博客大赛]U-Boot for AM335x (13) steps of building Linux Kernel

发布时间：2014-09-02 15:38:42

技术类别：嵌入式 个人分类：U-Boot

U-Boot for AM335x (13) steps of building Linux Kernel

这一节我记录了porting Linux Kernel的具体步骤，和以往相差比较大的是driver的DT策略，内核不再将描述driver的代码放置在uImage中了，而是放在\*.dtb文件里面，由u-boot.img传递给内核。

具体的步骤见下（绝对路径、名字神马的，替换成自己的就行）：

====> STEP 1: 安装交叉工具链，它和编译U-Boot的是同一个，Forlinx光盘里面有提供:

arm-arago-linux-gnueabi

也可以在TI的官方网站下载，还可以使用GCC的mainline编译（使用GCC源码编译自己的工具链，也是非常interesting的事情，ps:开源软件说的interesting，就是非常有挑战性的意思，就是可能会让你想咔咔咔砸掉电脑的意思。。。）

http://software-dl.ti.com/sitara\_linux/esd/AM335xSDK/latest/index\_FDS.html

这是TI的官方开发软件的地址，下载这个东东，它里面神马都有：

ti-sdk-am335x-evm-07.00.00.00-Linux-x86-Install.bin

（另外，这里也提供了单独下载的链接，比如制作SD卡的工具啦，am335x的源玛啦，交叉编译器啦。。。）

因为TI官网的版本比Forllinx提供的要高一点，所以我使用了TI提供的，顺便把U-Boot也重新编译了一次。重新运行MLO和u-boot.img，没有遇到问题，意外的非常顺利！另外，使用旧版本的编译最新版本的内核，我出现了下面的错误：

/tmp/cc3G2Hh3.s: Assembler messages:

/tmp/cc3G2Hh3.s:2006: Error: garbage following instruction -- `dmb ishst'

/tmp/cc3G2Hh3.s:2023: Error: garbage following instruction -- `dmb ishst'

/tmp/cc3G2Hh3.s:2055: Error: garbage following instruction -- `dmb ish'

/tmp/cc3G2Hh3.s:2067: Error: garbage following instruction -- `dsb ishst'

总之，以后的步骤还是会使用TI官网提供的4.7.3版本。

====> STEP 2: 从Linux Kernel的官方网站下载最新版本的stable内核，下载完成之后解压，并进入目录，清理它：

tar xvf linux-3.16.1.tar.xz

cd linux-3.16.1

make ARCH=arm CROSS\_COMPILE=/opt/arm-arago-linux-gcc-ti/i686-arago-linux/usr/bin/arm-linux-gnueabihf- -j8 mrproper

要是交叉工具链的bin目录加入了PATH环境变量，那么CROSS\_COMPILE就可以不用写完整的绝对路径。

-j8的意思是，指定同时执行的任务有8个，一般来说，电脑的CPU有几核，就指定它的两倍既可，使用lscpu命令可以查看。

====> STEP 3: 构造自己的config文件：

这是很关键的一步。

make ARCH=arm CROSS\_COMPILE=/opt/arm-arago-linux-gcc-ti/i686-arago-linux/usr/bin/arm-linux-gnueabihf- -j8 omap2plus\_defconfig

make ARCH=arm CROSS\_COMPILE=/opt/arm-arago-linux-gcc-ti/i686-arago-linux/usr/bin/arm-linux-gnueabihf- menuconfig

先调入默认的config文件omap2plus\_defconfig，这样很多配置就可以使用它的，而不用一个个研究了。然后使用menuconfig的方法生成.config，要修改的主要有这么几个地方：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

选项1：General setup --->

Default hostname

填入自己希望的hostname，我填写的是Maria，这个随意，也可以不填。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

选项2：General setup --->

Stack Protector buffer overflow detection (None) --->

选择none，而不是默认的strong，否则编译无法通过，错误信息如下：

cc1: error: unrecognized command line option "-fstack-protector-strong"

（这条是因为不调入omap2plus\_defconfig时，默认x86体系，而arm交叉编译器maybe不支持这个选项，调入了omap2plus\_defconfig时默认是none。）

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

选项3：Boot options --->

Default kernel command string

将root=/dev/mmcblk0p2 rootwait console=ttyO2,115200

修改为：root=/dev/mmcblk0p2 rootwait console=ttyO0,115200

====> STEP 4: 编译\*.dtb文件

make ARCH=arm CROSS\_COMPILE=/opt/arm-arago-linux-gcc-ti/i686-arago-linux/usr/bin/arm-linux-gnueabihf- -j8 am335x-evm.dtb

这一步将在arch/arm/boot/dts/目录下生成am335x-evm.dtb文件。

====> STEP 5: 编译uImage

make ARCH=arm CROSS\_COMPILE=/opt/arm-arago-linux-gcc-ti/i686-arago-linux/usr/bin/arm-linux-gnueabihf- -j8 LOADADDR=0x80008000 uImage

这一步将在这一步将在arch/arm/boot/下生成uImage文件，花的时间可能会比较长，大概十几分钟。

====> STEP 6: 运行

将MLO、u-boot.img、uImage、am335x-evm.dtb拷贝到SD的boot分区（假设SD卡已经使用TI公司提供的分区工具create-sdcard.sh分成了boot区和rootfs区），使用kermit连上开发板的com0，设置串口参数，等待串口出现U-Boot提示符：

U-Boot# fatls mmc 0

U-Boot# fatload mmc 0 0x82000000 uImage

U-Boot# fatload mmc 0 0x83000000 am335-evm.dtb

U-Boot# bootm 0x82000000 - 0x83000000

此时，就可以看到欢快的内核打印信息了～～～

哦耶～～～

## Booting kernel from Legacy Image at 82000000 ...

Image Name: Linux-3.16.1

Image Type: ARM Linux Kernel Image (uncompressed)

Data Size: 4172416 Bytes = 4 MiB

Load Address: 80008000

Entry Point: 80008000

Verifying Checksum ... OK

## Flattened Device Tree blob at 83000000

Booting using the fdt blob at 0x83000000

Loading Kernel Image ... OK

Using Device Tree in place at 83000000, end 8300b6e7

Starting kernel ...

[ 0.000000] Booting Linux on physical CPU 0x0

[ 0.000000] Linux version 3.16.1 (maria@localhost.localdomain) (gcc version 4.7.3 20130226 (prerelease) (crosstool-NG linaro-1.13.1-4.7-2013.03-20130313 - Linaro GCC 2013.03) ) #15 SMP Tue Sep 2 14:33:33 CST 2014

出现了内核打印信息，说明Linux Kernel就开始跑起来了，当然剩下的工作仍然很多，比如制作文件系统、解决各种各样打印问题、解决各种各样的驱动问题……但有了打印信息，就不会是无头无脑找问题的阶段了（当然，JTAG也是个好工具），接下来的任务，就是修改mainline中关于am335x-evm.dtb的内容，直到Forlinx开发板的硬件运行正确，再制作成自己的forlinx.dtb。

万里长征，go on walking......

（补充：Forlinx的OK335D开发板，使用arch/arm/boot/dtc/am335x-bone.dts，而不是am335x-evm.dts，能够顺利出现串口的命令提示符，就是说Kernel和file system都加载正确了！到目前为止，mainline的porting工作告一段落，之后的内容属于针对项目的研发，因此这个U-Boot for AM335x系列也结束了，wish me luck！^\_^）