为 SAM926X 编译 U-boot

Team MCUZone
http://www.mcuzone.com

版本：Rev1.0

www.mcuzone.com

2007-08

更新记录

2008.08.26

文档创建。

为 SAM926X 编译 U-boot

—基于 Virtual PC

Team MCUZone

本文叙述 U-boot 的编译过程，包括编译器的安装，U-boot 编译及使用。

一， 准备工作

1. 建立开发环境

请先按照《基于 VPC 建立 ARM_Linux 开发环境.doc》一文建立好开发环境，使得 linux 上的工作文件夹作为一个网络驱动器，挂接到 XP 主机上。

2. 下载相关文件

请到本站 ftp(ftp://www.mcuzone.com，用户名：mcuzone 密码:mcuzone)

下载下列文件：

AT91Bootstrap1.2.zip	241 KB	WinRAR ZIP 档案...	2007-8-26 14:35
arm-linux-cross-2.95.3.tar.bz2	35,424 KB	WinRAR 档案文件	2007-6-20 11:51
arm-linux-gcc-3.4.1.tar.bz2	41,744 KB	WinRAR 档案文件	2007-6-13 15:42
arm-softfloat-linux-gnu.tar.gz	89,735 KB	WinRAR 档案文件	2007-6-13 16:47
gcc-3.3.6-glib-2.3.6.tar.bz2	79,272 KB	WinRAR 档案文件	2007-6-13 16:09
u-boot-1.1.5-atmel.tar.bz2	6,837 KB	WinRAR 档案文件	2007-3-8 11:15

在 Linux 虚拟机上以 root 登录。

在 XP 上那个网络驱动器上新建文件夹 source，将下载的文件都放到该文件夹。

如果 source 文件夹在 Linux 上创建，那么需要修改权限，使得 XP 用户可以修改。

```
[root@RH9 work]# chown -R nobody:nobody ./source/
```

在 Linux 下即可看到这些文件：

```
[root@RH9 source]# pwd
/usr/work/source
[root@RH9 source]# ls
arm-linux-cross-2.95.3.tar.bz2  arm-softfloat-linux-gnu.tar.gz  u-boot-1.1.5-atmel.tar.bz2
arm-linux-gcc-3.4.1.tar.bz2     gcc-3.3.6-glib-2.3.6.tar.bz2
[root@RH9 source]#
```

二， 准备编译

1. 展开 U-boot 源码包

到 work 文件夹下，展开 U-boot 源码包：

```
[root@RH9 work]# tar -jxvf ./source/u-boot-1.1.5-atmel.tar.bz2
```

完成后多出一个相应的文件夹：

```
[root@RH9 work]# ls
source  u-boot-1.1.5
```

修改文件夹权限，使得在 XP 端可以编辑源代码：

```
[root@RH9 work]# chown -R nobody:nobody ./u-boot-1.1.5/
```

检查一下展开出来的代码，可以看到 ATMEL 修改后的所有 board 文件夹：

```
[root@RH9 work]# ls ./u-boot-1.1.5/board/atmel/
at91rm9200df  at91rm9200dk  at91rm9200ek  at91sam9260ek  at91sam9261ek  at91sam9263ek  atstk1000
[root@RH9 work]#
```

2. 展开并编译 bootstrap

根据 SAM9261 的 boot 顺序以及 boot 的限制，必须先用一个最初的 boot 代码完成处理器的初始化及 SDRAM 的初始化，再将 U-boot 代码复制到 SDRAM 中运行。

对于 bootstrap 的代码，其大小有严格的限制，就是不能超过片内 RAM 区域的大小，二是要按照 SAM-BA 所要求的格式编写。

ATMEL 已经提供了现成的 bootstrap 代码，就是 AT91Bootstrap1.2.zip，可以阅读文档 doc6277.pdf 来了解其工作过程。这里仅以 SAM9261(U-boot 置于 dataflash)为例来说明如何编译。

注意：如无必要，就不需要自行编译 bootstrap。

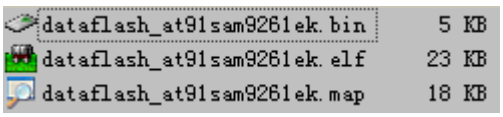
该 bootstrap 代码使用 arm-elf 工具链编译，因此需要在 PC 上安装该工具链，一个简单的方法就是安装 WinARM。

安装了 WinARM 后开启一个 cmd 窗口，使用下列命令检查安装结果：

```
C:\>arm-elf-gcc -v
Using built-in specs.
Target: arm-elf
Configured with: ../gcc-4.1.2/configure --target=arm-elf --prefix=/c/WinARM --disable-nls --disable-shared --disable-thr
eads --with-gcc --with-gnu-ld --with-gnu-as --enable-languages=c,c++ --enable-interwork --enable-multilib --with-newlib
--disable-libssp --disable-libstdcxx-pch --disable-libmudflap --disable-win32-registry --with-cpu=arm7tdmi --with-newlib
 --program-prefix=arm-elf- -v
Thread model: single
gcc version 4.1.2 (WinARM 4/2007)
```

说明安装成功，环境变量也以添加。

将 AT91Bootstrap1.2.zip 在 PC 机上展开，到 board 目录下面可以看到对应的开发板名字，进入 at91sam9261ek 文件夹，可以看到有两个配置，就是 dataflash 与 nandflash。现在先编译 dataflash 的版本，进入 dataflash 文件夹，可以看到已经编译成功的输出：



这些是 ATMEL 预先编译好的，可以直接使用。为了编译自己的代码，先将其保存到另外的文件夹。

然后在该文件夹下新建一个文本文件，文件内容为 cmd，保存后将文件改名为 bat 文件，比如 start.bat。

双击该 bat 文件，将运行一个 cmd 窗口，在窗口中输入 make 回车即可开始编译，很快就会完成，新的目标文件将生成。

现在要看一下当前文件夹下的 at91sam9261ek.h，注意下面的参数：

```
/* ********************************************************************* */
/* BootStrap Settings                                                    */
/*                                                                       */
/* ********************************************************************* */
#define AT91C_SPI_PCS_DATAFLASH   AT91C_SPI_PCS0_DATAFLASH  /* Boot on SPI NCS0 */

#define IMG_ADDRESS     0x8000      /* Image Address in DataFlash */
#define IMG_SIZE    0x32000     /* Image Size in DataFlash    */

#define MACH_TYPE       0x350       /* AT91SAM9261-EK */
#define JUMP_ADDR   0x23F00000    /* Final Jump Address         */
```

bootstrap 从 IMG_ADDRESS 开始读取代码，长度为 IMG_SIZE，并将其复制到 SDRAM 的 JUMP_ADDR 处。这些参数在 U-boot 编译和烧写的时候都需要保持一致。

3. 安装编译器

可以使用 arm-softfloat-linux-gnu-的工具链来编译 U-boot,因此需要安装该工具链。此工具链可以自行编译，也可以使用现成。自行编译可以使用 crosstool，其使用方法可以到 crosstool 的网站上看 howto。

本文以从网上编译好的工具链为例说明如何安装和使用。

由于目前还不清楚该工具链应该被安装到什么地方，因此可以在本地先将其展开：

```
[root@RH9 work]# tar -xjvf ./source/arm-softfloat-linux-gnu.tar.gz
```

完成后到新建立的文件夹下检查：

```
[root@RH9 work]# /usr/work/arm-softfloat-linux-gnu/bin/arm-softfloat-linux-gnu-gcc -v
Reading specs from /usr/work/arm-softfloat-linux-gnu/bin/../lib/gcc/arm-softfloat-linux-gnu/3.4.1
/specs
Configured with: /opt/crosstool-0.42/build/arm-softfloat-linux-gnu/gcc-3.4.1-glibc-2.3.3/gcc-3.4.
1/configure --target=arm-softfloat-linux-gnu --host=i686-host_pc-linux-gnu --prefix=/opt/crosstoo
l/gcc-3.4.1-glibc-2.3.3/arm-softfloat-linux-gnu --with-float=soft --with-headers=/opt/crosstool/g
cc-3.4.1-glibc-2.3.3/arm-softfloat-linux-gnu/arm-softfloat-linux-gnu/include --with-local-prefix=
/opt/crosstool/gcc-3.4.1-glibc-2.3.3/arm-softfloat-linux-gnu/arm-softfloat-linux-gnu --disable-nl
s --enable-threads=posix --enable-symvers=gnu --enable-__cxa_atexit --enable-languages=c,c++ --en
able-shared --enable-c99 --enable-long-long
Thread model: posix
gcc version 3.4.1
[root@RH9 work]#
```

可以看到该工具链正是使用 crosstool 编译产生。

--target 指明了目标的体系结构

--prefix 指明了安装路径

--with-headers 指明了头文件所在路径

按照上面的信息建立相关的文件夹：

```
[root@RH9 arm-softfloat-linux-gnu]# mkdir /opt/crosstool
[root@RH9 arm-softfloat-linux-gnu]# mkdir /opt/crosstool/gcc-3.4.1-glibc-2.3.3/
```

到新建的目录下重新展开编译器：

```
[root@RH9 work]# cd /opt/crosstool/gcc-3.4.1-glibc-2.3.3/
[root@RH9 gcc-3.4.1-glibc-2.3.3]# ls
[root@RH9 gcc-3.4.1-glibc-2.3.3]# tar -xjvf /usr/work/source/arm-softfloat-linux-gnu.tar.gz
```

完成后即可：

```
[root@RH9 gcc-3.4.1-glibc-2.3.3]# ls
arm-softfloat-linux-gnu
[root@RH9 gcc-3.4.1-glibc-2.3.3]#
```

原来 work 文件夹下的可以删除：

```
[root@RH9 work]# rm -Rf arm-softfloat-linux-gnu/
[root@RH9 work]# ls
source  u-boot-1.1.5
[root@RH9 work]#
```

4. 添加环境变量

为了使用方便，需要将工具链添加到用户的环境变量。可以使用如下方式：

运行：

```
[root@RH9 work]# vi ~/.bashrc
```

在文件末尾添加：

```
# Export Path
export PATH=$PATH:/opt/crosstool/gcc-3.4.1-glibc-2.3.3/arm-softfloat-linux-gnu/bin
```

保存后运行 source 使得新的设置对当前用户生效：

```
[root@RH9 work]# source ~/.bashrc
```

检查一下：

```
[root@RH9 work]# echo $PATH
/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
:/usr/X11R6/bin:/root/bin:/opt/crosstool/gcc-3.4.1-glibc-2.3.3/arm-softfloat-linux-gnu/bin
```

环境变量设置成功，现在可以直接运行下：

```
[root@RH9 work]# arm-softfloat-linux-gnu-gcc -v
Reading specs from /opt/crosstool/gcc-3.4.1-glibc-2.3.3/arm-softfloat-linux-gnu/lib/gcc/arm-softf
loat-linux-gnu/3.4.1/specs
Configured with: /opt/crosstool-0.42/build/arm-softfloat-linux-gnu/gcc-3.4.1-glibc-2.3.3/gcc-3.4.
1/configure --target=arm-softfloat-linux-gnu --host=i686-host_pc-linux-gnu --prefix=/opt/crosstoo
l/gcc-3.4.1-glibc-2.3.3/arm-softfloat-linux-gnu --with-float=soft --with-headers=/opt/crosstool/g
cc-3.4.1-glibc-2.3.3/arm-softfloat-linux-gnu/arm-softfloat-linux-gnu/include --with-local-prefix=
/opt/crosstool/gcc-3.4.1-glibc-2.3.3/arm-softfloat-linux-gnu/arm-softfloat-linux-gnu --disable-nl
s --enable-threads=posix --enable-symvers=gnu --enable-__cxa_atexit --enable-languages=c,c++ --en
able-shared --enable-c99 --enable-long-long
Thread model: posix
gcc version 3.4.1
```

OK。

三， 编译 U-boot

1. 配置参数

进入 U-boot 源码所在文件夹：

```
[root@RH9 work]# ls
source  u-boot-1.1.5
[root@RH9 work]# cd u-boot-1.1.5/
```

检查 Makefile 中工具链的配置:

```
[root@RH9 u-boot-1.1.5]# vi Makefile
```

默认的情况下 arm 的工具链已经被 ATMEL 设置为 arm-softfloat-linux-gnu-

```
128 ifeq ($(ARCH),arm)
129 CROSS_COMPILE = arm-softfloat-linux-gnu-
130 endif
```

如果不是需要修改为这个设置。

下面需要检查 U-boot 的 link 地址，以使其与 bootstrap 一致。

```
[root@RH9 work]# cat ./u-boot-1.1.5/board/atmel/at91sam9261ek/config.mk
TEXT_BASE = 0x23f00000
```

以上输出说明 link 起始地址在 0x23F00000，与 bootstrap 一致。如果不一致，则需要这

两者之一。

注意：修改代码的时候请保持文件的 UNIX 格式。

2．编译

下面以 9261 板子为例来说明如何编译。

运行：

```
[root@RH9 u-boot-1.1.5]# make at91sam9261ek_config
Configuring for at91sam9261ek board...
[root@RH9 u-boot-1.1.5]#
```

完成对 9261ek 板子的设置，注意_config 之前的 board 名字必须与 board 文件夹下的一致。

配置完成后即可进行编译：

```
[root@RH9 u-boot-1.1.5]# make
```

几分钟之后编译即会完成：

```
arm-softfloat-linux-gnu-objcopy --gap-fill=0xff -O srec u-boot u-boot.srec
arm-softfloat-linux-gnu-objcopy --gap-fill=0xff -O binary u-boot u-boot.bin
gzip -c u-boot.bin > u-boot-at91sam9261ek.gz
cp u-boot-at91sam9261ek.gz /tftpboot/u-boot-at91sam9261ek-2007-08-26.gz
cp u-boot.bin /tftpboot/u-boot-at91sam9261ek-2007-08-26.bin
```

主要生成下列文件：

```
-rwxr-xr-x    1 root     root       531846 Aug 26 12:54 u-boot
-rwxr-xr-x    1 root     root       191728 Aug 26 12:54 u-boot.bin
-rw-r--r--    1 root     root       109033 Aug 26 12:54 u-boot.map
```

其中 u-boot 是一个 elf 文件，可以用于调试：

```
[root@RH9 u-boot-1.1.5]# file u-boot
u-boot: ELF 32-bit LSB executable, ARM, version 1 (ARM), statically linked, not stripped
```

u-boot.bin 是 binary 格式的输出文件，可以直接到板子上运行。

u-boot.ma 是 map 文件，提供了目标文件的一些信息。

同时，生产的 bin 文件会被复制到/tftpboot 下以方便使用 tftp 下载测试。

四， 使用 U-boot

先在 PC 上安装 SAM-BA，这样可以使用其进行下载。

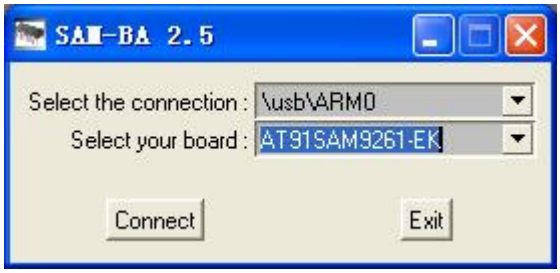
现以核心板的 JTAG 插座在左侧来描述：

找到核心板左上角的 J4，将 J4 右边的那个开关拨到上方，使得板子从内部 ROM 启动；

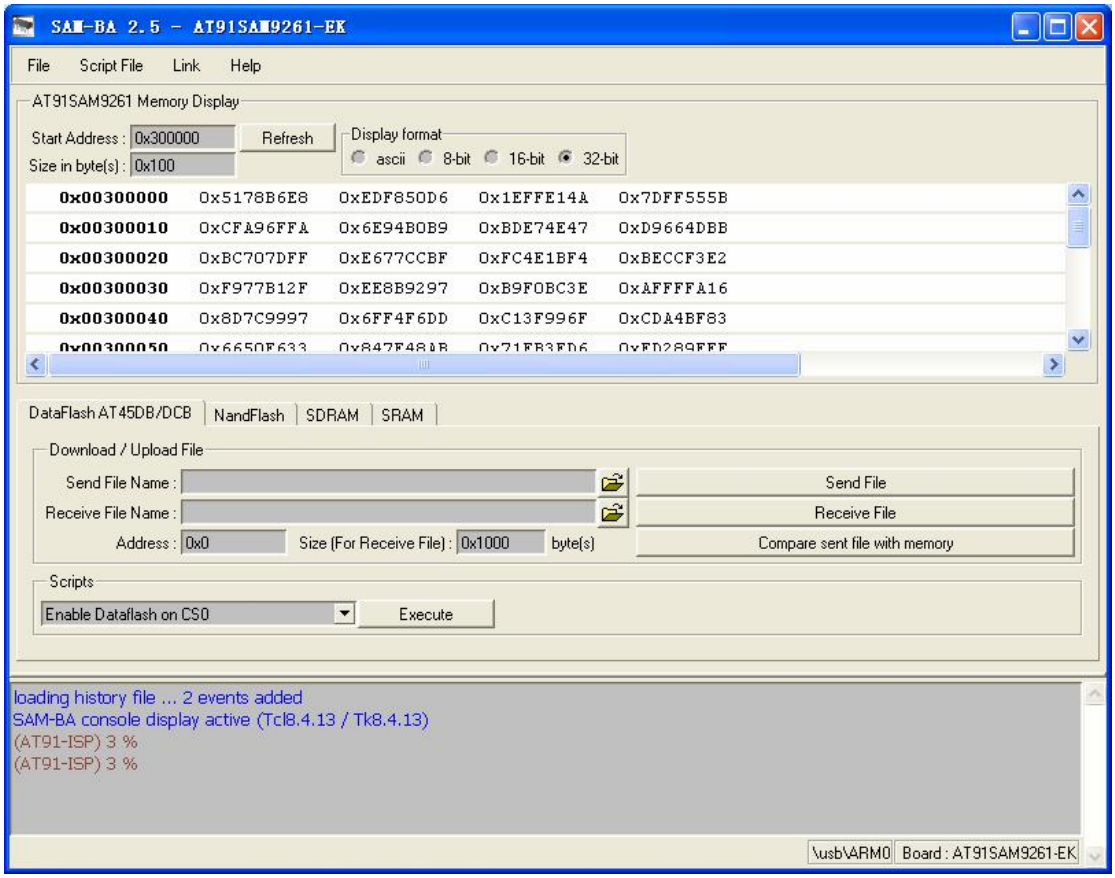
找到核心板右下角的 J21，将其下面的那个开关拨到右方，断开 dataflash 的 CS。

使用 mini USB 线连接 PC 与 9261 开发板，PC 端将出现一个新的 USB 设备，如果安装驱动，一路 next 即可。

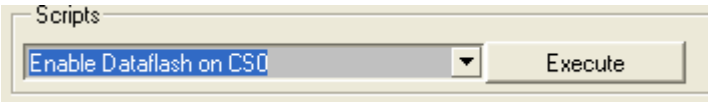
运行 SAM-BA,等待一段时间，在 SAM-BA 窗口中如下选择：



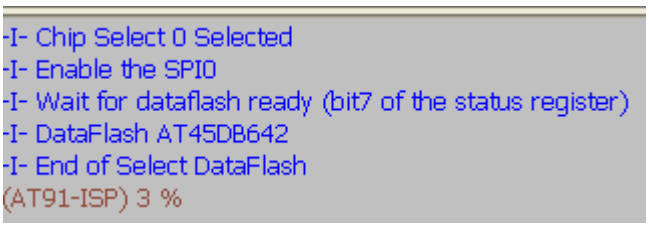
选择 connect，即会出现主界面：



此时再将 J21 下方的开关拨到左侧，运行下面的脚本：

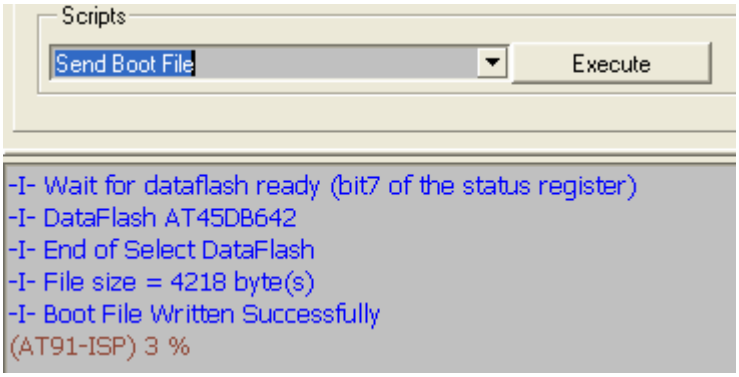


下面是正确的输出信息：

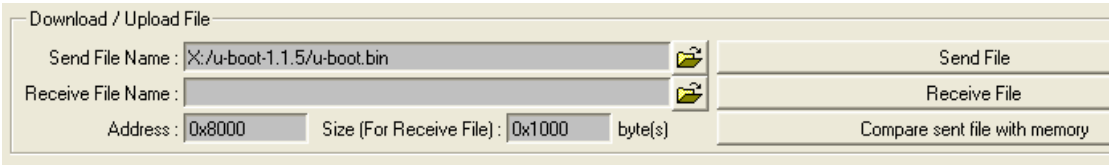


AT45DB642 已被正确探测到。

下面选择发送启动文件，在弹出的对话框中选择 ATMEL 提供的基于 dataflash 的 bootstrap，点击 execute 即可烧写 boot 文件。脚本会在保留位置填写对应的文件长度等信息。



再选择刚才编译好的 u-boot.bin，地址要选择 0x8000：



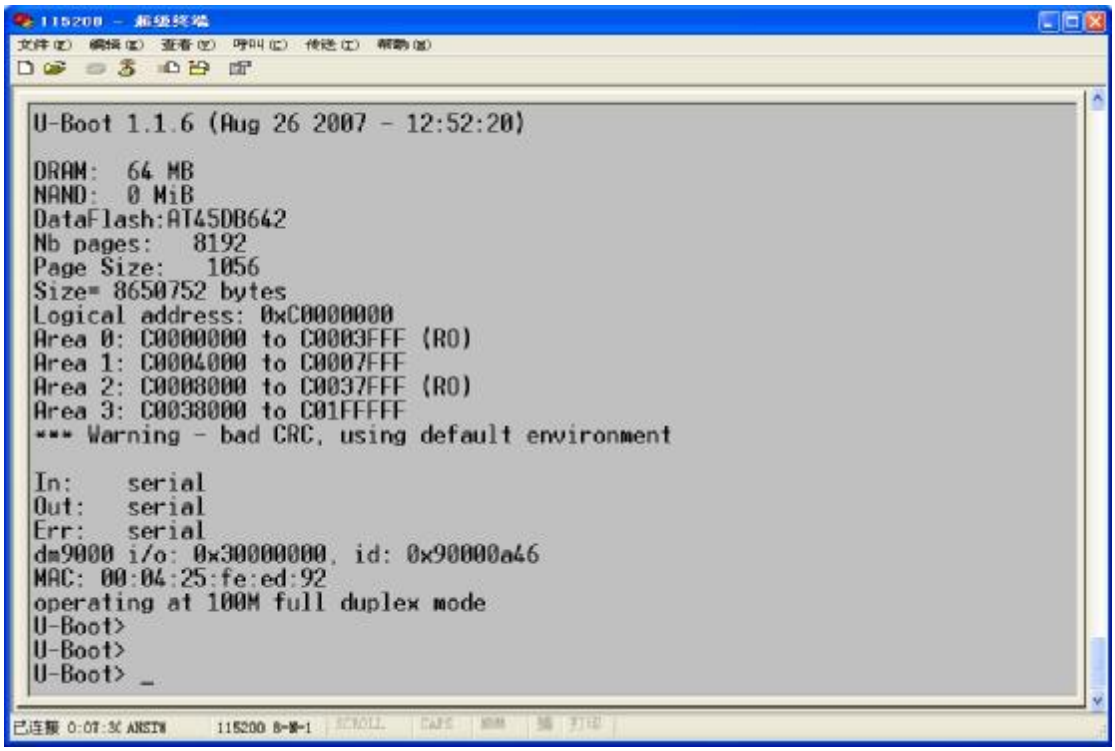
点击 Send File，文件就将被下载到 dataflash 上 0x8000 地址开始的地方。

```
(AT91-ISP) 3 % send_file {DataFlash AT45DB/DCB} "X:/u-boot-1.1.5/u-boot.bin" 0x8000 0
-I- Send File X:/u-boot-1.1.5/u-boot.bin at address 0x8000
-I- File size = 191728 byte(s)
```

完成后可以关闭 SAM-BA，用 USB 线连接底板的 J4 与 PC，安装好 USB 转串口的驱动程序，PC 上多一个串口，在设备管理器中记录下串口号。

打开一个超级中断，使用上面那个串口，设置为 115200,n,8,1。

拔掉 mini USB 线，然后再联接上，中断窗口里可以看到 U-boot 的启动画面：



之所以出现 bad CRC，原因在于没有在存储区域找到环境变量。

五， 后记

以上的叙述中并没有补充 U-boot 相关知识，有问题的时候可以看 U-boot 官网或者 google。

如果有任何建议和问题，请到 MCUZone 论坛发帖，谢谢！

[TODO]调试 U-boot

www.mcuzone.com
提供：
全系列 ARM 开发工具:仿真器，开发软件
ARM7,ARM9 开发学习板
AVR 开发工具
MSP430 开发工具
ATMEL 工业级 ARM 芯片