



IAC-A5D3x-Kit Linux
用户手册

版本号 v2.0

2014/10/08

杭州启扬智能有限公司版权所有
QIYANG TECHNOLOGY Co., Ltd
Copyright Reserv

目 录

前言.....	4
一、阅读前说明（请仔细阅读）.....	5
二、烧写 Linux 系统镜像.....	6
三、功能说明与测试.....	6
四、安装交叉编译工具链.....	6
五、编译 bootstrap.....	9
六、编译 uboot.....	13
七、内核的编译.....	15
八、文件系统制作.....	21
九、应用程序的开发.....	22
十、结束语.....	23

版本更新记录

版本	硬件平台	描述	日期	修订人
1.0	IAC-A5D3x-Kit	初始版本，首次发布	2014-02-08	yao
2.0	IAC-A5D3x-Kit	修改 uboot、内核、文件系统源码版本号	2014-10-08	wwx

前言

欢迎使用杭州启扬智能科技有限公司产品 IAC-A5D3x-Kit，本产品 Linux 部分包含 4 份手册：

- 《IAC-A5D3x-Kit Linux 用户手册.pdf》
- 《IAC-A5D3x-Kit 硬件说明手册.pdf》
- 《IAC-A5D3x-Kit Linux 模块说明与测试手册.pdf》
- 《IAC-A5D3x-Kit Linux 系统镜像烧写手册.pdf》

硬件相关部分可以参考《IAC-A5D3x-Kit 硬件说明手册.pdf》。
主板测试可以参考《IAC-A5D3x-Kit Linux 模块说明与测试手册.pdf》。
镜像烧写参考《IAC-A5D3x-Kit Linux 系统镜像烧写手册.pdf》。

本手册主要介绍交叉编译环境的搭建、源代码以及应用例程的编译。

使用本手册之前请仔细阅读《IAC-A5D3x-Kit 硬件说明手册.pdf》。

公司简介

杭州启扬智能科技有限公司位于美丽的西子湖畔,是一家集研发、生产、销售为一体的高新技术产业。公司致力于成为嵌入式解决方案的专业提供商,为嵌入式应用领域客户提供软硬件开发工具和嵌入式系统完整解决方案。产品范围主要包括: Cirrus Logic EP93xx 系列 ARM9 主板、ATMEL AT91SAM926x 系列主板, FreeScale iMX 系列主板, TI Davinci 系列音/视频通用开发平台等等。可运行 Linux2.4/2.6、WinCE5.0/6.0 操作系统,并可根据客户需求开发各种功能组合的嵌入式硬件系统。应用领域涉及: 工业控制、数据采集、信息通讯、医疗设备、视频监控、车载娱乐等等。

客户的需求是公司发展的动力,公司将不断完善自身,与客户互助互惠,共同发展。

电话: 0571-87858811, 87858822

传真: 0571-87858822

技术支持 E-MAIL: support@qiyangtech.com

网址: <http://www.qiyangtech.com>

地址: 杭州市西湖区西湖科技园西园一路 8 号 3A 幢 5 层

邮编: 310012

有任何技术问题或需要帮助, 请联系: supports@qiyangtech.com

第 4 页 共 24 页

购买产品, 请联系销售: sales@qiyangtech.com

更多信息请访问: <http://www.qiytech.com>

©2012 Qiyangtech 版权所有

一、阅读前说明（请仔细阅读）

◆ 装有 Linux 系统（ubuntu 或其它 Linux 发行版），本手册以 ubuntu 12.04 操作为例，具体搭建请参照《虚拟机安装 ubuntu 指导手册.pdf》！

◆ 由于编译过程中需要请文件拷贝到虚拟机 ubuntu，在 ubuntu 用户目录下创建一个工作目录：`mkdir ~/work` /* 这里的~/表示用户目录，实际对应的绝对路径为/home/st */

为了统一和陈述方便，所有文件都拷贝到该目录进行操作，具体目录用户可以自行创建，这里只是以~/work 目录为例！

◆ 关于 linux 下的常用命令以及 vi 的操作等在这里都不做详细说明，请用户自行查阅相关资料！

◆ 所有 PC 机和虚拟机的拷贝都采用 samba 共享访问的方式来拷贝！

◆ 串口连接：通过提供的串口线将开发板的调试串口(J7)与 PC 机的串口连接

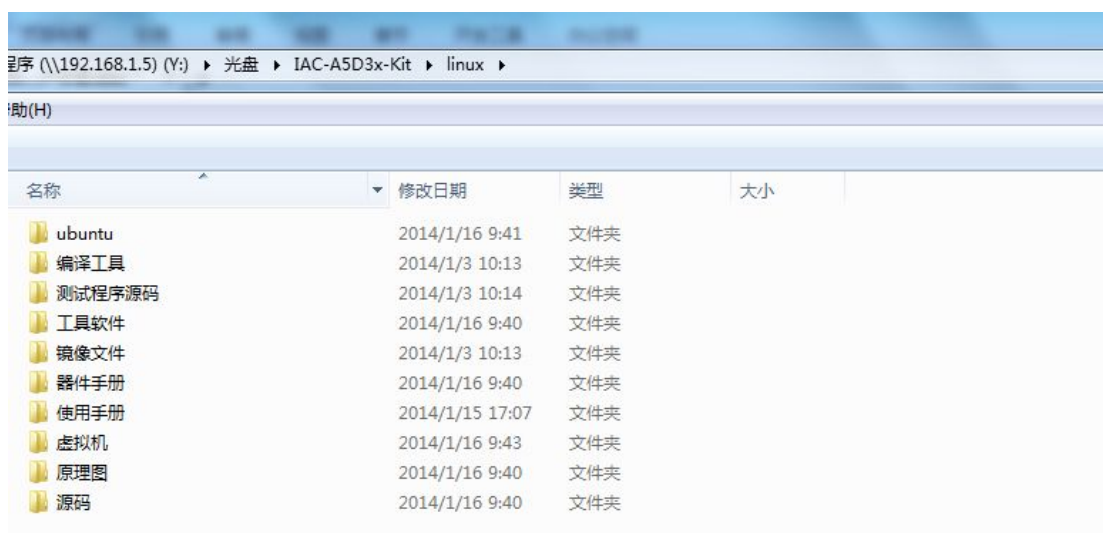
◆ 网络连接：通过网线将开发板的以太网接口(J16)与 PC 机的网络接口连接

◆ USB 连接：通过 USB 连接线将开发板的 USB Device(J22)与 PC 机的 USB 连接

◆ 串口设置：打开终端通讯软件（迷你终端或 Windows 下的超级终端），选择所用到的串口并设置如下参数：波特率（115200）、数据位（8 位）、停止位（1 位）、校验位（无）、数据流控制（无）

◆ 找到核心板上的 J1 跳线，在下面的操作步骤中要用到它

◆ 开发板标配光盘目录，文档说明中用到的所有工具软件以及代码文件全部在光盘的对应目录下，使用前请确保光盘资料齐全！



名称	修改日期	类型	大小
ubuntu	2014/1/16 9:41	文件夹	
编译工具	2014/1/3 10:13	文件夹	
测试程序源码	2014/1/3 10:14	文件夹	
工具软件	2014/1/16 9:40	文件夹	
镜像文件	2014/1/3 10:13	文件夹	
器件手册	2014/1/16 9:40	文件夹	
使用手册	2014/1/15 17:07	文件夹	
虚拟机	2014/1/16 9:43	文件夹	
原理图	2014/1/16 9:40	文件夹	
源码	2014/1/16 9:40	文件夹	

有任何技术问题或需要帮助，请联系：supports@qiyangtech.com

第 5 页 共 24 页

购买产品，请联系销售：sales@qiyangtech.com

更多信息请访问：<http://www.qiytech.com>

©2012 Qiyangtech 版权所有

二、烧写 Linux 系统镜像

如果需要重新烧写 linux 系统，开发板提供了 dataflash 和 nandflash 两种启动方式，请根据需要进行烧写，具体烧写方法请参照：

《IAC-A5D3x-Kit Linux 系统镜像烧写手册.pdf》

该手册详细介绍了 dataflash 和 nandflash 的烧写，以及通过网络更新镜像的方法。

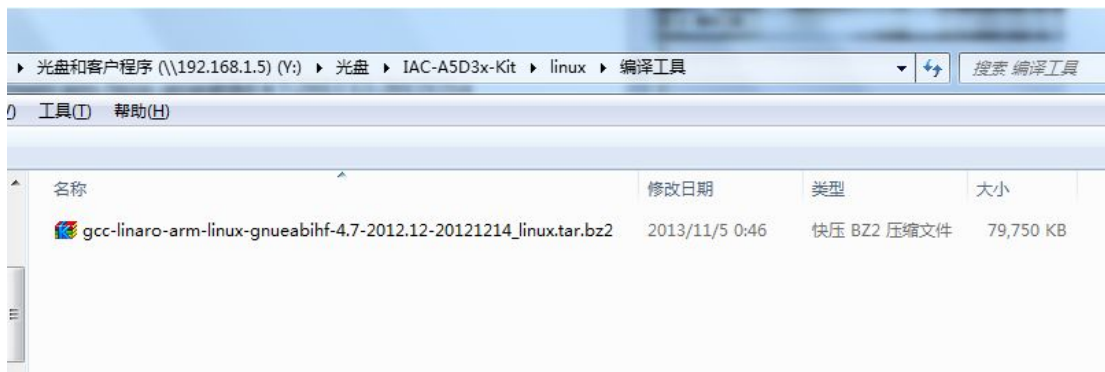
三、功能说明与测试

开发板标配文件系统中已经集成了测试程序，启动板子后可在/usr/local/board_test 目录下找到相应的测试程序，具体测试方法，请参照手册：

《IAC-A5D3x-Kit Linux 模块说明与测试手册.pdf》

四、安装交叉编译工具链

bootloader、kernel、fs 的重新编译都需要用到交叉编译器，所有的应用程序以及库文件需要在开发板上运行的话也需要交叉编译器进行编译。所以，在这里需要先安装一下交叉编译工具链，在光盘的编译工具的目录下已经有制作好的交叉编译工具，用户可以直接安装使用，该交叉编译器的 GCC 版本为 4.7.3。



下面就来介绍一下如何安装交叉编译工具链：

将gcc-linaro-arm-linux-gnueabi-4.7-2012.12-20121214_linux.tar.bz2 这个交叉编译工具链拷贝到~/work目录下。

```
st@st-virtual-machine:~/work$ ls
gcc-linaro-arm-linux-gnueabi-4.7-2012.12-20121214_linux.tar.bz2
st@st-virtual-machine:~/work$
```

用如下命令进行解压:

```
$ tar -xjvf gcc-linaro-arm-linux-gnueabi-4.7-2012.12-20121214_linux.tar.bz2
```

在当前目录下生成 gcc-linaro-arm-linux-gnueabi-4.7-2012.12-20121214_linux 文件夹。

```
gnueabi/4.7.3/liblto_plugin.so.0.0.0
gcc-linaro-arm-linux-gnueabi-4.7-2012.12-20121214_linux/libexec/gcc/arm-linux-gnueabi/4.7.3/collect2
gcc-linaro-arm-linux-gnueabi-4.7-2012.12-20121214_linux/libexec/gcc/arm-linux-gnueabi/4.7.3/liblto_plugin.so.0
gcc-linaro-arm-linux-gnueabi-4.7-2012.12-20121214_linux/libexec/gcc/arm-linux-gnueabi/4.7.3/liblto_plugin.so
st@st-virtual-machine:~/work$ ls
gcc-linaro-arm-linux-gnueabi-4.7-2012.12-20121214_linux
gcc-linaro-arm-linux-gnueabi-4.7-2012.12-20121214_linux.tar.bz2
st@st-virtual-machine:~/work$
```

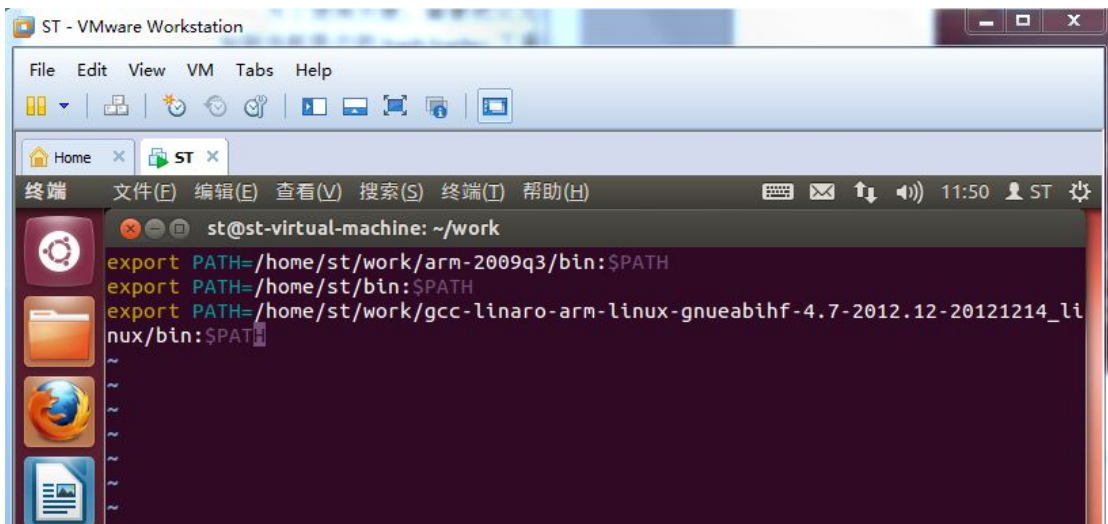
为了使用方便, 需要把交叉编译器的路径添加到系统环境变量 PATH 里面, 在这里添加到当前用户的 bash.bashrc 下面就可以了。

```
$ vi ~/.bashrc
```

在文件后面添加以下路径:

```
export PATH=
/home/st/work/gcc-linaro-arm-linux-gnueabi-4.7-2012.12-20121214_linux/bin:$PATH
```

注意: 上面环境变量无换行



保存，退出！

使新的环境变量生效

\$ `source ~/.bashrc`

环境变量生效之后，下面我们来确认一下交叉编译器是否安装成功：

```

st@st-virtual-machine: ~/work
st@st-virtual-machine:~/work$ arm-linux-gnueabi-gcc -v
使用内建 specs。
COLLECT_GCC=arm-linux-gnueabi-gcc
COLLECT_LTO_WRAPPER=/home/st/work/gcc-linaro-arm-linux-gnueabi-4.7-2012.12-20121214_linux/bin/./libexec/gcc/arm-linux-gnueabi/4.7.3/lto-wrapper
目标: arm-linux-gnueabi
配置为: /cbuild/slaves/oorts/crostoool-ng/builds/arm-linux-gnueabi-linux/.build/src/gcc-linaro-4.7-2012.12/configure --build=i686-build_pc-linux-gnu --host=i686-build_pc-linux-gnu --target=arm-linux-gnueabi --prefix=/cbuild/slaves/oorts/crostoool-ng/builds/arm-linux-gnueabi-linux/install --with-sysroot=/cbuild/slaves/oorts/crostoool-ng/builds/arm-linux-gnueabi-linux/install/arm-linux-gnueabi/libc --enable-languages=c,c++,fortran --enable-multilib --with-arch=armv7-a --with-tune=cortex-a9 --with-fpu=vfpv3-d16 --with-float=hard --with-pkgversion='crostoool-NG linaro-1.13.1-4.7-2012.12-20121214 - Linaro GCC 2012.12' --with-bugurl=https://bugs.launchpad.net/gcc-linaro --enable-__cxa_atexit --enable-libmudflap --enable-libgomp --enable-libssp --with-gmp=/cbuild/slaves/oorts/crostoool-ng/builds/arm-linux-gnueabi-linux/.build/arm-linux-gnueabi/build/static --with-mpfr=/cbuild/slaves/oorts/crostoool-ng/builds/arm-linux-gnueabi-linux/.build/arm-linux-gnueabi/build/static --with-mpc=/cbuild/slaves/oorts/crostoool-ng/builds/arm-linux-gnueabi-linux/.build/arm-linux-gnueabi/build/static --with-ppl=/cbuild/slaves/oorts/crostoool-ng/builds/arm-linux-gnueabi-linux/.build/arm-linux-gnueabi/build/static --with-cloog=/cbuild/slaves/oorts/crostoool-ng/builds/arm-linux-gnueabi-linux/.build/arm-linux-gnueabi/build/static --with-libelf=/cbuild/slaves/oorts/crostoool-ng/builds/arm-linux-gnueabi-linux/.build/arm-linux-gnueabi/build/static --with-host-libstdcxx='-L/cbuild/slaves/oorts/crostoool-ng/builds/arm-linux-gnueabi-linux/.build/arm-linux-gnueabi/build/static/lib -lpwl' --enable-threads=posix --disable-libstdcxx-pch --enable-linker-build-id --enable-gold --with-local-prefix=/cbuild/slaves/oorts/crostoool-ng/builds/arm-linux-gnueabi-linux/install/arm-linux-gnueabi/libc --enable-c99 --enable-long-long --with-mode=thumb
线程模型: posix
gcc 版本 4.7.3 20121205 (prerelease) (crostoool-NG linaro-1.13.1-4.7-2012.12-20121214 - Linaro GCC 2012.12)
st@st-virtual-machine:~/work$
    
```

从这里可以看出该交叉编译器的gcc版本为4.7.3。

交叉编译器安装好之后，就可以用来编译系统源码和应用程序了。

五、编译 bootstrap

bootstrap 为 linux 的二级引导程序，一级引导程序已经固化在 CPU 芯片的内部 ROOM 中，用户无法修改！bootstrap 主要负责基本的时钟、GPIO、内存等的初始化工作。IAC-A5D3x-Kit 开发板根据存放引导镜像介质的不同，提供了一下 3 种启动引导方式：

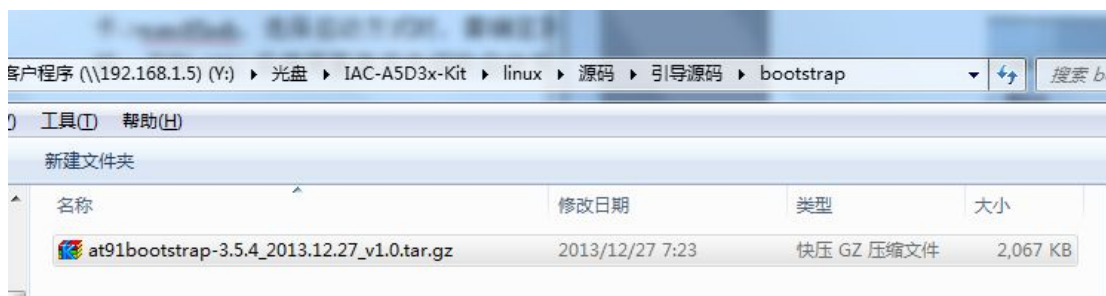
- ◆ dataflash 启动
- ◆ nandflash 启动
- ◆ SD 卡启动

CPU 在启动过程中根据内部程序，以以下优先级查找存储介质中的启动代码：

Dataflash > nandflash > SD 卡

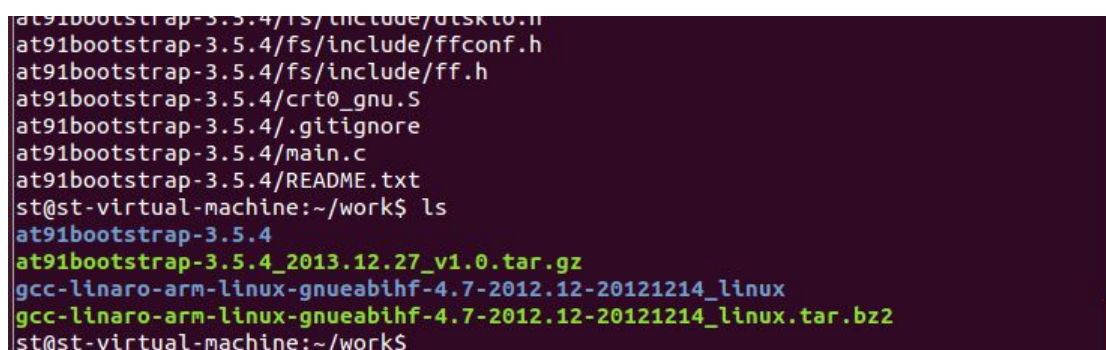
所以在选择启动方式启动的之前，要先确定其他更高等级的引导介质中不存在启动代码或者介质已经断开连接，否则 CPU 将使用更高优先级的启动方式来启动。

光盘中已经提供了编译好的 bootstarp 镜像文件，可直接使用。如果需要重新编译，光盘中也已经提供了移植好的 bootstrap 源码：



将光盘中的 bootstrap 源码拷贝到~/work 目录下，并解压

```
$ tar -xzf at91bootstrap-3.5.4_2013.12.27_v1.0.tar.gz
```



解压之后将得到 at91bootstrap-3.5.4 文件夹，进入该文件夹

有任何技术问题或需要帮助，请联系：supports@qiyangtech.com

第 9 页 共 24 页

购买产品，请联系销售：sales@qiyangtech.com

更多信息请访问：<http://www.qiytech.com>

©2012 Qiyangtech 版权所有

\$ cd at91bootstrap-3.5.4

```
st@st-virtual-machine:~/work$ ls
at91bootstrap-3.5.4
at91bootstrap-3.5.4_2013.12.27_v1.0.tar.gz
gcc-linaro-arm-linux-gnueabi-4.7-2012.12-20121214_linux
gcc-linaro-arm-linux-gnueabi-4.7-2012.12-20121214_linux.tar.bz2
st@st-virtual-machine:~/work$ cd at91bootstrap-3.5.4/
st@st-virtual-machine:~/work/at91bootstrap-3.5.4$ ls
board          config          elf32-littlearm.lds  KNOWN_ISSUES  README.txt
build_df.sh    Config.in       fs                   lib            scripts
build_nf.sh    crt0_gnu.S     host-utilities      main.c        toplevel_cpp.mk
build_sd.sh    driver         include              Makefile
st@st-virtual-machine:~/work/at91bootstrap-3.5.4$
```

如上图所示，为了方便用户的编译，我们已经根据启动方式的不同，把对应启动方式的编译步骤制作成了相应的脚本文件，如下：

- ◆ build_df.sh 对应 dataflash 启动 bootstrap 编译脚本
- ◆ build_nf.sh 对应 nataflash 启动 bootstrap 编译脚本
- ◆ build_sd.sh 对应 SD 卡启动 bootstrap 编译脚本

用户可根据需要选择执行对应的脚本程序进行编译，这里以 dataflash 为例进行简单介绍，其他启动方式编译方法一样就不再重复说明。执行 build_df.sh 脚本

\$./build_df.sh

执行如上脚本后，将显示以下界面

```
st@st-virtual-machine: ~/work/at91bootstrap-3.5.4
.config - at91bootstrap vBR2_VERSION Configuration

at91bootstrap Configuration
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> selects a feature,
while <N> will exclude a feature. Press <Esc><Esc> to exit, <?> for
Help, </> for Search. Legend: [*] feature is selected [ ] feature is

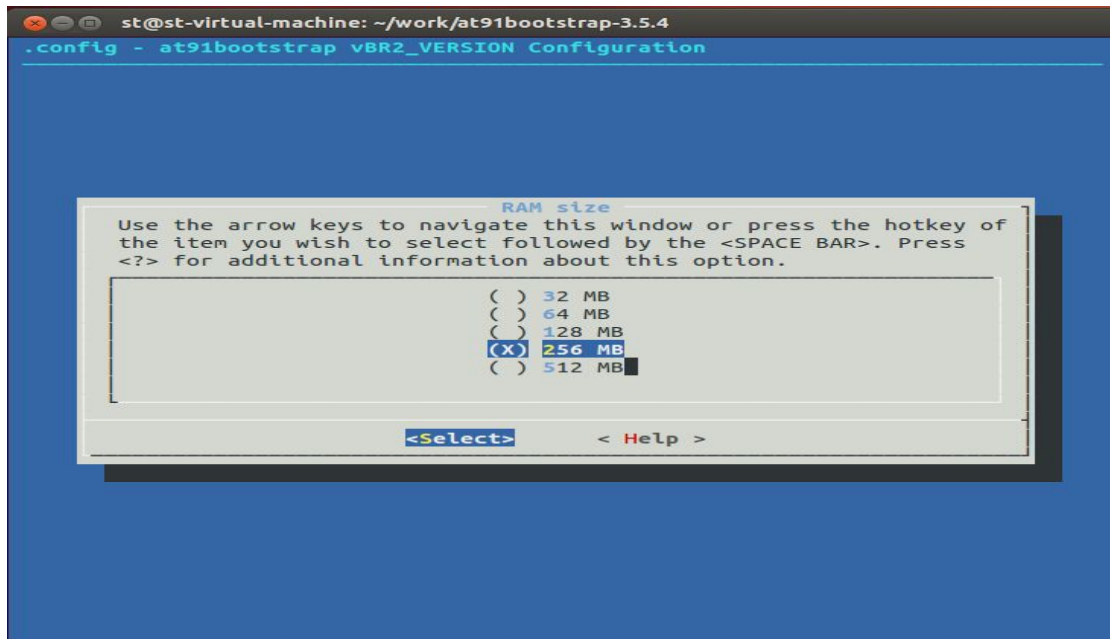
(at91sama5d3xek) Board Name
Board Type (at91sama5d3xek) --->
Crystal Frequency (Build for use of an 12.000 MHz crystal) --
CPU clock (528 MHz) --->
Bus Speed (133 MHz) --->
Memory selection --->
Image Loading Strategy (Load U-Boot into last MBYTE of SDRAM)
U-Boot Image Storage Setup --->
[ ] Perform a memory test at startup
[*] Debug Support
    Debug Level (General debug information) --->
[*] Call Hardware Initialization
[ ] Call User specific Hardware Initialization
[*] Use external 32KHZ oscillator as source of slow clock
[*] Disable Watchdog
---
Load an Alternate Configuration File
Save an Alternate Configuration File

<Select> < Exit > < Help >
```

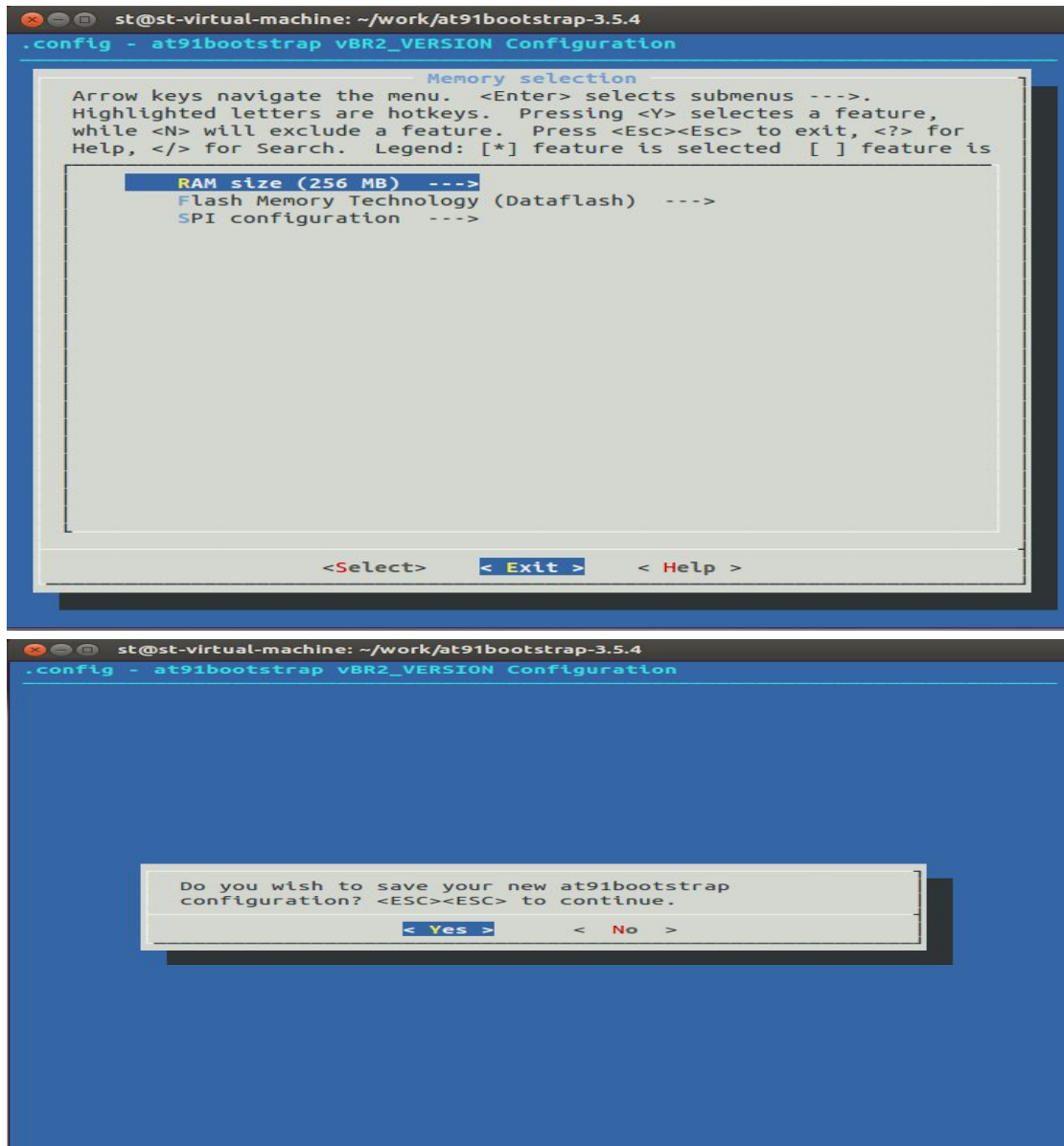
用户可根据需要进行配置，配置完毕后选择 YES 保存退出，若不需要修改，则默认退出即可。对于 IAC-A5D3x-Kit 默认配置，则需将 Memory selection 选择内存为 256M。

Memory selection --->

RAM size (512 MB) --->



选择好之后，退出



选择< Yes > 进行保存

配置完毕后，直接执行 `make` 命令进行编译。

`$ make`

编译完成并无错显示


```

st@st-virtual-machine: ~/work/at91bootstrap-3.5.4
G_CPU_CLK_528MHZ -DCONFIG_BUS_SPEED_133MHZ -DCONFIG_HAS_PIO3 -DCONFIG_LOAD_ONE_WI
RE -DCONFIG_MMC_SUPPORT -DCONFIG_DDR2 -DCONFIG_RAM_256MB -DCONFIG_DATAFLASH -DCON
FIG_SMALL_DATAFLASH -DAT91C_SPI_CLK=33000000 -DAT91C_SPI_PCS_DATAFLASH=AT91C_SPI_
PCS0_DATAFLASH -DBOOTSTRAP_DEBUG_LEVEL=DEBUG_INFO -DCONFIG_DISABLE_WATCHDOG

ld FLAGS
=====
-nostartfiles -Map=/home/st/work/at91bootstrap-3.5.4/binaries/at91sama5d3kek-data
flashboot-uboot-3.5.4.map --cref -static -T elf32-littlearm.lds --gc-sections -Tt
ext 0x300000

AS      /home/st/work/at91bootstrap-3.5.4/crt0_gnu.S
CC      /home/st/work/at91bootstrap-3.5.4/main.c
CC      /home/st/work/at91bootstrap-3.5.4/board/at91sama5d3kek/at91sama5d3kek
.c
CC      /home/st/work/at91bootstrap-3.5.4/lib/string.c
CC      /home/st/work/at91bootstrap-3.5.4/lib/eabi_utils.c
CC      /home/st/work/at91bootstrap-3.5.4/lib/div.c
CC      /home/st/work/at91bootstrap-3.5.4/driver/debug.c
CC      /home/st/work/at91bootstrap-3.5.4/driver/at91_slowclk.c
CC      /home/st/work/at91bootstrap-3.5.4/driver/at91_pio.c
CC      /home/st/work/at91bootstrap-3.5.4/driver/pmc.c
CC      /home/st/work/at91bootstrap-3.5.4/driver/at91_pit.c
CC      /home/st/work/at91bootstrap-3.5.4/driver/at91_wdt.c
CC      /home/st/work/at91bootstrap-3.5.4/driver/dbgu.c
CC      /home/st/work/at91bootstrap-3.5.4/driver/ddramc.c
CC      /home/st/work/at91bootstrap-3.5.4/driver/at91_spi.c
CC      /home/st/work/at91bootstrap-3.5.4/driver/dataflash.c
CC      /home/st/work/at91bootstrap-3.5.4/driver/ds24xx.c
mkdir -p /home/st/work/at91bootstrap-3.5.4/binaries
LD      at91sama5d3kek-dataflashboot-uboot-3.5.4.elf
Size of at91sama5d3kek-dataflashboot-uboot-3.5.4.bin is 4280 bytes
[Successful] It's OK to fit into SRAM area
st@st-virtual-machine:~/work/at91bootstrap-3.5.4$
    
```

可在 binaries 目录下得到可执行文件 at91sama5d3kek-dataflashboot-uboot-3.5.4.bin

```

[Successful] It's OK to fit into SRAM area
st@st-virtual-machine:~/work/at91bootstrap-3.5.4$ ls
binaries  config      elf32-littlearm.lds  lib          scripts
board    Config.in   fs                   main.c       topLevel_cpp.mk
build_df.sh crt0_gnu.o  host-utilities      main.o
build_nf.sh crt0_gnu.S  include              Makefile
build_sd.sh driver      KNOWN_ISSUES        README.txt
st@st-virtual-machine:~/work/at91bootstrap-3.5.4$ ls binaries
at91sama5d3kek-dataflashboot-uboot-3.5.4.bin
at91sama5d3kek-dataflashboot-uboot-3.5.4.elf
at91sama5d3kek-dataflashboot-uboot-3.5.4.map
st@st-virtual-machine:~/work/at91bootstrap-3.5.4$
    
```

六、编译 uboot

光盘中有移植好的 uboot 源码，用户可直接编译使用。

有任何技术问题或需要帮助，请联系：supports@qiyangtech.com

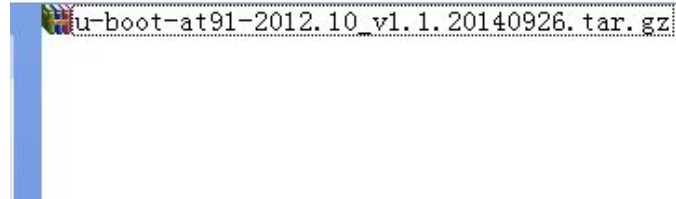
第 13 页 共 24 页

购买产品，请联系销售：sales@qiyangtech.com

更多信息请访问：<http://www.qiytech.com>

©2012 Qiyangtech 版权所有

3X-Kit\源码\引导源码\u-boot



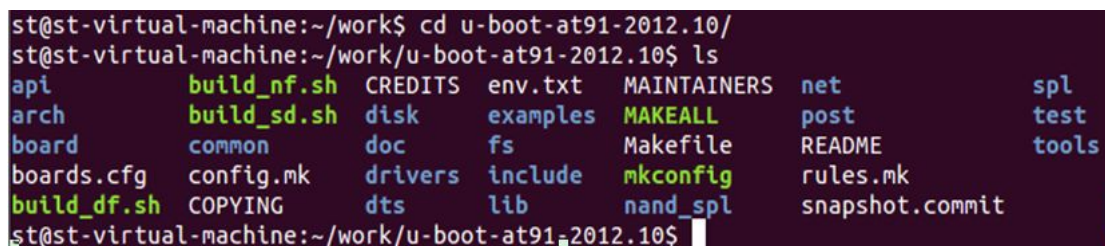
将光盘中的 u-boot 源码拷贝到~/work 工作目录中，并用以下命令解压：

```
$ tar -xzf u-boot-at91-2012.10_v1.1.20140926.tar.gz
```

解压之后得到 u-boot-at91-2012.10 文件夹，进入该文件夹

```
$ cd u-boot-at91-2012.10
```

```
$ ls
```



如上图所示，为了方便用户的编译，我们已经根据启动方式的不同，把对应启动方式的编译步骤制作成了相应的脚本文件，如下：

- ◆ build_df.sh 对应 dataflash 启动 bootstrap 编译脚本
- ◆ build_nf.sh 对应 nataflash 启动 bootstrap 编译脚本
- ◆ build_sd.sh 对应 SD 卡启动 bootstrap 编译脚本

用户可根据需要选择执行对应的脚本程序进行编译，这里以 dataflash 为例进行简单介绍，其他启动方式编译方法一样。

```
$ ./build_df.sh
```

编译无错误，在当前目录下得到可烧写到开发板的镜像文件 uboot.bin

```
$ ls
```

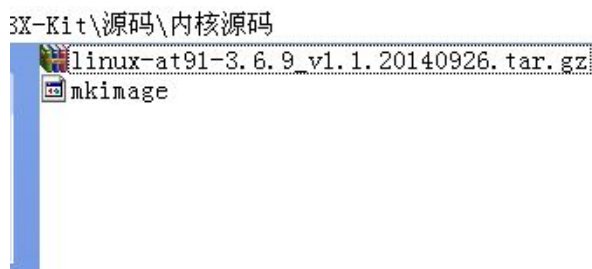
```

st@st-virtual-machine: ~/work/u-boot-at91-2012.10
d.o drivers/mtd/nand/libnand.o drivers/mtd/onenand/libonenand.o drivers/mtd/spi/l
ibspi_flash.o drivers/mtd/ubi/libubi.o drivers/net/libnet.o drivers/net/phy/libph
y.o drivers/pci/libpci.o drivers/pcmcia/libpcmcia.o drivers/power/libpower.o driv
ers/rtc/librtc.o drivers/serial/libserial.o drivers/spi/libspi.o drivers/twserial
/libtws.o drivers/usb/eth/libusb_eth.o drivers/usb/gadget/libusb_gadget.o drivers
/usb/host/libusb_host.o drivers/usb/musb/libusb_musb.o drivers/usb/phy/libusb_phy
.o drivers/usb/ulpi/libusb_ulpi.o drivers/video/libvideo.o drivers/watchdog/libwa
tchdog.o fs/cramfs/libcramfs.o fs/ext4/libext4fs.o fs/fat/libfat.o fs/fdos/libfdo
s.o fs/jffs2/libjffs2.o fs/reiserfs/libreiserfs.o fs/ubifs/libubifs.o fs/yaffs2/l
ibyaffs2.o fs/zfs/libzfs.o lib/libfdt/libfdt.o lib/libgeneric.o lib/lzma/liblzma.
o lib/lzo/liblzo.o lib/zlib/libz.o net/libnet.o post/libpost.o test/libtest.o boa
rd/atmel/sama5d3xek/libsama5d3xek.o --end-group /home/st/work/u-boot-at91-2012.10
/arch/arm/lib/eabi_compat.o -L /home/st/work/gcc-linaro-arm-linux-gnueabihf-4.7-
2012.12-20121214_linux/bin/./lib/gcc/arm-linux-gnueabihf/4.7.3 -lgcc -Map u-boot
.map -o u-boot
arm-linux-gnueabihf-objcopy -O srec u-boot u-boot.srec
arm-linux-gnueabihf-objcopy --gap-fill=0xff -O binary u-boot u-boot.bin
make -C examples/standalone all
make[1]: 正在进入目录 `/home/st/work/u-boot-at91-2012.10/examples/standalone'
make[1]: 没有什么可以做的为 `all'。
make[1]:正在离开目录 `/home/st/work/u-boot-at91-2012.10/examples/standalone'
make -C examples/api all
make[1]: 正在进入目录 `/home/st/work/u-boot-at91-2012.10/examples/api'
make[1]: 没有什么可以做的为 `all'。
make[1]:正在离开目录 `/home/st/work/u-boot-at91-2012.10/examples/api'
st@st-virtual-machine:~/work/u-boot-at91-2012.10$ ls
api                common            dts                MAKEALL            rules.mk           u-boot.bin
arch               config.mk         env.txt            Makefile           snapshot.commit   u-boot.lds
board              COPYING           examples           mkconfig           spl                u-boot.map
boards.cfg         CREDITS          fs                 nand_spl           System.map         u-boot.srec
build_df.sh        disk              include            net                 test
build_nf.sh        doc               lib                post                tools
build_sd.sh        drivers           MAINTAINERS       README             u-boot
st@st-virtual-machine:~/work/u-boot-at91-2012.10$
    
```

☆注意: 烧写的时候请确保 uboot 和 bootstrap 镜像是采用同一种启动方式编译得到的。

七、内核的编译

光盘中有移植配置好的内核源码文件



说明: linux-at91-3.6.9_v1.1.20140926.tar.gz 为内核源代码

mkimage 为生成 uImage 所需要的工具

将光盘中的 linux-at91-3.6.9_v1.1.20140926.tar.gz 和 mkimage 拷贝到~/work 工作目录

有任何技术问题或需要帮助, 请联系: supports@qiyangtech.com

第 15 页 共 24 页

购买产品, 请联系销售: sales@qiyangtech.com

更多信息请访问: <http://www.qiytech.com>

©2012 Qiyangtech 版权所有

中，解压内核源码

```
$ tar -xzvf linux-at91-3.6.9_v1.1.20140926.tar.gz
```

解压之后，在当前目录下生成 linux-at91-3.6.9 文件夹，进入该文件夹

```
$ cd linux-at91-3.6.9
```

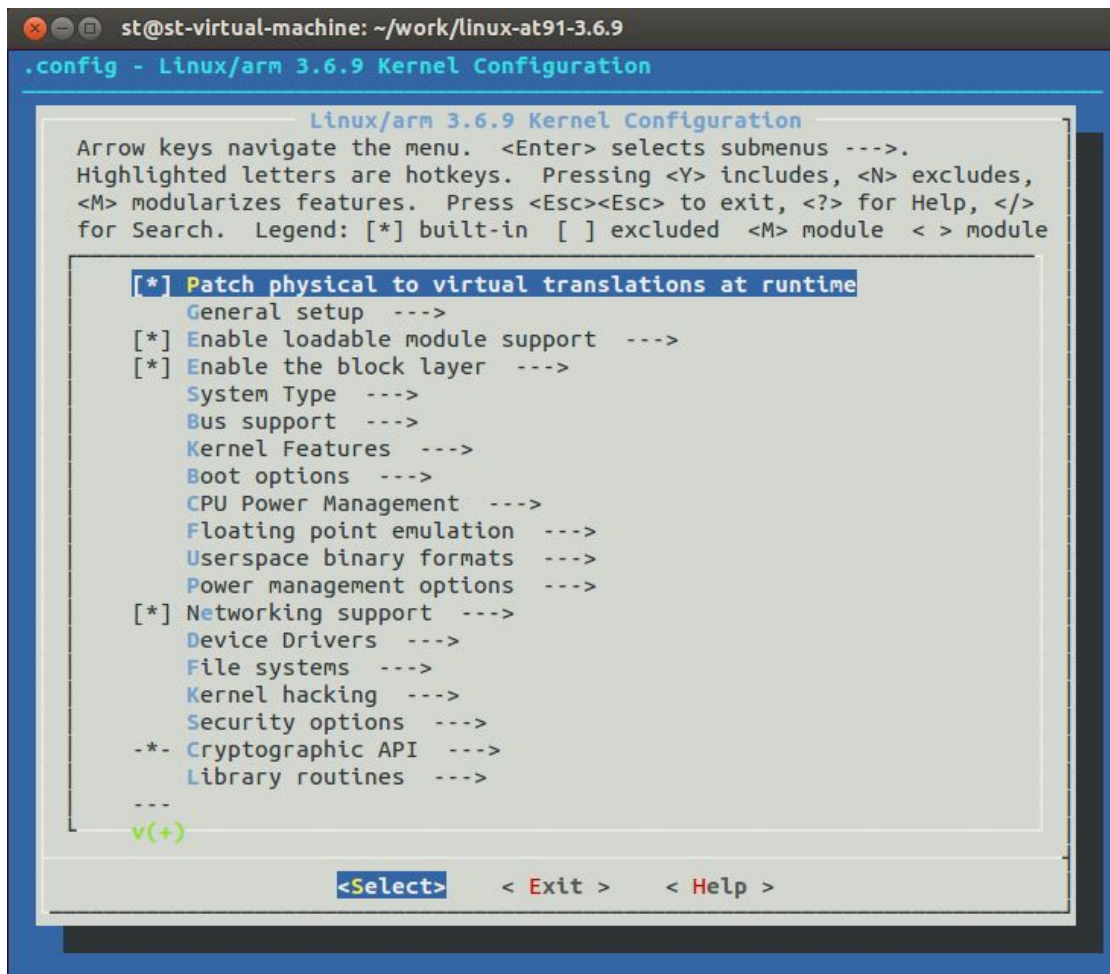
```
$ ls
```

```
st@st-virtual-machine:~/work$ cd linux-at91-3.6.9/
st@st-virtual-machine:~/work/linux-at91-3.6.9$ ls
arch      Documentation  ipc           Makefile      samples      virt
block    drivers        Kbuild       mm            scripts
build_dts.sh  firmware      Kconfig      Module.symvers security
COPYING   fs             kernel       net           sound
CREDITS   include       lib          README        tools
crypto    init          MAINTAINERS  REPORTING-BUGS  usr
st@st-virtual-machine:~/work/linux-at91-3.6.9$
```

编译之前，需要先用如下命令配置内核

```
$ make menuconfig
```

执行之后会弹出如下内核选项配置界面



用户可以对内核功能选项做必要的调整，其他的配置以及裁剪在这里就不予以详细说明，请用户根据自己实际需求来配置。如无特殊需求，直接使用默认内核选项配置来编译内核就可以了。在这里以修改内核分辨率为例子，来做一个简单的修改内核配置的说明。

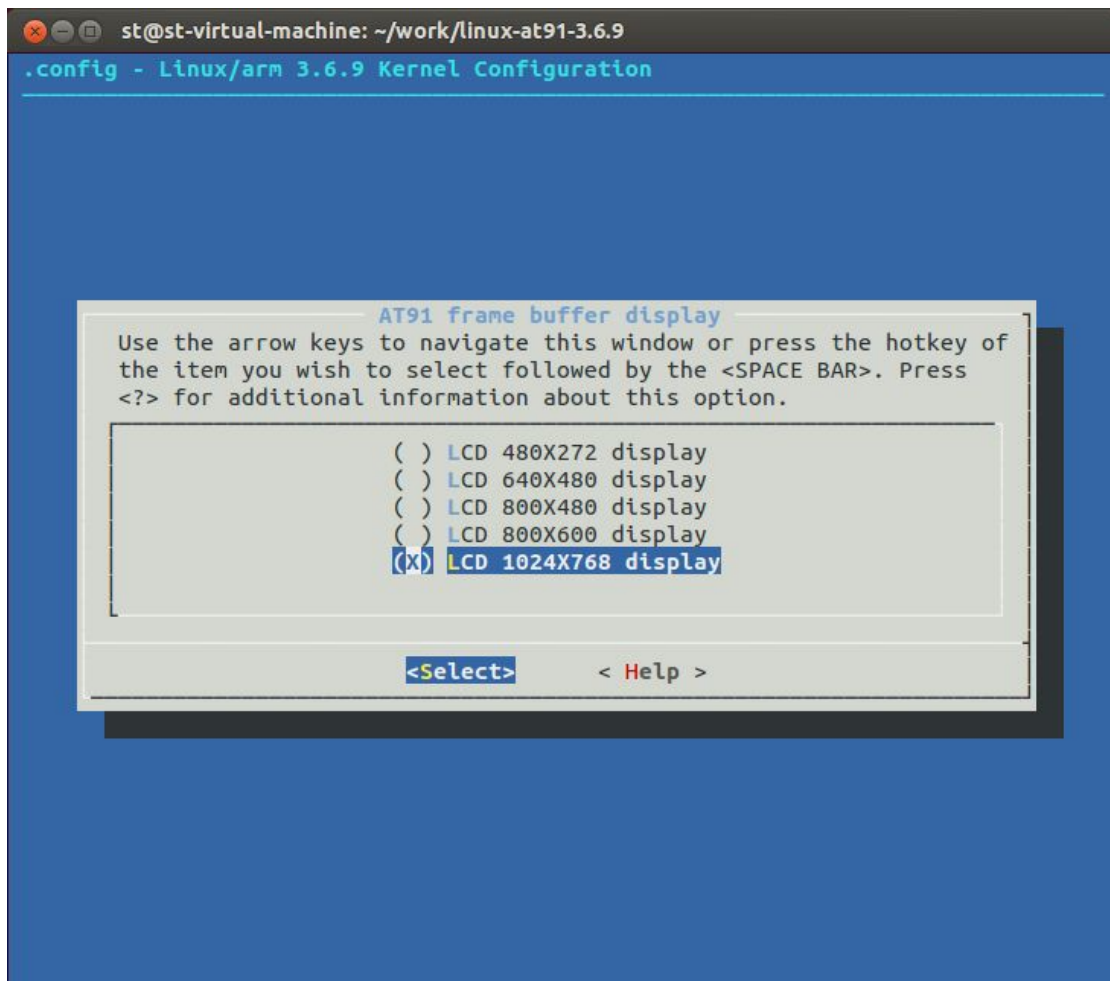
LCD 分辨率选择的内核选项的配置路径：

Device Drivers --->

Graphics supports --->

Support for frame buffer devices --->

AT91 frame buffer display (LCD 1024X768 display)



将默认的配置从 LCD 800X600 display 改成 LCD 1024X768 display, 配置好之后保存退出。

开始编译

\$ **make**

第一次编译可能需要 8~10 分钟, 这里以普通电脑双核 2.4GHz 主频安装虚拟机来作为参考, 不同的电脑或者服务器会存在一些差异。

开始编译内核镜像

\$ **make uImage**

执行之后出现如下错误


```
st@st-virtual-machine: ~/work/linux-at91-3.6.9
st@st-virtual-machine:~/work/linux-at91-3.6.9$ 
st@st-virtual-machine:~/work/linux-at91-3.6.9$ 
st@st-virtual-machine:~/work/linux-at91-3.6.9$ 
st@st-virtual-machine:~/work/linux-at91-3.6.9$ 
st@st-virtual-machine:~/work/linux-at91-3.6.9$ 
st@st-virtual-machine:~/work/linux-at91-3.6.9$ 
st@st-virtual-machine:~/work/linux-at91-3.6.9$ 
st@st-virtual-machine:~/work/linux-at91-3.6.9$ 
st@st-virtual-machine:~/work/linux-at91-3.6.9$ 
st@st-virtual-machine:~/work/linux-at91-3.6.9$ 
st@st-virtual-machine:~/work/linux-at91-3.6.9$ 
st@st-virtual-machine:~/work/linux-at91-3.6.9$ 
st@st-virtual-machine:~/work/linux-at91-3.6.9$ 
st@st-virtual-machine:~/work/linux-at91-3.6.9$ 
st@st-virtual-machine:~/work/linux-at91-3.6.9$ 
st@st-virtual-machine:~/work/linux-at91-3.6.9$ 
st@st-virtual-machine:~/work/linux-at91-3.6.9$ 
st@st-virtual-machine:~/work/linux-at91-3.6.9$ 
st@st-virtual-machine:~/work/linux-at91-3.6.9$ 
st@st-virtual-machine:~/work/linux-at91-3.6.9$ 
st@st-virtual-machine:~/work/linux-at91-3.6.9$ 
st@st-virtual-machine:~/work/linux-at91-3.6.9$ make uImage
CHK      include/linux/version.h
CHK      include/generated/utsrelease.h
make[1]: "include/generated/mach-types.h"是最新的。
CALL     scripts/checksyscalls.sh
CHK      include/generated/compile.h
Kernel:  arch/arm/boot/Image is ready
Kernel:  arch/arm/boot/zImage is ready
UIMAGE   arch/arm/boot/uImage
"mkimage" command not found - U-Boot images will not be built
make[1]: *** [arch/arm/boot/uImage] 错误 1
make: *** [uImage] 错误 2
st@st-virtual-machine:~/work/linux-at91-3.6.9$
```

上图提示缺少 `mkimage` 命令，生成内核镜像需要用到 `mkimage` 工具，刚才已经把 `mkimage` 工具拷贝到工作目录中，需要把它添加到系统环境变量中，以便系统能够自动调用。在这里简单一点，把 `mkimage` 拷贝到交叉编译器的 `bin` 目录下。

```
$ cp ../mkimage ~/work/gcc-linaro-arm-linux-gnueabi-4.7-2012.12-20121214_linux/bin/
```

现在我们可以执行编译命令，顺利进行编译内核镜像

```
$ make uImage
```

```
st@st-virtual-machine:~/work/linux-at91-3.6.9$ cp ../mkimage ~/work/gcc-linaro-arm-linux-gnueabi-4.7-2012.12-20121214_linux/bin/
st@st-virtual-machine:~/work/linux-at91-3.6.9$ make uImage
CHK      include/linux/version.h
CHK      include/generated/utsrelease.h
make[1]: "include/generated/mach-types.h"是最新的。
CALL     scripts/checksyscalls.sh
CHK      include/generated/compile.h
Kernel:  arch/arm/boot/Image is ready
Kernel:  arch/arm/boot/zImage is ready
UIMAGE   arch/arm/boot/uImage
Image Name:      Linux-3.6.9
Created:         Wed Jan 22 10:53:05 2014
Image Type:      ARM Linux Kernel Image (uncompressed)
Data Size:      2643344 Bytes = 2581.39 kB = 2.52 MB
Load Address:   20008000
Entry Point:    20008000
Image arch/arm/boot/uImage is ready
st@st-virtual-machine:~/work/linux-at91-3.6.9$
```

编译完成后在 arch/arm/boot/目录下生成可烧写到开发板的内核镜像文件 uImage

```
Image arch/arm/boot/uImage ts ready
st@st-virtual-machine:~/work/linux-at91-3.6.9$ ls arch/arm/boot/
bootp compressed dts Image install.sh Makefile tftpd32.exe uImage zImage
st@st-virtual-machine:~/work/linux-at91-3.6.9$
```

编译完了之后我们就可以在 arch/arm/boot/目录下找到重新编译好的内核映像 uImage。

编译完 uImage，我们就可以来编译设备树映像了

\$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-hf- dtbs

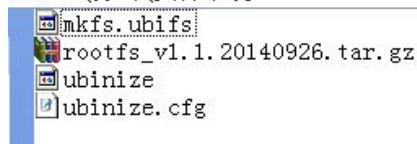
编译完成后，可在 arch/arm/boot/目录下找到 sama5d3x 系列的设备树，其中 sama5d3xek.dtb 即为得到的设备树镜像，其中 x 代表当前使用的 cpu 型号对应数字，如 sama5d34ek.dtb 为 sama5d34 编译得到的设备树镜像。

```
st@st-virtual-machine: ~/work/linux-at91-3.6.9
DTC: dts->dtb on file "arch/arm/boot/dts/at91sam9g15ek.dts"
DTC arch/arm/boot/at91sam9g25ek.dtb
DTC: dts->dtb on file "arch/arm/boot/dts/at91sam9g25ek.dts"
DTC arch/arm/boot/at91sam9g35ek.dtb
DTC: dts->dtb on file "arch/arm/boot/dts/at91sam9g35ek.dts"
DTC arch/arm/boot/at91sam9x25ek.dtb
DTC: dts->dtb on file "arch/arm/boot/dts/at91sam9x25ek.dts"
DTC arch/arm/boot/at91sam9x35ek.dtb
DTC: dts->dtb on file "arch/arm/boot/dts/at91sam9x35ek.dts"
DTC arch/arm/boot/sama5d31ek.dtb
DTC: dts->dtb on file "arch/arm/boot/dts/sama5d31ek.dts"
DTC arch/arm/boot/sama5d31ek_pda.dtb
DTC: dts->dtb on file "arch/arm/boot/dts/sama5d31ek_pda.dts"
DTC arch/arm/boot/sama5d33ek.dtb
DTC: dts->dtb on file "arch/arm/boot/dts/sama5d33ek.dts"
DTC arch/arm/boot/sama5d33ek_pda.dtb
DTC: dts->dtb on file "arch/arm/boot/dts/sama5d33ek_pda.dts"
DTC arch/arm/boot/sama5d34ek.dtb
DTC: dts->dtb on file "arch/arm/boot/dts/sama5d34ek.dts"
DTC arch/arm/boot/sama5d34ek_pda.dtb
DTC: dts->dtb on file "arch/arm/boot/dts/sama5d34ek_pda.dts"
DTC arch/arm/boot/sama5d35ek.dtb
DTC: dts->dtb on file "arch/arm/boot/dts/sama5d35ek.dts"
st@st-virtual-machine:~/work/linux-at91-3.6.9$ ls arch/arm/boot/
aks-cdu.dtb at91sam9x25ek.dtb kizbox.dtb tftpd32.exe
at91sam9263ek.dtb at91sam9x35ek.dtb Makefile tny_a9260.dtb
at91sam9g15ek.dtb bootp sama5d31ek.dtb tny_a9263.dtb
at91sam9g20ek_2mmc.dtb compressed sama5d31ek_pda.dtb tny_a9g20.dtb
at91sam9g20ek.dtb dts sama5d33ek.dtb uImage
at91sam9g25ek.dtb ethernut5.dtb sama5d33ek_pda.dtb usb_a9260.dtb
at91sam9g35ek.dtb evk-pro3.dtb sama5d34ek.dtb usb_a9263.dtb
at91sam9m10g45ek.dtb Image sama5d34ek_pda.dtb usb_a9g20.dtb
at91sam9n12ek.dtb install.sh sama5d35ek.dtb zImage
st@st-virtual-machine:~/work/linux-at91-3.6.9$
```

八、文件系统制作

光盘中有制作好的的文件系统源码和制作工具

3X-Kit\源码\文件系统

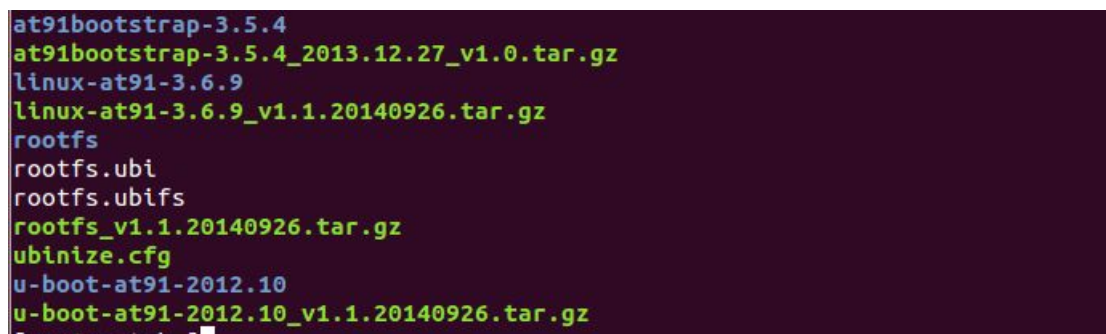


将文件系统源码和制作工具拷贝到~/work 工作目录下，并解压

文件系统源码需要 root 权限才能进行完整解压，在解压命令之前加上 sudo

```
$ sudo rootfs_v1.1.20140926.tar.gz
```

解压后生成 rootfs 文件夹



把制作工具 ubinize 和 mkfs.ubifs 拷贝到交叉编译器的 bin 目录下。

```
$ cp ubinize ~/work/gcc-linaro-arm-linux-gnueabi-4.7-2012.12-20121214_linux/bin/
```

```
$ cp mkfs.ubifs ~/work/gcc-linaro-arm-linux-gnueabi-4.7-2012.12-20121214_linux/bin/
```

生成 rootfs.ubifs

```
$ sudo mkfs.ubifs -r rootfs -m 2048 -e 126976 -c 1816 -o rootfs.ubifs
```

生成 rootfs.ubi

```
$ sudo ubinize -o rootfs.ubi -m 2048 -p 128KiB -s 2048 ./ubinize.cfg
```



```
st@st-virtual-machine:~/work$ sudo mkfs.ubifs -r rootfs -m 2048 -e 126976 -c 181
6 -o rootfs.ubifs
st@st-virtual-machine:~/work$ sudo ubinize -o rootfs.ubi -m 2048 -p 128KiB -s 20
48 ./ubinize.cfg
st@st-virtual-machine:~/work$ ls
at91bootstrap-3.5.4
at91bootstrap-3.5.4_2013.12.27_v1.0.tar.gz
gcc-linaro-arm-linux-gnueabi-hf-4.7-2012.12-20121214_linux
gcc-linaro-arm-linux-gnueabi-hf-4.7-2012.12-20121214_linux.tar.bz2
linux-at91-3.6.9
linux-at91-3.6.9_2014.01.03_v1.0.tar.gz
mkfs.ubifs
mkimage
rootfs
rootfs-2014.2.8_v2.1.tar.gz
rootfs.ubi
rootfs.ubifs
ubinize
ubinize.cfg
u-boot-at91-2012.10
u-boot-at91-2012.10_2013.12.27_v1.0.tar.gz
st@st-virtual-machine:~/work$
```

生成的 rootfs.ubi 就是我们可以烧写到开发板的 ubi 文件系统镜像

九、应用程序的开发

您可以在 PC 机上自由地开发应用程序，这里以最简单的 Hello World 为例。在~/work 目录下创建 app 文件夹，并进入 app 文件夹

```
$ mkdir app
```

```
$ cd app
```

首先编写一个 Hello World 程序代码如下：

```
#include <stdio.h>

int main(void)
{
    printf("Hello World !\n");
    return 0;
}
```

保存在 hello.c 文件

```
st@st-virtual-machine:~/work/app$ vi hello.c
st@st-virtual-machine:~/work/app$
st@st-virtual-machine:~/work/app$
st@st-virtual-machine:~/work/app$
st@st-virtual-machine:~/work/app$ ls
hello.c
st@st-virtual-machine:~/work/app$ cat hello.c
#include <stdio.h>
int main(void)
{
    printf("Hello World ! \n");
    return 0;
}
st@st-virtual-machine:~/work/app$
```

要使得程序能够运行在开发板上，需要使用之前安装好的交叉编译器对应用程序进行编译，编译命令如下：

```
$ arm-linux-gnueabi-gcc -o hello hello.c
```

```
$ file hello
```

```
st@st-virtual-machine:~/work/app$ arm-linux-gnueabi-gcc -o hello hello.c
st@st-virtual-machine:~/work/app$ ls
hello hello.c
st@st-virtual-machine:~/work/app$ file hello
hello: ELF 32-bit LSB executable, ARM, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.31, BuildID[sha1]=0x31f8d5b22d847b154ef6a69a87bd2d3091aadd9, not stripped
st@st-virtual-machine:~/work/app$
```

编译后可以看到在当前目录下生成了用于 ARM 平台的可执行的二进制文件 hello。

接下来我们把 hello 这个可执行程序通过 SD、U 盘、tftp 下载或者 nfs 挂载的方式拷贝到开发板，就可以在开发板上执行 hello 程序了，拷贝过程在这里不予以详细介绍！

十、结束语

以上内容可能不够详实，有任何技术问题或建议，欢迎联系我们：supports@qiyangtech.com，也可登录我司论坛进行交流：<http://www.qiytech.com/bbs/>，关于更多产品的信息，欢迎联系销售 sales@qiyangtech.com，或登录 <http://www.qiytech.com/index.html>。

杭州启扬智能科技有限公司

电话：0571-87858811 / 87858822

传真：0571-89935912

支持：0571-89935913

E-MAIL: supports@qiyangtech.com

网址： <http://www.qiyangtech.com>

地址：杭州市西湖科技园西园一路 8 号 3A 幢 5 层

邮编：310012