
$ make all
```

分析：第一条命令 make mini2440_config 实际上执行了如下命令：

```
./mkconfig mini2440 arm arm920t mini2440 samsung s3c24x0
```

在根目录下有一个 mkconfig 文件，里面有这个脚本的使用方法：

```
#!/bin/sh -e

# Script to create header files and links to configure
# U-Boot for a specific board.
#
# Parameters:  Target  Architecture  CPU  Board [VENDOR] [SOC]
#
```

(C) 2002-2010 DENX Software Engineering, Wolfgang Denk <wd@denx.de>
#

上面的命令就是将“mini2440 arm arm920t mini2440 samsung s3c24x0”作为参数传递给当前目录下的 mkconfig 脚本执行。

make mini2440_config 执行的结果：

（1） 创建到目标板相关的文件的链接

```
ln -s asm-arm asm
ln -s arch-s3c24x0 asm-arm/arch
ln -s proc-armv asm-arm/proc
```

（2） 创建 include/config.mk 文件，内容如下所示：

```
ARCH   = arm
CPU    = arm920t
BOARD  = mini2440
VENDOR = samsung
SOC    = s3c24x0
```

（3） 创建与目标板相关的文件 include/config.h，如下所示：

```
/* Automatically generated - do not edit */
#define CONFIG_BOARDDIR board/samsung/mini2440
#include <config_defaults.h>
#include <configs/mini2440.h>
#include <asm/config.h>
```

make all 的就是使用上面的文件执行编译，生成需要的 u-boot 文件。

2 总结及展望

通过上面简单分析可以看出，u-boot 是通过将不同功能，不用 CPU 分类等放入相应的文件夹，在顶层使用配置文件来配置需要的 u-boot。
我们的 STM32 项目可以参照 u-boot 实现方法来组织软件架构，在项目起步时把框架搭好，为以后的硬件、软件升级做好打算。

2.1 [选择 RTOS]

STM32 有足够的资源运行 RTOS。加入成熟的 RTOS 可以大大简化软件的开发难度和提升软件的可读性及可移植性，特别是可以轻松应对大规模的应用程序。不同的任务放在不同的 task 中，不同的 task 之间通过定义明确的命令进行交流。程序出了问题也很容易定位，这样大大提供了开发效率。

(This is the last page)