构建自己的 Cubieboard Debian Linux

soloforce 汇编整理

2013 年 12 月 28 日

摘要

本文在 x86-64 Ubuntu Linux 上为 Cubieboard——包括 A10-Cubieboard、A20-Cubieboard2 以及最新的 A20-Cubieboard3(即 Cubietruck) 构建一个基于 armhf 的 Debian Linux，包括 SPL、U-BOOT、内核 (Kernel)、根系统 (ROOTFS)。授人以渔是就是授人以自由——本文合适的阅读者是喜欢操刀折腾系统的爱好者或者相关从业者，如果您仅仅想体验一下 Cubieboard 系列产品或是直接利用 Cubietech 团队及社区的成果，那么您可以选择安装官方发布版或社区发布版。Cubieboard 是一套出色的开源硬件平台方案，而得益于开源社区的群策群力、协和共荣，Cubieboard 软件系统也可以通过自己下载现成的软件或代码配置、编译起来，最终形成一个可用的嵌入式 Linux 系统。

关键字: 嵌入式,Cubieboard, 构建, Debian Linux

目录

1 环境准备

本文在一台运行着 Lubuntu64-12.10 上开始构建目标系统；若非特别说明，下文的构建方法适用 Cubieboard1、Cubieboard2 以及 Cubieboard3，您必须明确自己使用的板子类型，并选择相对应的构建方法。为了最大限度地节约时间，我们先把必须下载的东西下载好，然后再进一步阐释；要下载的内容分为三部分，这三部分彼此不干扰，所以可以同时进行。我们用 root 用户在指定的目录下进行所有操作：

- 工作目录为 $WORK_DIR
- 目标系统 rootfs 目录为 $ROOTFS_DIR

笔者的设定如下：

```
# WORK_DIR=/home/soloforce/develop/cubieboard
# ROOTFS_DIR=${WORK_DIR}/chroot-armhf
```

此外，本文用阴影区域代表命令或文件内容；因为页幅限制，有的行末尾有"\" 折行符，表示该行尚未结束，在下一行继续——所以折行符不算是命令或文件内容的一部分。此外，本文中用到的命令或代码、脚本片段虽然用拷贝、粘贴的方式更省事，但切记一些关键的文字不能照搬照抄，比如设备名，如''/dev/sdc'' 之类的，请务必根据实际情况做修改，以免造成不可回复的损失。

1.1 下载必须的工具软件

```
# apt-get install build-essential libncurses5-dev u-boot-tools \
  qemu-user-static debootstrap git binfmt-support libusb-1.0-0-dev pkg-config
# apt-get install gcc-arm-linux-gnueabihf
```

1.2 下载源码

从 github 下载 spl&u-boot、内核源码、sunxi 工具包等。注意内核源码超过 1.5G，耗时最长。如果您曾经下载过这些代码，记得分别用 git pull 更新后再进行后续操作，因为代码仓库每天都有变化。

```
# cd ${WORK_DIR}
# git clone https://github.com/cubieboard/u-boot-sunxi.git -b cubie/sunxi
# git clone https://github.com/cubieboard/sunxi-tools.git
# git clone https://github.com/cubieboard/sunxi-boards.git
```

下载下载内核源码，现在 Cubieboard1, Cubieboard2, Cubieboard3 的内核源码都可以在 cubieboard 内核源码仓库上找到了。以 sunxi-3.4 稳定版本的分支为例，用 git 下载到本地。

```
# git clone https://github.com/cubieboard/linux-sunxi.git -b cubie/sunxi-3.4
```

1.3 下载并配置 Debian 基础系统

```
# mkdir ${ROOTFS_DIR}
# cd ${ROOTFS_DIR}
# debootstrap --foreign --arch armhf wheezy .
# cp /usr/bin/qemu-arm-static usr/bin/
# LC_ALL=C LANGUAGE=C LANG=C chroot . /debootstrap/debootstrap --second-stage
# LC_ALL=C LANGUAGE=C LANG=C chroot . dpkg --configure -a
```

到此为止，Debian 基础系统已经配置好了；现在可以把 $ROOTFS_DIR 保存为一个压缩包，以备日后之用。

```
# cd ${ROOTFS_DIR}
# tar jcpvf ../debian-rootfs-armhf-clean.tar.bz2 *
```

2 编译组件

2.1 编译 U-BOOT

```
# cd ${WORK_DIR}/u-boot-sunxi
# make distclean CROSS_COMPILE=arm-linux-gnueabihf-
```

对于 Cubieboard1，这样编译

```
# make cubieboard CROSS_COMPILE=arm-linux-gnueabihf-
```

对于 Cubieboard2，则这样编译

```
# make cubieboard2 CROSS_COMPILE=arm-linux-gnueabihf-
```

对于 Cubietruck(Cubieboard3)，则这样编译

```
# make cubietruck CROSS_COMPILE=arm-linux-gnueabihf-
```

2.2 编译 sunxi-tools

sunxi-tools 提供了 fexc、nand-part 等工具，在系统安装、定制的过程中可能会用到，所以可以先编译它们。因为这些工具常常在宿主系统上 (x86-64 Ubuntu Linux) 来针对目标设备、目标系统运行，所以可以把它们编译为 x86-64 的可执行文件。

```
# cd ${WORK_DIR}/sunxi-tools
# make
```

2.3 配置、编译内核

2.3.1 拷贝预设内核配置文件

对于 Cubieboard1

```
# cd ${WORK_DIR}/linux-sunxi
# cp arch/arm/configs/sun4i_defconfig .config
```

对于 Cubieboard2 和 Cubieboard3, 则如下

```
# cd ${WORK_DIR}/linux-sunxi
# cp arch/arm/configs/sun7i_defconfig .config
```

2.3.2 配置内核

下面开始配置内核，这是一个非常冗繁的过程，您必须足够耐心和仔细；而且，配置一个新的内核往往做不到一次性成功，可能需要多次测试内核才能达到您的目的——还是那句话：要足够耐心和仔细！

```
# make ARCH=arm menuconfig
```

配置的选项很关键，要着重注意以下几项：

- SATA 硬盘支持
- GPIO 支持
- 无线网络支持
- USB 网卡支持
- Tun/Tap 设备支持
- 摄像头支持
- 红外线支持
- USB 串口设备支持
- HDMI 输出支持

下面是一些笔者为 Cubieboard2 的内核进行配置的截图

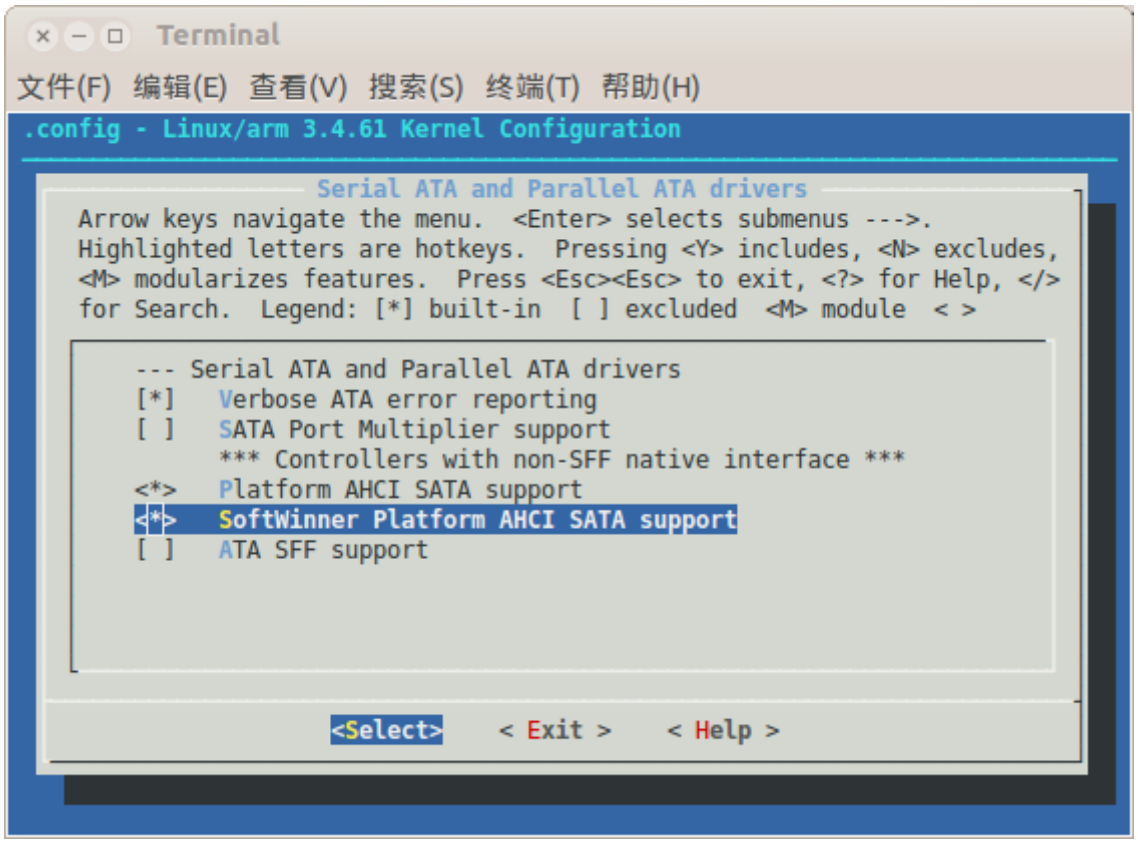

图 1: Device Drivers ---> Serial ATA and Parallel ATA drivers ---> SoftWinner Platform AHCI SATA support

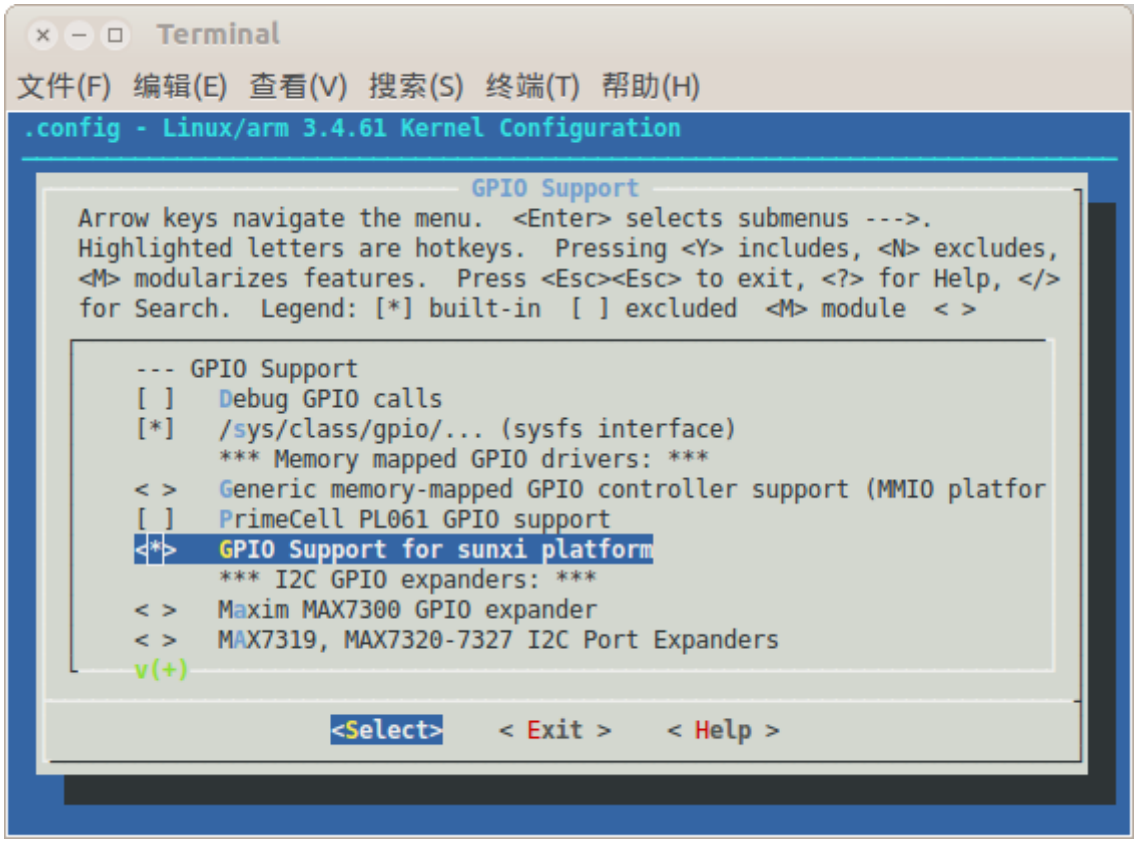

图 2: Device Drivers ---> GPIO Support ---> GPIO Support for sunxi platform

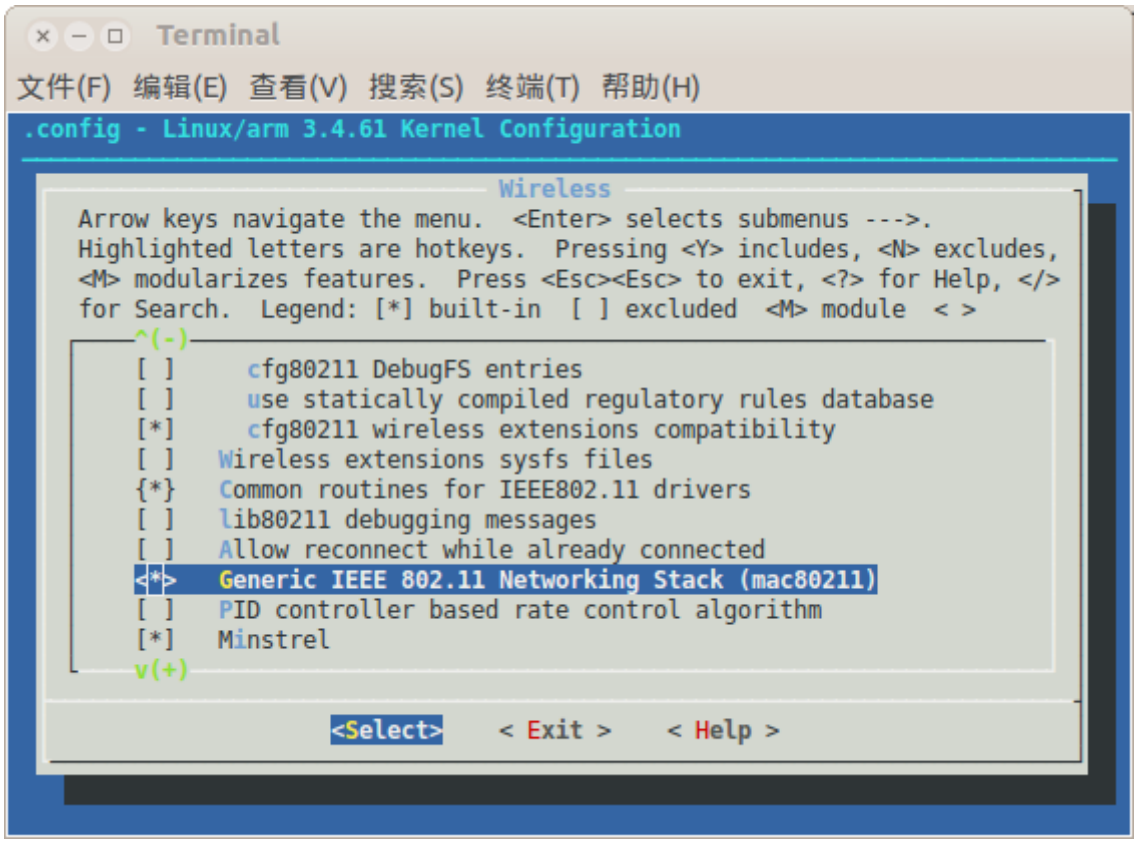

图 3: Networking support ---> Wireless ---> Generic IEEE 802.11 Networking Stack (mac80211)

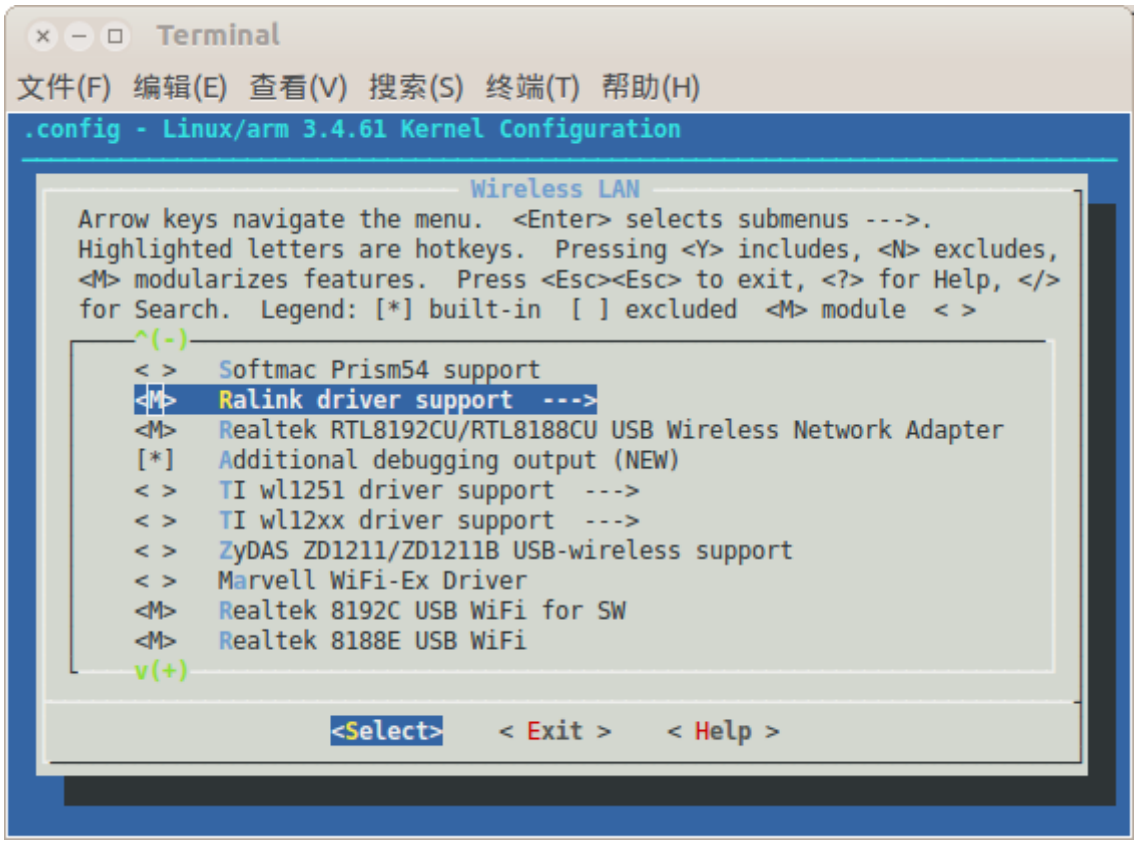

图 4: Device Drivers ---> Network device support ---> Wireless LAN ---> Ralink driver support

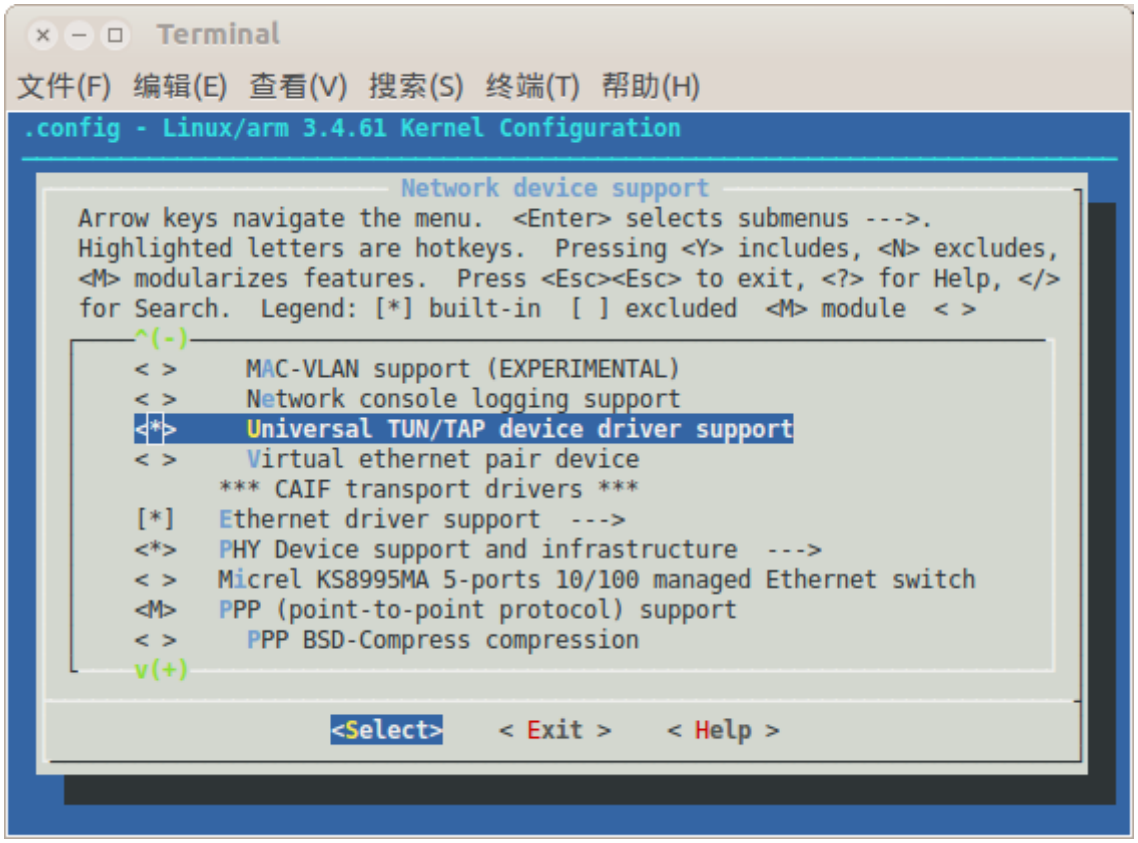

图 5: Device Drivers ---> Network device support ---> Universal TUN/TAP device driver support

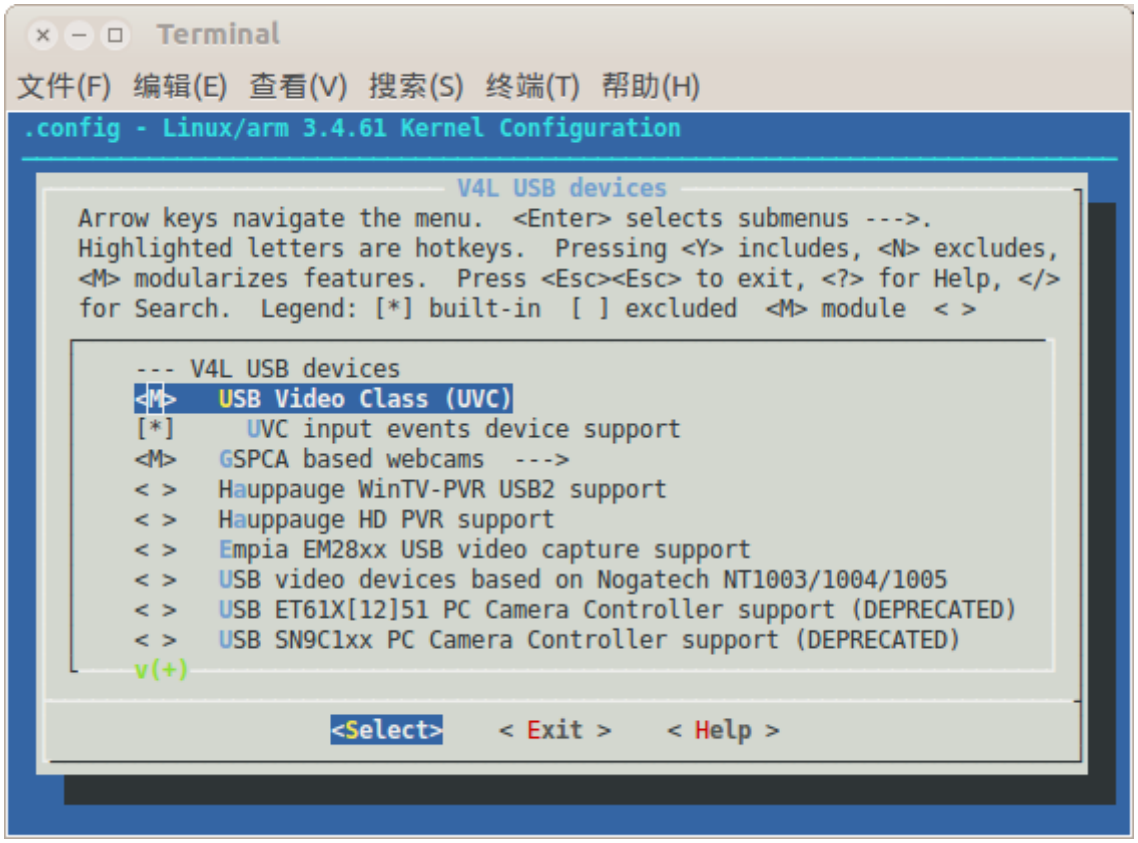

图 6: Device Drivers ---> Multimedia support ---> Video capture adapters ---> V4L USB devices ---> USB Video Class (UVC)

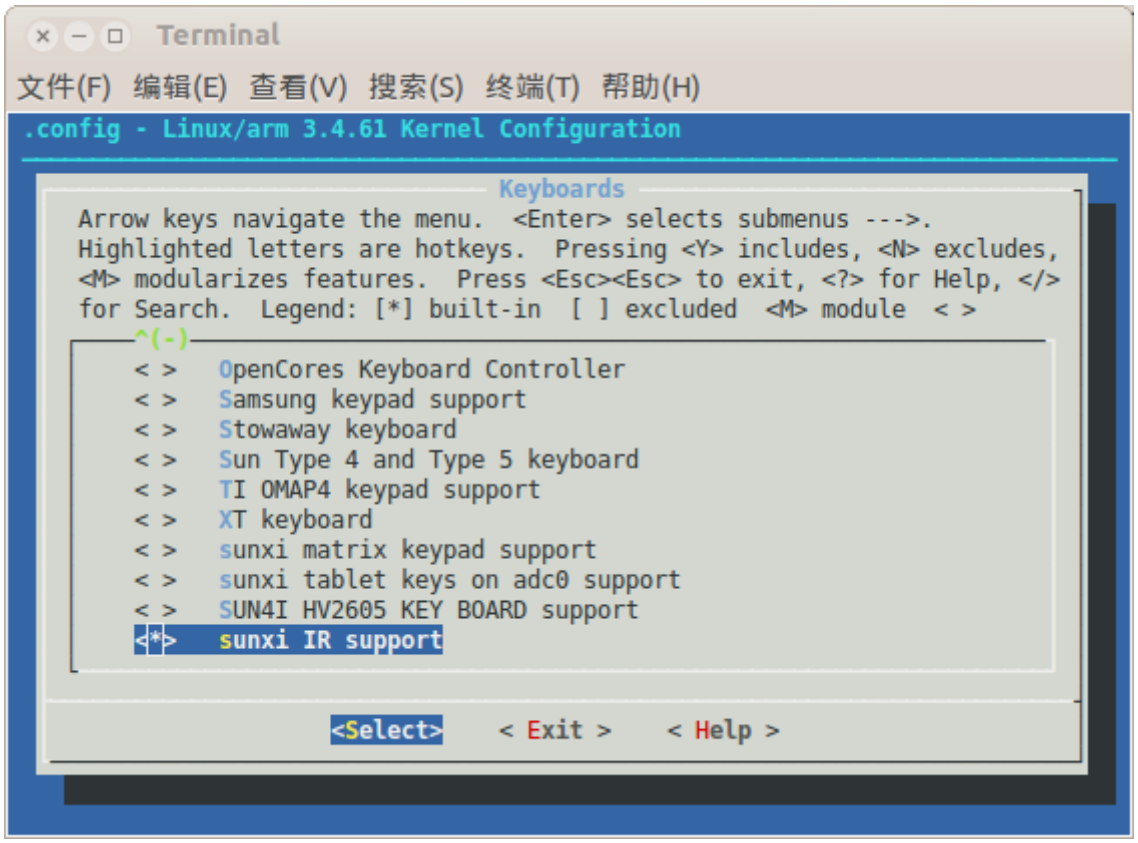

图 7: Device Drivers ---> Input device support ---> Keyboards ---> sunxi IR support

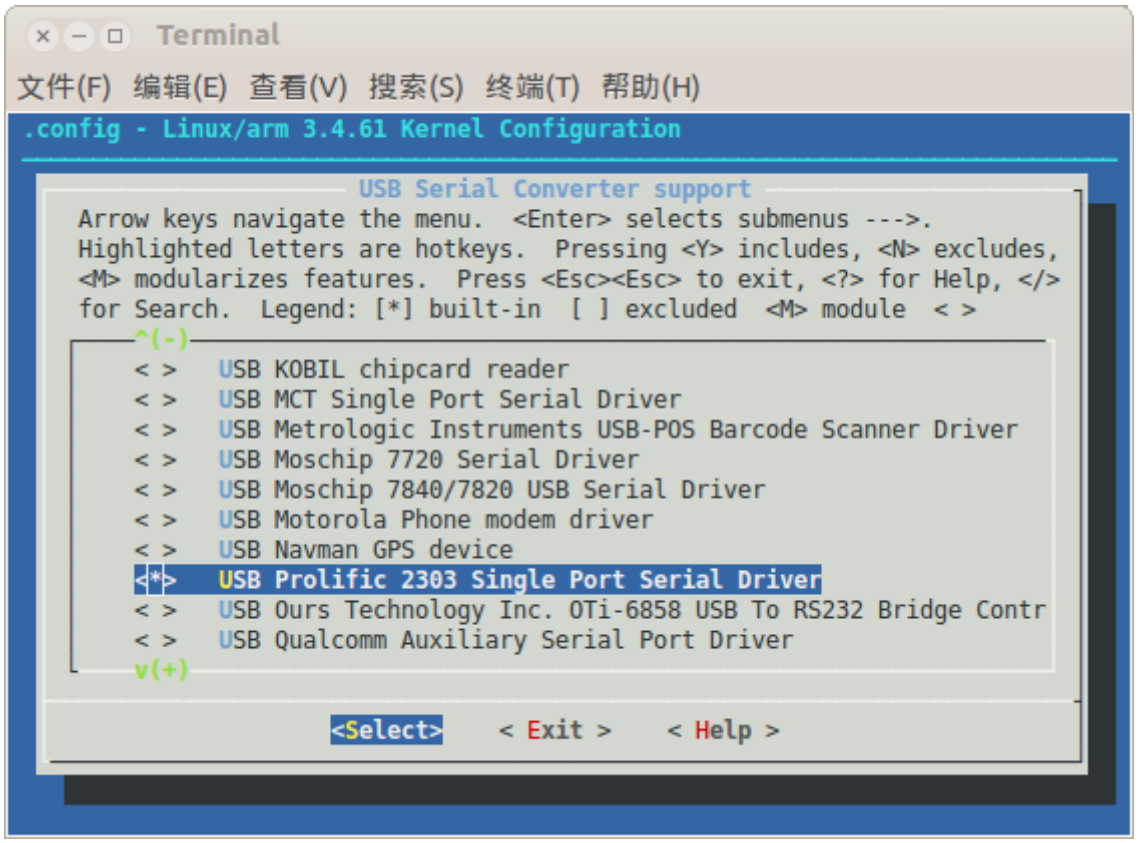

图 8: Device Drivers ---> USB support ---> USB Serial Converter support ---> USB Prolific 2303 Single Port Serial Driver

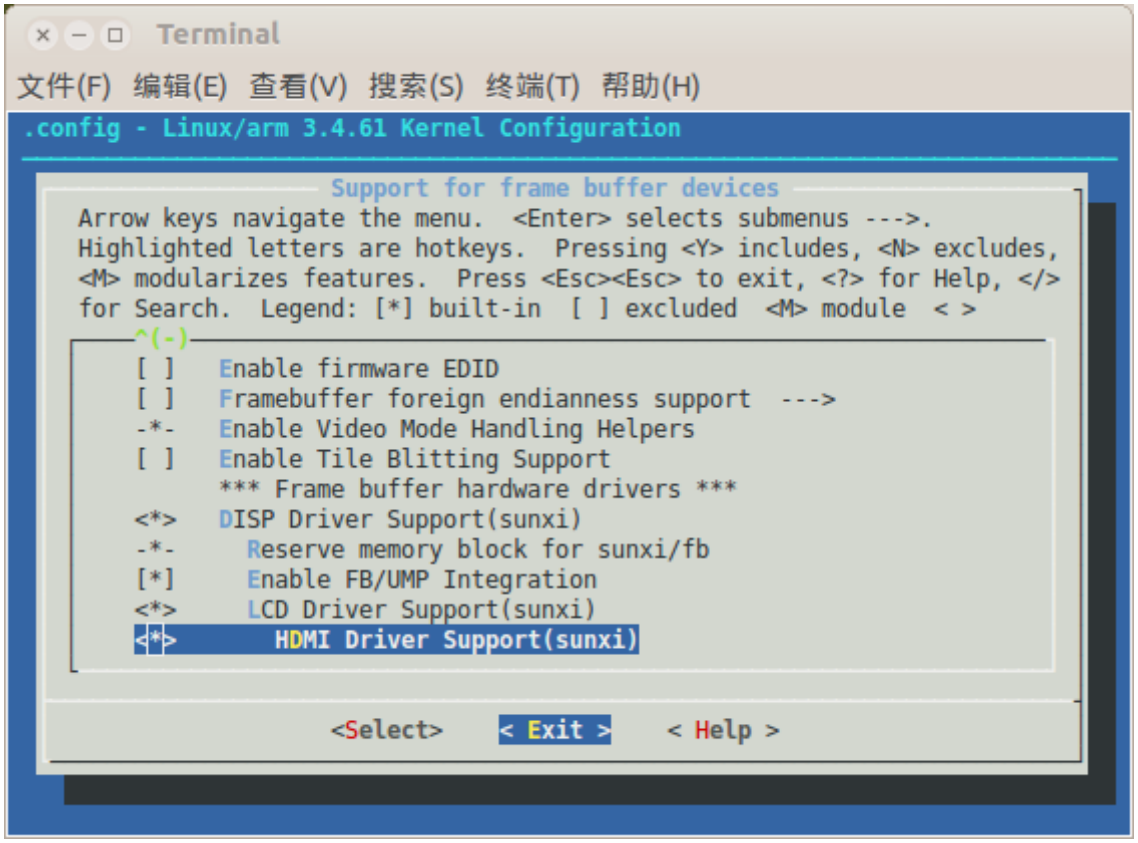

图 9: Device Drivers ---> Graphics support ---> Support for frame buffer devices ---> HDMI Driver Support(sunxi)

2.3.3 编译内核

现在可以开始编译内核及模块了

```
# make -j5 ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- uImage modules
```

2.3.4 发掘更多的可用内存

Cubieboard 的默认内核配置文件中提供了 mali-GPU 支持，并为 X11、DRI、Framebuffer 保留了 200 多兆内存，这些内存将无法被用户使用；如果想把 Cubieboard 作为一个无显示输出的服务器使用，那么这些保留内存应当释放出来供用户使用，此时必须修改内核的配置，使其中包含如下选项（修改时可在 menuconfig 中按'/' 键来搜索相关选项），然后再重新编译内核：

```
CONFIG_SUN4I_G2D=n
CONFIG_FB_SUNXI_RESERVED_MEM=n
CONFIG_SUN4I_G2D_MODULE=n
CONFIG_FB_SUNXI_RESERVED_MEM=n
CONFIG_SUNXI_MALI_RESERVED_MEM=n
CONFIG_DRM_MALI=n
CONFIG_MALI=n
CONFIG_MALI400=n
```

3 建立 ROOTFS

除非特别说明，本节的所有操作都在 $ROOTFS_DIR 下面进行，所以我们进入该目录先。首先且最重要的是设置管理员密码。

```
# cd ${ROOTFS_DIR}
# chroot . passwd
```

3.1 重要步骤

Debian rootfs 基础系统已经在前文下载安装，现在要进一步配置。和主机名称：

```
# echo "Cubieboard" > ${ROOTFS_DIR}/etc/hostname
# echo "127.0.0.1    Cubieboard" >> ${ROOTFS_DIR}/etc/hosts
```

内核模块安装，这是最关键步骤：

```
# cp ${WORK_DIR}/linux-sunxi/arch/arm/boot/uImage ${ROOTFS_DIR}/boot/
# make -C ${WORK_DIR}/linux-sunxi INSTALL_MOD_PATH=${ROOTFS_DIR} ARCH=arm \
    CROSS_COMPILE=arm-linux-gnueabihf- modules_install
```

设置串口调试终端参数，这也是很重要的哦：

```
# echo T0:2345:respawn:/sbin/getty -L ttyS0 115200 vt100 >> etc/inittab
```

配置并更新软件源：

```
# echo deb http://http.debian.net/debian/ wheezy main  contrib non-free > etc/apt/sources.list
# echo deb http://security.debian.org/ wheezy/updates main  contrib non-free >> etc/apt/sources.list
# chroot . apt-get update
# chroot . apt-get upgrade
```

3.2 可选步骤

安装 Open SSH Server

```
# chroot . apt-get install openssh-server
```

安装 locales

```
# chroot . apt-get install locales
# echo "en_US.UTF-8 UTF-8" > etc/locale.gen
# echo "zh_CN.UTF-8 UTF-8" >> etc/locale.gen
# chroot . locale-gen
```

您还可以依样安装其他软件，比如 wifi tools、wpasupplicant、USB 无线网卡的固件（本人用的是巴法络 (BUFFALO) WLI-UC-GNM NANO 迷你无线网卡，一款 mini 型支持软 AP 的 USB 无线网卡)

```
# chroot . apt-get install wireless-tools wpasupplicant firmware-ralink
```

3.3 生成内核启动参数文件

Cubieboard 支持多种启动参数设定方案，本文介绍两种。笔者采用的是方案二，因为修改参数更为简便:-)

- 方案一：利用 boot.scr。

编辑 $ROOTFS_DIR/boot/boot.cmd 如下 (切记具体内容要根据用户实际情况而定，如 root 分区的位置设定):

```
setenv bootargs console=ttyS0,115200 hdmi.audio=EDID:0 disp.screen0_output_mode=EDID:1280x1024p60 \
    root=/dev/mmcblk0p1 rootwait panic=10 ${extra}
ext2load mmc 0 0x43000000 boot/script.bin
ext2load mmc 0 0x48000000 boot/uImage
bootm 0x48000000
```

然后生成 boot.scr

```
# mkimage -C none -A arm -T script -d boot/boot.cmd boot/boot.scr
```

- 方案二：利用 uEnv.txt

我们也可以用 uEnv.txt 文件设定启动参数——不再需要 boot.scr 文件了，删除之。编辑 $ROOTFS_DIR/boot/uEnv.txt (没有就创建一个，切记具体内容要根据用户实际情况而定，如 root 分区的位置)

```
mmcboot=fatload mmc 0 0x43000000 script.bin || fatload mmc 0 0x43000000 evb.bin;  \
    fatload mmc 0 0x48000000 uImage; if fatload mmc 0 0x43100000 uInitrd; \
    then bootm 0x48000000 0x43100000; else bootm 0x48000000; fi
uenvcmd=run mmcboot
bootargs=console=ttyS0,115200 console=tty0 disp.screen0_output_mode=EDID:1280x1024p50 \
    hdmi.audio=EDID:0 root=/dev/mmcblk0p1
```

3.4 生成 Script.bin

我们先把.fex 文件拷贝到 $ROOTFS_DIR/boot 中。对于 Cubieboard1，如下操作：

```
# cp ${WORK_DIR}/sunxi-boards/sys_config/a10/cubieboard.fex boot/script.fex
```

对于 Cubieboard2，如下操作：

```
# cp ${WORK_DIR}/sunxi-boards/sys_config/a20/cubieboard2.fex boot/script.fex
```

对于 CubieTruck，则：

```
# cp ${WORK_DIR}/sunxi-boards/sys_config/a20/cubietruck.fex boot/script.fex
```

3.4.1 配置 GPIO[可选]

喜欢玩外设控制的用户可以在此时配置 GPIO 端口。本文配置了靠近 SATA 口的 30 个可用扩展端口：编辑 $ROOTFS_DIR/boot/script.fex，在最后加上如下一节：

```
[gpio_para]
gpio_used = 1
gpio_num = 30
gpio_pin_1 = port:PD01<1><default><default><default>
gpio_pin_2 = port:PD02<1><default><default><default>
gpio_pin_3 = port:PD03<1><default><default><default>
gpio_pin_4 = port:PD04<1><default><default><default>
gpio_pin_5 = port:PD05<1><default><default><default>
gpio_pin_6 = port:PD06<1><default><default><default>
gpio_pin_7 = port:PD07<1><default><default><default>
gpio_pin_8 = port:PD08<1><default><default><default>
gpio_pin_9 = port:PD09<1><default><default><default>
gpio_pin_10 = port:PD10<1><default><default><default>
```

```
gpio_pin_11 = port:PD11<1><default><default><default>
gpio_pin_12 = port:PD12<1><default><default><default>
gpio_pin_13 = port:PD13<1><default><default><default>
gpio_pin_14 = port:PD14<1><default><default><default>
gpio_pin_15 = port:PD15<1><default><default><default>
gpio_pin_16 = port:PD16<1><default><default><default>
gpio_pin_17 = port:PD17<1><default><default><default>
gpio_pin_18 = port:PD18<1><default><default><default>
gpio_pin_19 = port:PD19<1><default><default><default>
gpio_pin_20 = port:PD20<1><default><default><default>
gpio_pin_21 = port:PD21<1><default><default><default>
gpio_pin_22 = port:PD22<1><default><default><default>
gpio_pin_23 = port:PD23<1><default><default><default>
gpio_pin_24 = port:PD24<1><default><default><default>
gpio_pin_25 = port:PD25<1><default><default><default>
gpio_pin_26 = port:PD26<1><default><default><default>
gpio_pin_27 = port:PD27<1><default><default><default>
gpio_pin_28 = port:PH07<1><default><default><default>
gpio_pin_29 = port:PB10<1><default><default><default>
gpio_pin_30 = port:PB11<1><default><default><default>
```

3.4.2 配置 SPI[可选，仅适用于 Cubieboard1]

如果需要 SPI 支持，可以在内核选项中选中 SPI for sun 支持。然后再修改 script.fex 并重新生成 script.bin 即可。编辑 $ROOFS_DIR/boot/script.fex，修改相关内容如下：

```
[spi0_para]
spi_used = 1
spi_cs_bitmap = 1
spi_cs0 = port:PI10<2><default><default><default>
spi_sclk = port:PI11<2><default><default><default>
spi_mosi = port:PI12<2><default><default><default>
spi_miso = port:PI13<2><default><default><default>

[spi_devices]
spi_dev_num = 1

[spi_board0]
modalias = "spidev"
max_speed_hz = 12000000
bus_num = 0
chip_select = 0
mode = 3
full_duplex = 0
manual_cs = 0
```

3.4.3 生成 **script.bin**

好的，又到关键的一步了，深呼吸一下，键入下面的指令：

```
# ${WORK_DIR}/sunxi-tools/fex2bin boot/script.fex boot/script.bin
```

3.5 设置网络

现在设置网络参数，编辑 etc/network/interfaces，下面的配置是动态获取 IP 的设置

```
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp
```

下面的配置则是设定固定 IP 的设置

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.0.0.10
netmask 255.255.255.0
network 10.0.0.0
broadcast 10.0.0.255
gateway 10.0.0.1
```

如果需要设定无线网卡，可以在追加如下配置（假定无线网卡的设备名为 wlan0，且系统要安装软件包 wireless-tools、wpasupplicant ）

```
auto wlan0
iface wlan0 inet dhcp
pre-up ip link set wlan0 up
pre-up iwconfig wlan0 essid your-ap-ssid
wpa-ssid your-ap-ssid
wpa-psk your-ap-passwd
```

4 安装到 TF 卡

笔者的 TF 卡设备名被认作/dev/sdc，实际值要视你自己的情况而定，一定要搞清楚 TF 卡对应的设备名，以避免不必要的损失。

```
# CARD=/dev/sdc
```

4.1 安装 u-boot

清空前面 1M 左右的数据，为存放 u-boot 预留空间

```
# cd ${WORK_DIR}/u-boot-sunxi
# dd if=/dev/zero of=$CARD bs=1k count=1024
# dd if=u-boot-sunxi-with-spl.bin of=$CARD bs=1024 seek=8
```

4.2 介质分区

为 TF 卡分区, 本文没有用 sfdisk 工具，而是用了 fdisk 工具。分区有两种方案：

4.2.1 单一分区方案

笔者采用的即是单一分区方案，文件系统格式必须是内核支持的格式（即相关驱动要编译进内核而不是编译成模块），如 ext3、ext4，执行如下操作：

```
# fdisk $CARD
...(进行分区操作)...
# mkfs.ext4 ${CARD}1
```

拷贝整个 ROOTFS 到 TF 卡

```
# cd ${ROOTFS_DIR}
# mount ${CARD}1 /mnt
# tar --exclude=qemu-arm-static -cf - . | tar -C /mnt -xvf -
# sync && umount /mnt
```

4.2.2 两个分区以上的方案

如果把 TF 卡分成两个分区（或两个以上分区，类同），此时最好把第一个分区格式化为 vfat 格式，否则可能无法启动（笔者曾碰到过这样的情形），并且要设定 $ROOTFS_DIR/etc/fstab 内容如下：

```
/dev/mmcblk0p1 /boot vfat defaults 0 2
/dev/mmcblk0p2 / ext4 defaults,noatime 0 1
```

同时记得修改 $ROOTFS_DIR/boot/uEnv.txt，指定 root 分区设备为 mmcblk0p2

```
bootargs=console=ttyS0,115200 console=tty0 disp.screen0_output_mode=EDID:1280x1024p50 \
    hdmi.audio=EDID:0 root=/dev/mmcblk0p2
```

然后拷贝整个 ROOTFS 到 TF 卡

```
# cd ${ROOTFS_DIR}
# mount ${CARD}2 /mnt
# mkdir /mnt/boot
# mount ${CARD}1 /mnt/boot
# tar --exclude=qemu-arm-static -cf - . | tar -C /mnt -xvf -
# sync && umount /mnt/boot && umount /mnt
```

如果没有意外的话——您自己动手丰衣足食得到的 Cubieboard Debian Linux TF 卡版本或许已经可以运行了，只是还没把 TF 卡插到 Cubieboard 上测试而已！

5 Cubietruck 相关

5.1 关于 WIFI

Cubietruck 采用 AP6210 的 Bluetooth+WIFI 模块 (bchdmd)，所以在编译内核的时候必须选中以下驱动：

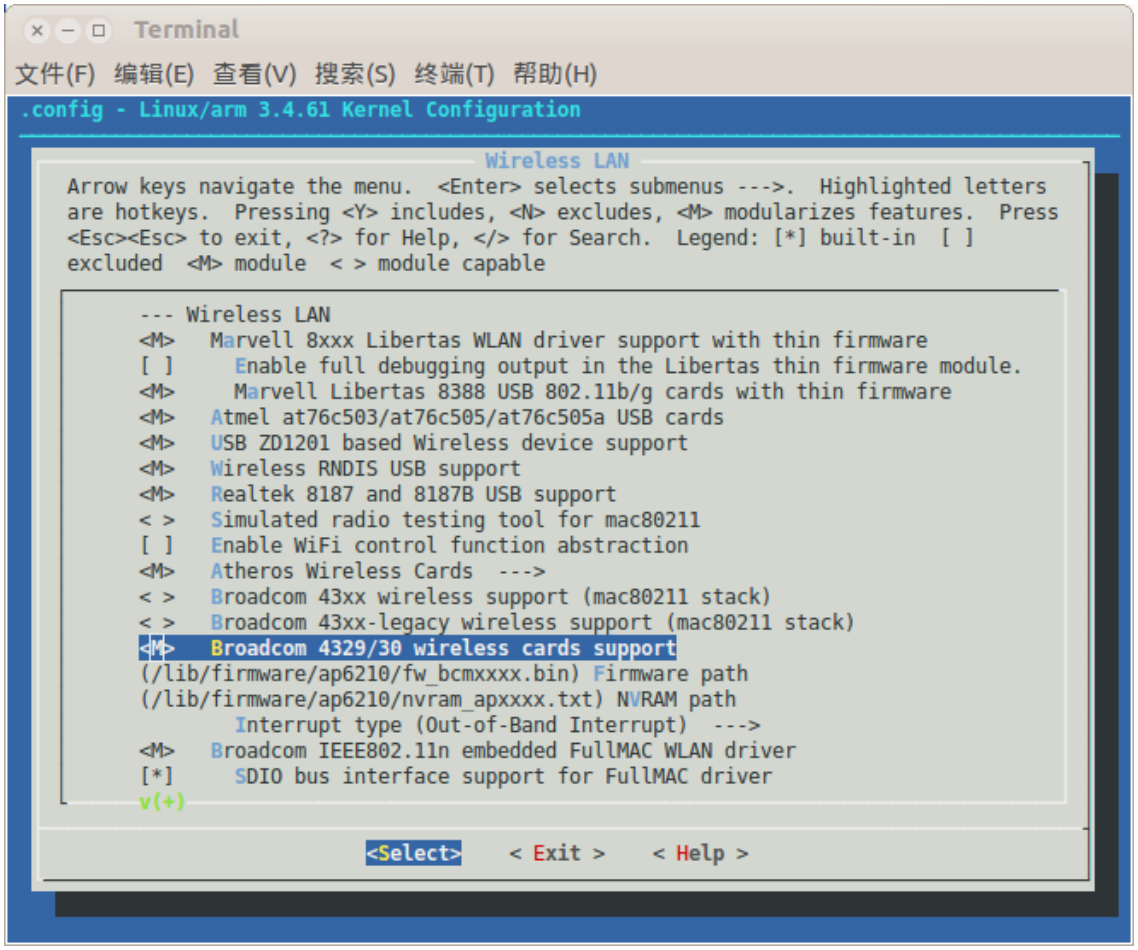

图 10: Device Drivers ---> Network device support ---> Wireless LAN ---> Broadcom 4329/30 wireless cards support

还有几点要注意：首先，不要选中 *Enable WiFi control function abstraction* ，可能是相应的功能尚未完善会导致冲突；其次，模块 bchdmd 依赖的 AP6210 固件默认是缺失的，需要自己拷贝到目标系统的/lib/firmware/ap6210/下面，这些固件可以从官方发布的 Lubuntu 系统 (http://dl.cubieboard.org/software/a20-cubietruck/lubuntu/) 中拷贝出来；第三，bchdmd 还依赖 gpio-sunxi，必须在内核中选中之。

为了在启动内核时自动加载 WIFI 相应模块，可以编辑 ***/etc/modules***，加入如下几行

```
gpio_sunxi
bchdmd
```

系统重启后检查 ***/sys/class/net/wlan0*** 是否成功创建，是的话就可以通过 iwconfig、ifconfig、wpasupplicant 等工具配置无线网卡了。

6 安装到 NAND[可选，仅适用于 Cubieboard1]

一旦确认 TF 卡上新构建的 Debian Linux 系统已经处于可用状态，我们可以进一步把这个系统安装到 Cubieboard 内置的 4G Nand Flash 里面，这样做的好处是……总之有好处。如果您希望继续折腾，请坚持看下去——所要做的事情比想像的要简单。
作为区分，"PC#" 表示在构建 Cubieboard Debian Linux 系统的 PC 系统中执行命令，而"TF_Cubie#" 表示安装在 TF 卡的 Cubieboard 系统中执行命令。
首先，把前文生成的 ROOTFS 打包

```
PC# cd ${ROOTFS_DIR}
PC# tar jcvpf ../debian-rootfs-armhf.tar.bz2 ./
```

然后，下载 Cubieboard 的 Nand Bootloader 映像并安装之：

```
TF_Cubie# wget https://cubieboard.googlecode.com/files/cubie_nand_uboot_partition_image.bin
TF_Cubie# dd bs=4096 if=cubie_nand_uboot_partition_image.bin of=/dev/nand
```

上述命令会把 bootloader 以及 boot 相关程序写入/dev/nand 中，并把 nand 分成两个分区 nanda, nandb。为了确保写入成功，重启一下 Cubieboard。

```
TF_Cubie# sync
TF_Cubie# reboot
```

等重启完毕后，就可以对 Nand 进行分区、格式化、挂载等操作了。如果需要重新分区的话可以用 nand-part 工具，在 sunxi-tools 里面可以找到，可能要自行编译。

```
TF_Cubie# mkfs.ext4 /dev/nandb
TF_Cubie# mkdir /mnt/nanda
TF_Cubie# mkdir /mnt/nandb
TF_Cubie# mount /dev/nanda /mnt/nanda
TF_Cubie# mount /dev/nandb /mnt/nandb
```

把前文生成的 debian-rootfs-armhf.tar.bz2，从 PC 上发送到 Cubieboard 并解压缩到 nandb 上，并拷贝 script.bin 到 nanda 分区

```
TF_Cubie# tar jxpvf debian-rootfs-armhf.tar.bz2 -C /mnt/nandb
TF_Cubie# cp /mnt/nandb/boot/script.bin /mnt/nanda/
```

删掉不需要的引导文件

```
TF_Cubie# rm /mnt/boot/uEnv.txt
```

优化 fstab，减少 Nand 的擦写次数，既提速又延长介质的使用时间：把下面的指令加入到/mnt/etc/fstab

```
tmpfs /tmp tmpfs defaults,noatime,mode=1777  0  0
tmpfs /var/tmp tmpfs defaults,noatime,mode=1777  0  0
tmpfs /var/log tmpfs defaults,noatime,mode=0777  0  0
tmpfs /var/log/apt tmpfs defaults,noatime  0  0
```

注意，上面把/var/log 目录设定为 tmpfs，会导致系统及一些应用程序的日志无法持久保存，请酌情采用。
卸载文件系统，关闭系统和电源：

```
TF_Cubie# umount /mnt/nanda
TF_Cubie# umount /mnt/nandb
TF_Cubie# sync && shutdown -h now
```

然后，拔掉 TF 卡，重开电源，启动进入 Nand 系统！

7 创建系统映像【可选】

分享，是的，分享。如果您想把自己制作的系统分享给朋友们，可以参考本节。这里没有给出详细的指令说明，请自行参考相关资源。

```
# cd ${WORK_DIR}
# dd if=/dev/zero of=disk.img count=4000000 （此处为2G大小；您可以自行修改）
# losetup /dev/loop0 disk.img
# dd if=/dev/zero of=/dev/loop0 bs=1k count=1024
# cd ${WORK_DIR}/u-boot-sunxi
# dd if=u-boot-sunxi-with-spl.bin of=/dev/loop0 bs=1024 seek=8
```

下面对映像设备进行分区，并检查一下第一个分区的偏移是否为 2048 个 block (每个 block 为 512Bytes)

```
# fdisk /dev/loop0
...分区动作...这里分两个区,第一个分区64MB,第二个分区占据所有剩余空间...
# fdisk -l /dev/loop0 （查看分区的偏移量）

Disk /dev/loop0: 2048 MB, 2048000000 bytes
255 heads, 63 sectors/track, 248 cylinders, total 4000000 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x1408d14e

      Device Boot      Start         End      Blocks   Id  System
/dev/loop0p1            2048      133119       65536   83  Linux
/dev/loop0p2          133120     3999999     1933440   83  Linux
```

下面就可以把分区挂载起来，并拷贝系统了

```
# cd ${WORK_DIR}
# losetup -d /dev/loop0 && losetup /dev/loop0 disk.img （重新关联一次）
# losetup -o 1048576 /dev/loop1 /dev/loop0 （关联第一分区,1048576=512*2048）
# losetup -o 68157440 /dev/loop2 /dev/loop0 （关联第二分区,68157440=512*133120）
# mkfs.vfat /dev/loop1
# mkfs.ext4 /dev/loop2
# mount /dev/loop2 /mnt
# mkdir /mnt/boot
# mount /dev/loop1 /mnt/boot
# cd ${WORK_DIR}/chroot-armhf
# tar --exclude=qemu-arm-static -cf - . | tar -C /mnt -xvf -
# sync && umount /mnt/boot && umount /mnt
```

```
# losetup -d /dev/loop2
# losetup -d /dev/loop1
# losetup -d /dev/loop0
```

好了，disk.img 已经完成，您可以把它发布到网上了:-> 如果要安装到 TF 卡，执行：

```
# dd if=disk.img of=$CARD bs=4k
```

8 接下来做什么

Cubieboard Debian Linux 的构建要告一段落了，拔下读卡器，拔下 TF 卡，插到 CuebieBoard 上，接上电源 (从 5V 口或者 miniUSB 口接入)，接上 TTL 串口调试线，接通电源……然后，祈祷吧;-) 如果看到字符迅速滚动，最后出现类似 login 的字样，恭喜您！若看到 panic 之类，然后所有的文字卡死不滚动了，那么，也恭喜您！您有机会亲手 debug 一下了:-o

当然，即使系统已经可以启动了，也可能还没有达到您预期的要求，比如某个内核模块忘记编译进去了导致您的 USB 网卡或者 USB WIFI 模块或者蓝牙模块没法工作——别急，您可以返回前面的步骤重新配置内核、编译内核、安装内核及模块……迟早，您的系统在您的手下会越来越好用！

8.1 设置时区和日期

```
# rm /etc/localtime && ln -s /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
# date MMDDhhmmyyyy
```

Cubieboard1 和 Cubieboard2 没有引出 RTC，所以无法通过 hwclock -w 保存系统时间, 解决办法是通过网络来校正:

```
# apt-get install ntpdate
# nano -w /etc/default/ntpdate
```

更改 ntpdate 文件如下:

```
NTPDATE_USE_NTP_CONF=no
NTPSERVERS="0.asia.pool.ntp.org 1.asia.pool.ntp.org 2.asia.pool.ntp.org 3.asia.pool.ntp.org"
NTPOPTIONS=""
```

然后执行

```
# ntpdate-debian
```

现在设置开机启动 ntpdate, 在/etc/rc.local 文件的 exit 0 这一行之前加入:

```
ntpdate-debian &
```

Cubietruck 引出了 RTC，也有板载时钟电池，这样日期设置的问题就自然解决了。但笔者发现，Cubietruck 的 RTC 精度有限，安装个 ntpdate 有助于校正时间，还是有好处的。

8.2 安装其他软件

先记得更新一下软件源，然后就可以放心的 apt-get 了

```
# apt-get update
# apt-get upgrade
# apt-get install whatever-you-want
```

致谢

感谢 Cubietech 团队为我们带来如此好玩的产品！感谢 Cubieboard 社区的每一个人的贡献！本文参考了不少资料文献，在文末给出的相关参考链接未能包含所有出处，请谅解。此外，限于笔者的知识和见识水平，文中必定存在谬误，请方家好手不吝批评赐教！

参考文献

[1] http://linux-sunxi.org/Building_on_Debian

[2] http://linux-sunxi.org/FirstSteps

[3] https://github.com/linux-sunxi/u-boot-sunxi/wiki

[4] https://github.com/cubiebook/cubiebook/blob/master/chapter3/debian/debian.md