

## 第 2 章 开发 EBD9200 起点

### 内容:

通过上一章，你可能已经了解了 EBD9200 的概况，整体介绍如何开发 EBD9200。通过这一章，你应该对 EBD9200 的开发有个具体的了解。

当你拿到 EBD9200 的板子，看着在你面前的一大堆开发资料，如果你已经开发过类似的系统，这一章，你大可不必去看，去到文档中找你关注的内容，如果你没有接触过 ARM，可能你看到板子时，你会不知所措，不要紧，这也是正常现象，这一章会让你消除你的恐惧心理，坦然地面对你的产品开发。

这一章我们不讲实质性的内容，而是让你对板子加电，然后去熟悉他，知道它的开发步骤。

- 对板子加电，按照规程运行已经烧写的程序，操作步骤

### 1. 一步步开始，为 EBD9200 加电

当你拿到 EBD9200 板子时，英贝德已经全部为你测试好，同时预先在 Flash 中为你烧写好了 Bootloader, Linux 内核, Linux 文件系统；对于你的空板子，请看本章第三章的内容。

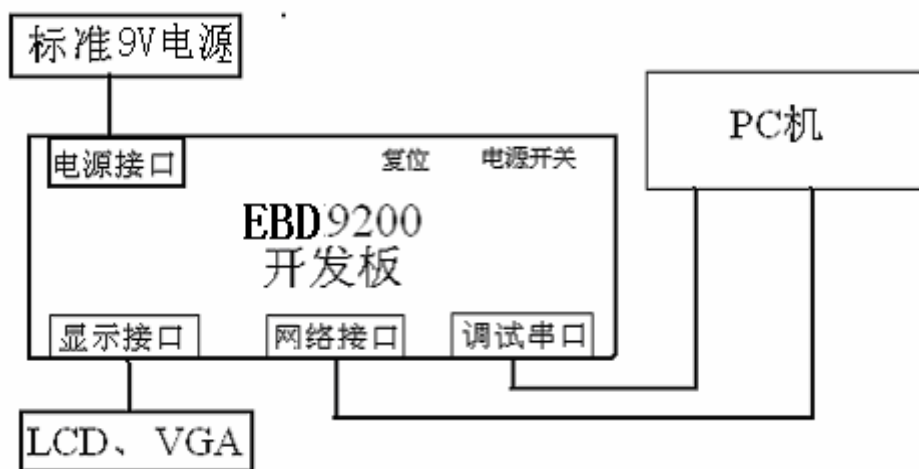
#### STEP 1:

在熟悉 EBD9200 的基础上，进行各种跳线设置（参照第一章的跳线设置表格），主要检查以下跳线（出厂时的设置）：

J2(核心板上): 短接 2、3 脚;	J11: 短接;
J13: 短接;	J16: 短接;
J33: 接上端;	J34: 断开;
J35: 短接;	
JP1 接右端	J23: 短接 1、2 脚;

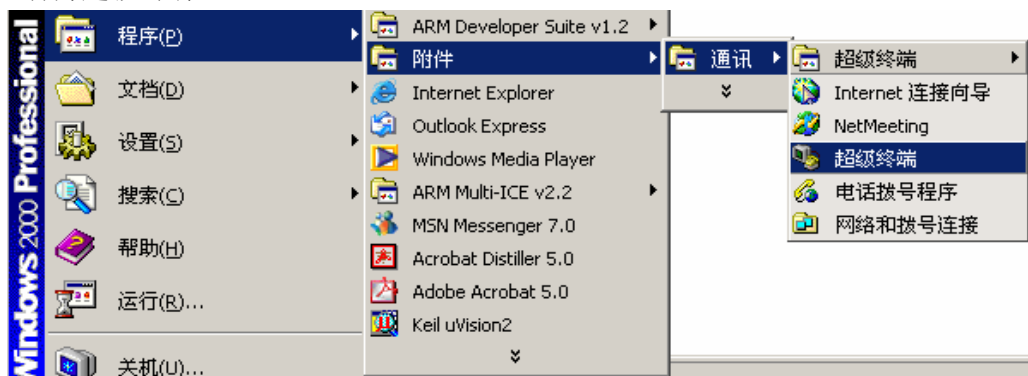
#### STEP 2:

按照下图，进行 EBD9200 开发系统的连接。开发板接 9V 电源。



**STEP 3:**

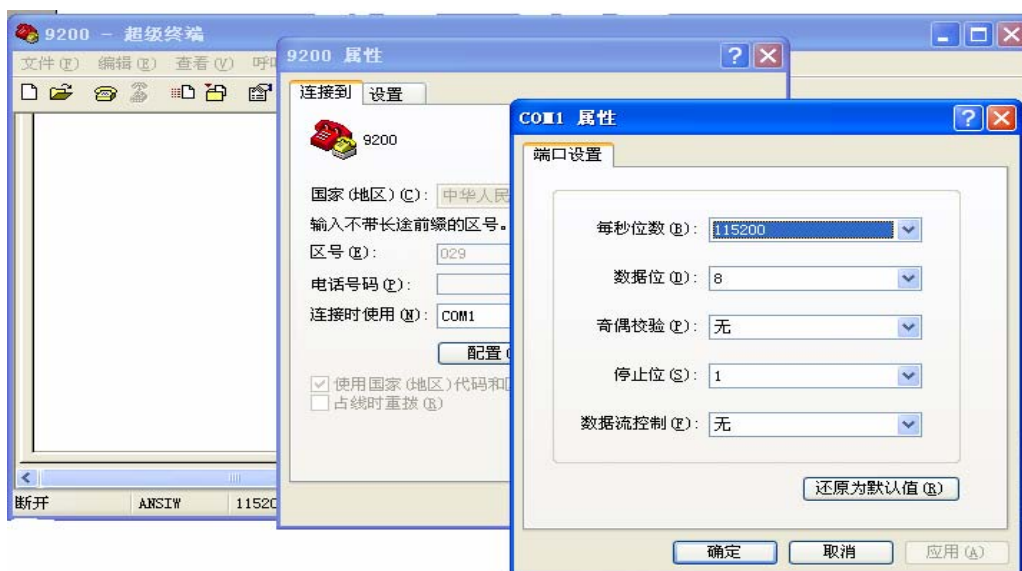
打开超级终端



在 PC 宿主机的设置:

设置串口为 115200, 8, 无, 1, 无, 让超级终端处于接收状态;

连接 VGA 或 LCD(如果是 EBD9200-I 型板)等(连法在后面有说明)

**STEP 4: 加电和复位**

加电或者复位后, 超级终端出现的打印信息如下:

```
boot 1.0 (Oct 16 2004 - 22:21:32)
Uncompressing image...

U-Boot 1.1.1 (Nov 16 2004 - 18:01:43)
U-Boot code: 21F00000 -> 21F16F2C BSS: -> 21F1B368
RAM Configuration:
Bank #0: 20000000 32 MB
Flash: 16 MB
In: serial
Out: serial
Err: serial
Hit any key to stop autoboot: 0
```

稍等片刻（系统加载中），接着出现信息如下：（内核解压、启动）

```
in: serial
Out: serial
Err: serial
Hit any key to stop autoboot: 0
## Booting image at 21000000 ...
Image Name:
Image Type: ARM Linux Kernel Image (gzip compressed)
Data Size: 848204 Bytes = 828.3 kB
Load Address: 20008000
Entry Point: 20008000
Verifying Checksum ... OK
Uncompressing Kernel Image ... _
```

连接的 0:00:47 ANSIW 115200 8-N-1 SCROLL CAPS NUM 捕 打印

启动过程显示：

```

Image Type: ARM Linux Kernel Image (gzip compressed)
Data Size: 779502 Bytes = 761.2 kB
Load Address: 20008000
Entry Point: 20008000
Verifying Checksum ... OK
Uncompressing Kernel Image ... OK

Starting kernel ...

Linux version 2.4.27-vrs1 (susu@localhost.localdomain) (gcc version 2.95.3 20010
315 (release)) #449 Wed Mar 29 10:30:13 CST 2006
CPU: Arm920Tid(wb) revision 0
Machine: ATMEL AT91RM9200
On node 0 totalpages: 8192
zone(0): 8192 pages.
zone(1): 0 pages.
zone(2): 0 pages.
Kernel command line: mem=32M console=ttyS0,115200 root=/dev/ram rw initrd=0x2110
0000,8000000 ramdisk_size=20000
at91xx_get_rtc_time
READ:1307
Console: colour dummy device 80x30
Calibrating delay loop... 89.70 BogoMIPS
Memory: 32MB = 32MB total
Memory: 22680KB available (1471K code, 324K data, 80K init)
Dentry cache hash table entries: 4096 (order: 3, 32768 bytes)
Inode cache hash table entries: 2048 (order: 2, 16384 bytes)
Mount cache hash table entries: 512 (order: 0, 4096 bytes)
Buffer cache hash table entries: 1024 (order: 0, 4096 bytes)
Page-cache hash table entries: 8192 (order: 3, 32768 bytes)
CPU: Testing write buffer: pass
POSIX conformance testing by UNIFIX
Linux NET4.0 for Linux 2.4
Based upon Swansea University Computer Society NET3.039
Initializing RT netlink socket
Starting kswapd
devfs: v1.12c (20020818) Richard Gooch (rgooch@atnf.csiro.au)
devfs: boot_options: 0x1
JFFS2 version 2.1. (C) 2001 Red Hat, Inc., designed by Axis Communications AB.
S1D13XXX: Phys address:C2880000 Phys Reg address:C287E000
Warning: Remapping obsolete /dev/fb* minor 255 to 7
fb0: s1d13xxx frame buffer device
Console: switching to colour frame buffer device 80x30
fb0: s1d13xxx frame buffer device 显示驱动
display 640 x 480 16Bpp
Starting USB Keyboard Driver... Done
RAMDISK driver initialized: 16 RAM disks of 20000K size 1024 blocksize
Uniform Multi-Platform E-IDE driver Revision: 7.00beta4-2.4
ide: Assuming 50MHz system bus speed for PIO modes; override with idebus=xx
EBD9200 IDE initialization - driver version 1.0, 2-18-06.
hda: Hitachi XX.V.3.4.0.0, CFA DISK drive
ide0 at 0xc2a81040-0xc2a81047,0xc2a81038 on irq 28
hda: attached ide-disk driver.
hda: 250368 sectors (128 MB) w/1KiB Cache, CHS=978/8/32
Partition check: CF/IDE驱动
/dev/ide/host0/bus0/target0/lun0: p1
SCSI subsystem driver Revision: 1.00
kmod: failed to exec /sbin/modprobe -s -k scsi_hostadapter, errno = 2

```



```

kmod: failed to exec /sbin/modprobe -s -k scsi_hostadapter, errno = 2
physmap flash device: 1000000 at 10000000
cfi_cmdset_0001: Erase suspend on write enabled
Using buffer write method
using static partition definition
Creating 1 MTD partitions on 'Physically mapped flash(E28F128J3H)' : JFFS2文件系统
0x00100000-0x00fe0000 : "Nor partition,jffs2"
usb.c: registered new driver hub
usb.c: registered new driver usbmouse
usbmouse.c: v1.6:USB HID Boot Protocol mouse driver
usb.c: registered new driver usbkbd
usbkbd.c: :USB HID Boot Protocol keyboard driver
initializing USB Mass Storage driver...
usb.c: registered new driver usb-storage
USB Mass Storage support registered.
mice: PS/2 mouse device common for all mice
ttyS%d0 at MMIO 0xfeff200 (irq = 1) is a AT91_SERIAL
ttyS%d1 at MMIO 0xfefc4000 (irq = 7) is a AT91_SERIAL
eth0: Link now 100-FullDuplex
eth0: AT91 ethernet at 0xfefbc000 int=24 100-FullDuplex (12:34:56:78:9a:bc) 网络驱动
eth0: Davicom 9196 PHY (Copper)
AT91 Watchdog timer enabled (5 seconds)
AT91 Real Time Clock driver
host/usb-ohci.c: USB OHCI at membase 0xc3a8c000, IRQ 23
usb.c: new USB bus registered, assigned bus number 1
hub.c: USB hub found
USB HOST驱动

hub.c: 2 ports detected
AT91 SPI driver loaded
at91_dataflash: Atmel AT45DB021B detected (spi0) (270600 bytes)
Creating 1 MTD partitions on "Atmel AT45DB021B":
0x00000000-0x00042000 : "spi dataflash partition 1, jffs2"
DATAPLASH_JFFS2驱动
SmartMedia card inserted.
NAND device: Manufacturer ID: 0xec, Chip ID: 0x76 (Samsung NAND 64MiB 3,3V)
Creating 2 MTD partitions on "NAND 64MiB 3,3V":
0x00000000-0x02000000 : "NAND partition 1, root-yaffs"
0x02000000-0x04000000 : "NAND partition 2, use-yaffs"
NAND FLASH_YAFFS驱动
CAN-BUS driver loaded
NET4: Linux TCP/IP 1.0 for NET4.0
CAN驱动
IP Protocols: ICMP, UDP, TCP
IP: routing cache hash table of 512 buckets, 4Kbytes
TCP: Hash tables configured (established 2048 bind 2048)
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
NetWinder Floating Point Emulator V0.97 (double precision)
RAMDISK: Couldn't find valid RAM disk image starting at 0.
Freeing initrd memory: 7812K

最后出现登陆信息

INI1: version 2./4 booting
mount: mount point /dev/pts does not exist
INIT: Entering runlevel: 3
Starting system logger: syslogd
Starting INET services: inetd
Starting mouse
Starting tty
Starting console mouse services: gpm
Starting rtc gpm: oops() invoked from gpm.c(952)
/dev/mouse: No such file or directory

AT91RM9200DK login: root

```

以 root 用户登录 Linux 系统，即可以敲入命令操作了，比如 cd、ls 等。

```
Starting rtc gpm: oops() invoked from gpm.c(952)
/dev/mouse: No such file or directory

AT91RM9200DK login: root
[root@AT91RM9200DK /root]$cd /
[root@AT91RM9200DK /]$ls
var          usr          tmp          soft          sbin          root
rd           proc          mnt          lost+found    lib           home
etc          dev          bin
[root@AT91RM9200DK /]$
```

连接的 0:08:43 ANSIW 115200 8-N-1 SCROLL CAPS NUM 捕 打印

当然，我们已经在文件系统里面加入一些测试程序，敲入 `cd /soft`、`ls` 即可列出来。

```
[root@AT91RM9200DK /]$ls
var          usr          tmp          soft          sbin          root
rd           proc          mnt          lost+found    lib           home
etc          dev          bin
[root@AT91RM9200DK /]$cd /soft
[root@AT91RM9200DK /soft]$ls
hello        hello1        start-hello   ds1307        cantest
[root@AT91RM9200DK /soft]$_
```

连接的 0:03:14 ANSIW 115200 8-N-1 SCROLL CAPS NUM 捕 打印

可以分别运行试试，`start-hello` 只是打印出字符信息，运行如下：

```
[root@AT91RM9200DK /soft]$ls
hello        hello1        start-hello   ds1307        cantest
[root@AT91RM9200DK /soft]$./start-hello

hello!!!!

welcome to www.szembed.com
[root@AT91RM9200DK /soft]$
```

连接的 0:07:47 ANSIW 115200 8-N-1 SCROLL CAPS NUM 捕 打印

`ds1307` 是一个板上外扩的实时时钟测试程序(内核的 `i2c` 驱动已做好)，运行如下：

```
[root@AT91RM9200DK /soft]$./start-hello

hello!!!!

welcome to www.szembed.com
[root@AT91RM9200DK /soft]$./ds1307
i2c-0 open
Start test DS1307
Do you want to Reset Time(Y/N)
```

连接的 0:19:57 ANSIW 115200 8-N-1 SCROLL CAPS NUM 捕 打印

提示是否重新设定时间，默认在键盘上输入 n、接着回车即可。

```
[root@HI91RM9200DK /soft]$ ./ds1307
i2c-0 open
Start test DS1307
Do you want to Reset Time(Y/N)n
Now is =      ' 12-Oct-2005  14:28:14   Wednesday '
Now is =      ' 12-Oct-2005  14:28:15   Wednesday '
Now is =      ' 12-Oct-2005  14:28:16   Wednesday '
Now is =      ' 12-Oct-2005  14:28:17   Wednesday '
Now is =      ' 12-Oct-2005  14:28:18   Wednesday '
Now is =      ' 12-Oct-2005  14:28:19   Wednesday '
Now is =      ' 12-Oct-2005  14:28:20   Wednesday '
_
```

连接的 0:24:11 ANSIW 115200 8-N-1 SCROLL CAPS NUM 捕 打印

要暂停运行，键如 Ctrl+C 即可退出测试。

如果你的板子是 EBD9200-I 型的话，hello、hello1、cantest 这三个程序还可以运行，如果你购买的是 EBD9200-II 型，这三个程序就对你没有用了。

cantest 是 CAN 现场总线测试的，你只需要用两根导线把底版右上角的 J1 和 J2 绿色端子平行连接起来，接着运行 ../cantest，板上的两路 CAN 即可相互通信了。

```
[root@AT91RM9200DK /soft]$ ls
hello      hello1      start-hello  ds1307      cantest
[root@AT91RM9200DK /soft]$ ./cantest
CAN-Bus Test Start!
CAN 0 open
CAN 1 open
can_interrupt 1 ffff8280

can_interrupt 0 ffff4280
CAN1: receive 8 data=' 10 11 12 13 14 15 16 17' ID=1026
can_interrupt 1 ffff8280
CAN0: receive 8 data=' 200 198 196 194 192 190 188 186' ID=1026
can_interrupt 0 ffff4280
CAN1: receive 8 data=' 0 2 4 6 8 10 12 14' ID=1026
can_interrupt 1 ffff8280
CAN0: receive 8 data=' 1 3 5 7 9 11 13 15' ID=1026
_
```

连接的 0:46:33 ANSIW 115200 8-N-1 SCROLL CAPS NUM 捕 打印

要暂停运行，键入 Ctrl+C 即可退出测试。

hello、hello1 分别是 LCD、CRT 的显示测试程序，在我们移植的文件系统里移植好了 miniGUI 的库文件，运行他们的时候在显示终端出现一个图形的界面，有图片，窗口，文字等。

如果你购买的只是 EBD9200-I 型板(不带 LCD 屏)，我们默认为你烧写的内核是带 CRT 驱动的，你只需要把 CRT 显示器的 D-型接头接到开发板的接头 J9 即可，运行 hello 便在你的显示器出现 GUI 界面。

```

[root@HI91RM9200DK /soft]$ls
hello          hello1          start-hello    ds1307          cantest
[root@AT91RM9200DK /soft]$./hello
IAL: Does not find matched engine.
IAL: Use the first engine: Dummy
_

```

连接的 1:16:12 ANSIW 115200 8-N-1 SCROLL CAPS NUM 捕 打印

要暂停运行，键入 Ctrl+C 即可退出。

如果你购买的只是 EBD9200-I 型板和我们提供的 LCD（640x480 或 320x240），我们默认你烧写的内核是带 LCD 驱动的，你需要把 LCD 板上的 40 针排线接到开发板底版的 40 针排线接座 J11(注意，不是旁边的 J7)，并且用 5V 的直流电源接到 LCD 板，运行 hello 便在你的屏出现 GUI 界面。

**（警告，通电情况下不允许拔插任何东西，要拔插任何东西必须关掉电源，同时确保手上不带静电，接 LCD 的是 5V 电源，千万不要把开发板的 9V 电源接上。）**

以上的只出现一个 GUI 的界面，键盘、鼠标的不能驱动，那是因为我们默认烧写的文件系统里没有配置 USB 鼠标、键盘。应个别客户的要求我们有的板就烧写有配置 USB 鼠标、键盘的文件系统，如果你是这样的客户的话，在板子上电之前你应把带 USB 接口的鼠标、键盘分别插入底版的左上角的 USB host 接口 J20（至少要一个 USB 鼠标，否则你 GUI 就运行不了），这样系统启动起来，你的鼠标、键盘都可以用了。**（光盘里面配置好 USB 鼠标、键盘的文件系统、ramdiskM 就是，而 ramdisk 就没有配置）**

## 2. 测试你的网络

为了激活 LINUX 下的网络，需要直接输入如下指令（LINUX 下的指令）

为你的板子配置 MAC 地址：

```
[root@AT91RM9200DK /root]$ifconfig eth0 hw ether 12:34:56:78:99:aa
```

为你的板子配置 IP 地址：

```
[root@AT91RM9200DK /root]$ifconfig eth0 192:168:0:11
```

PING 你的 PC 主机：

```
[root@AT91RM9200DK /root]$ping 192.168.0.125
```



```

[root@AT91RM9200DK /root]$cd /
[root@AT91RM9200DK /]$ifconfig eth0 hw ether 12:34:56:78:99:aa
eth0: Setting MAC address to 12:34:56:78:99:aa
[root@AT91RM9200DK /]$ifconfig eth0 192.168.0.11
eth0: Link now 100-FullDuplex
[root@AT91RM9200DK /]$ping 192.168.0.125
PING 192.168.0.125 (192.168.0.125): 56 data bytes
64 bytes from 192.168.0.125: icmp_seq=0 ttl=128 time=0.8 ms
64 bytes from 192.168.0.125: icmp_seq=1 ttl=128 time=0.3 ms
64 bytes from 192.168.0.125: icmp_seq=2 ttl=128 time=0.3 ms
64 bytes from 192.168.0.125: icmp_seq=3 ttl=128 time=0.3 ms
64 bytes from 192.168.0.125: icmp_seq=4 ttl=128 time=0.3 ms
64 bytes from 192.168.0.125: icmp_seq=5 ttl=128 time=0.3 ms
64 bytes from 192.168.0.125: icmp_seq=6 ttl=128 time=0.3 ms
64 bytes from 192.168.0.125: icmp_seq=7 ttl=128 time=0.3 ms
64 bytes from 192.168.0.125: icmp_seq=8 ttl=128 time=0.3 ms
-

```

连接的 2:07:12 ANSIW 115200 8-N-1 SCROLL CAPS NUM 捕 打印

要暂停运行，键入 Ctrl+C 即可退出。

说明：板子 IP 地址（192:168:0:11）和 PC 主机 IP 地址（192:168:0:125）必须处在同一网段内，比如板子 IP 地址（192:168:10:33）和 PC 主机 IP 地址（192:168:10:100）均可。

### 3. 测试 CF 卡

插上 CF 卡，跳线 J35 短接(主盘)，J23 短接 2、3 脚(3.3V),启动系统。

```

~ #mount -t vfat /dev/discs/disc0/part1 mnt/
~ #cp 1.txt /mnt
~ #mkdir /mnt/susu

```

查看 mount 上的目录，可以看到该目录下有刚才拷贝的文件和建立的目录。

```
~ #umount mnt/
```

将其 umount 后，再次 mount 上来可以发现拷贝的文件仍然存在，这时删除该文件然后 umount，再次 mount 后，可以发现拷贝的文件已经被删除，由此可以该分区可以正常读写。

### 4. 测试 SmartMedia Card

插上 SmartMedia Card，J15 断开，J16 短接，启动系统

```

~ #mount -t yaffs /dev/mtdblock/0 /mnt
~ #cp 1.txt /mnt
~ #mkdir /mnt/susu

```

查看 mount 上的目录，可以看到该目录下有刚才拷贝的文件和建立的目录。

```
~ #umount mnt/
```

将其 umount 后，再次 mount 上来可以发现拷贝的文件仍然存在，这时删除该文件然后 umount，再次 mount 后，可以发现拷贝的文件已经被删除，由此可以该分区可以正常读写。

### 5. 测试 U 盘

插上 U 盘，启动系统

```
~ #mknod /dev/sda1 b 8 1
~ #mount /dev/sda1 /mnt
~ #cp 1.txt /mnt
~ #mkdir /mnt/susu
```

查看 mount 上的目录, 可以看到该目录下有刚才拷贝的文件和建立的目录。

```
~ #umount mnt/
```

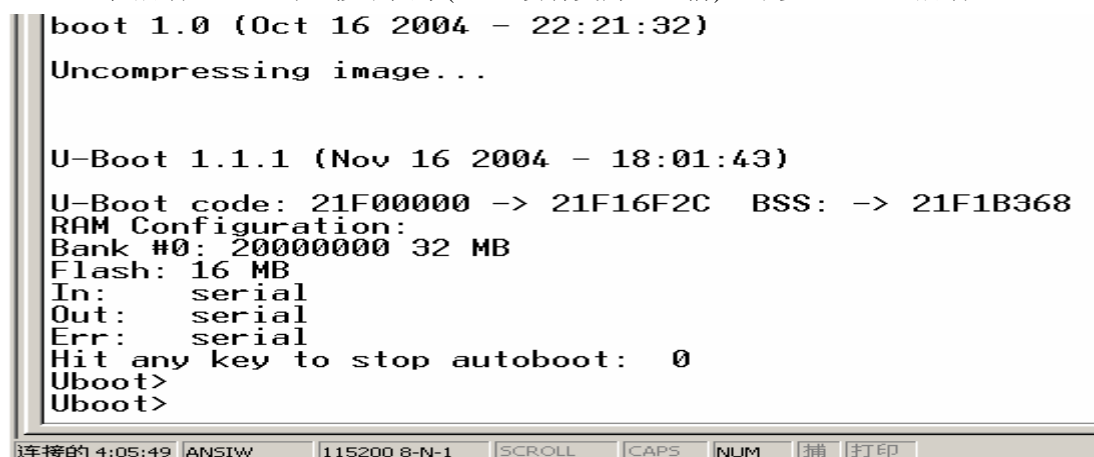
将其 umount 后, 再次 mount 上来可以发现拷贝的文件仍然存在, 这时删除该文件然后 umount, 再次 mount 后, 可以发现拷贝的文件已经被删除, 由此可以该分区可以正常读写。

## 6. 直接下载 bin 文件到你的板子跑

在光盘里面我们提供板子上各部分功能的测试 bin 文件, 他们均是在 ADS 1.2 下写的测试程序, 在光盘目录(EBD9200-I 测试文档与映射)里有他们的 bin 文件, 同时光盘目录(EBD9200-I 型开发板测试代码)相应源代码。

使用步骤:

1) 在启动 u-boot 时, 按下回车(3.2.1 没有变为 0 之前), 不要让 Linux 启动。

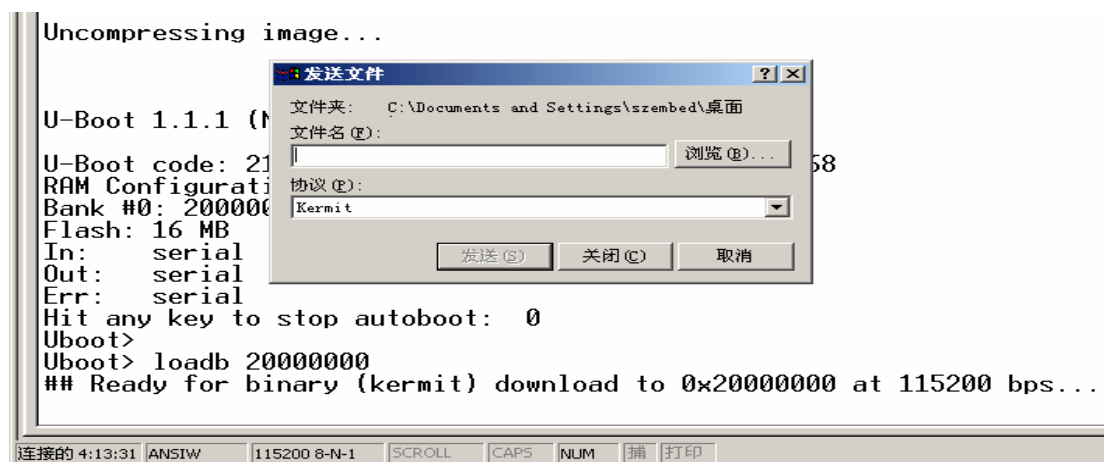


```
boot 1.0 (Oct 16 2004 - 22:21:32)
Uncompressing image...

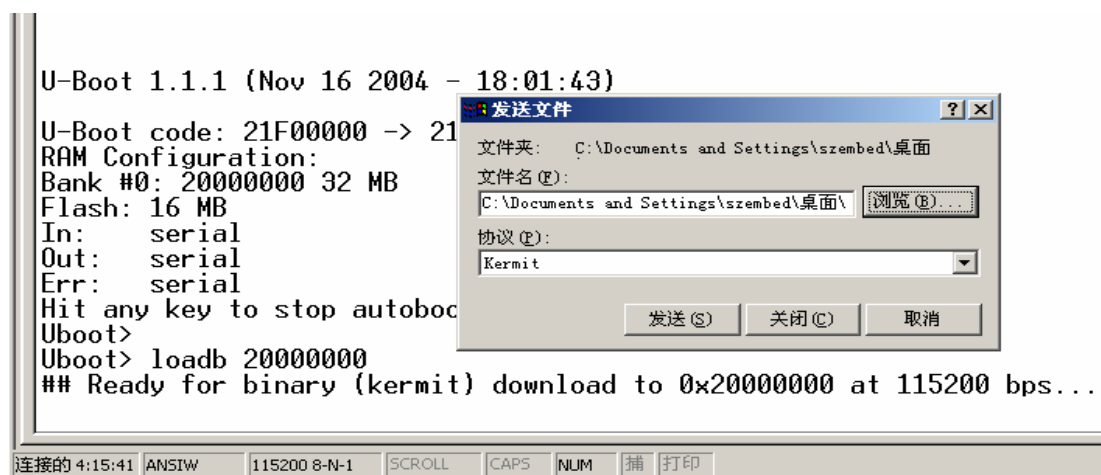
U-Boot 1.1.1 (Nov 16 2004 - 18:01:43)

U-Boot code: 21F00000 -> 21F16F2C BSS: -> 21F1B368
RAM Configuration:
Bank #0: 20000000 32 MB
Flash: 16 MB
In: serial
Out: serial
Err: serial
Hit any key to stop autoboot: 0
Uboot>
Uboot>
```

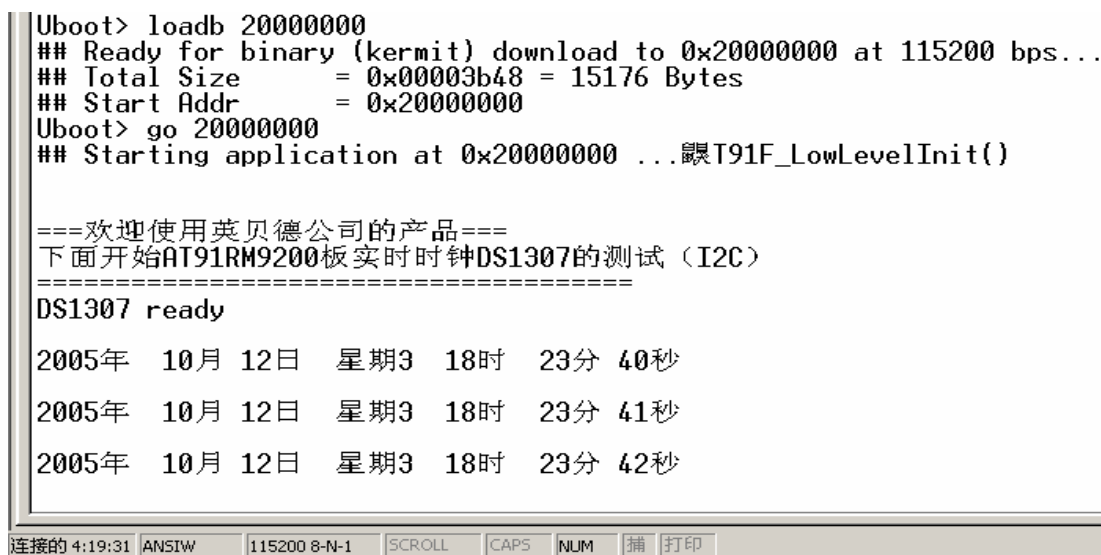
2) 输入命令 **Uboot>loadb 20000000** //指定下载映射在 SDRAM 的地址  
回车, 接着点鼠标右键->选择 发送文件, 出现窗口如下:



选择要发送的文件 (协议为 kermit), 接着点发送。



敲入 go 20000000 回车 运行如下:



每一秒钟显示一串。

## 7. 采用 ADS 和 Magic-ICE 开发 EBD9200 的步骤

对于 ADS1.2 和 Magic-ICE 的详细内容, 本说明书不作详细介绍, 在此, 你应该在了解 EBD9200 的基础上, 知道英贝德为你提供了什么? 如何使用 Magic-ICE 的步骤。

### STEP 1:

如果你采用英贝德提供的 ARM 开发工具, 那么你可以放心的使用和开发, 如果采用别的开发商提供的开发工具, 请确认你的开发工具是否支持 ARM920T;

### STEP 2:

进行跳线设置, 你必须把 J23 跳线到左端, 这样你才能使用 ICE 工具;

### STEP 3:

你需要从盘中找到 ARM9 的配置文件 (AT91RM9200DK.cfg), 在用 Multi-ICE Server 连接时使用;

### STEP 4:

现在你可以进行开发了, 先运行 Multi-ICE Server, 进行 ARM9 的配置, 然后你可以象以往开发 ARM 一样进行 EBD9200 的开发。

更具体的说明在第 4 章 采用 ADS 和 Magic-ICE 开发。

## 7. 采用嵌入式 Linux 开发 EBD9200 的步骤

对于嵌入式 Linux 的开发流程，首先可以参考文档《嵌入式 Linux 在 ARM 中的移植》，而对于嵌入式 Linux 在额 EBD9200 中的开发，我想有下述步骤。

### STEP 1:

首先你应该了解的是：**嵌入式 Linux 操作系统的组成**。我们可以和 PC 机相对应来理解，在 PC 机上，Windows 的启动大致有 BIOS、内核、文件系统和初始化程序几个部分，当然相对应而言，嵌入式 Linux 的移植有 Bootloader、Linux 内核、文件系统、初始化和用户的应用程序几部分，BootLoader 完成系统的初始化和引导。

### STEP 2:

上述部分可能开发商都给你提供编译好的文件，你只需要按照说明书下载，但是如果你需要更改，这就是你接下来应该了解的：**嵌入式系统的开发环境，是交叉开发环境**，这就需要交叉编译器，一般开发商光盘中都有，你可以直接使用，否则，你需要到网站上下载。

### STEP 3:

你现在可以进行 EBD9200 的 Linux 系统开发了，检查下面的内容是否具备：

- 编译好的 BootLoader、Linux 内核影象和 Linux 文件系统影象
- 交叉编译环境
- 开发文档

### STEP 4:

你可以在 Linux 下编辑、编译你的上层程序，对 Linux 内核、文件系统进行设置和编译，最后按照说明书进行烧写 FLASH 和运行。

更具体的说明在 [（第 5 章 嵌入式 linux 在 EBD9200 上的开发）](#)