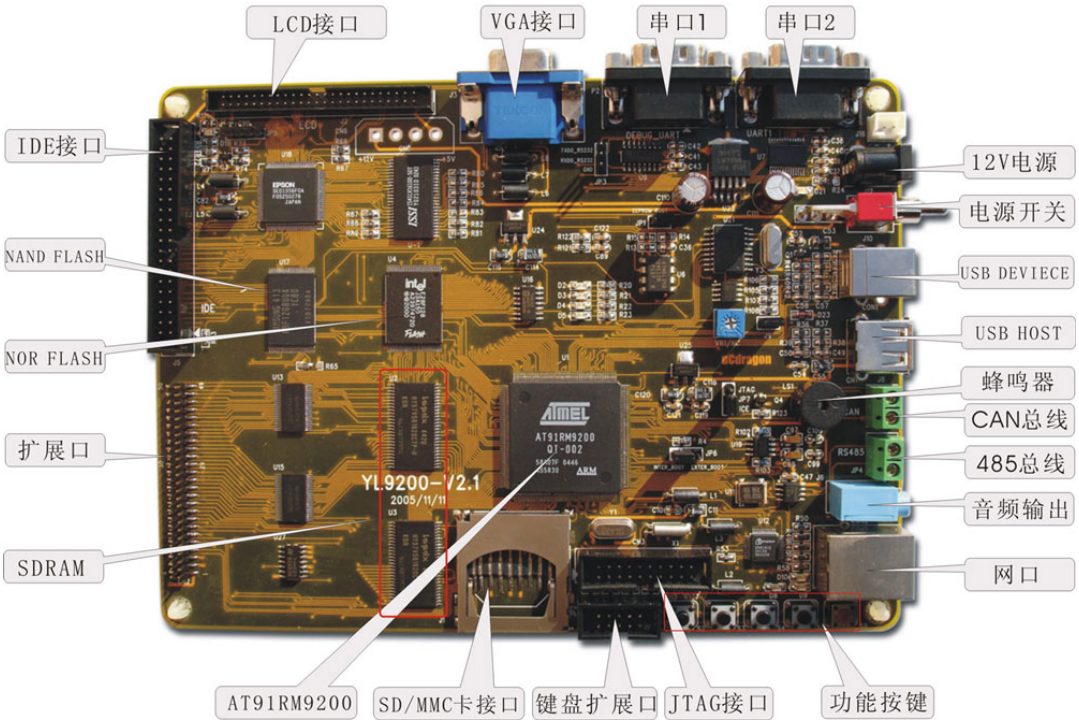
YL9200 使用手册 V2.2

第一章 YL9200 开发板套件介绍

1.1 YL9200 开发板简介

YL9200 是一款 ARM920T 内核的工业级的开发板，CPU 内嵌 100M 以太网，带有 USB2.0 协议的 USB HOST 和 Device 接口，支持 SD 卡、IIS 音频和全功能 9 线串口等。主频 180MHz，带有 MMU 存储器管理单元。性能稳定，功能强大，是工业控制、网络通讯等应用的首选。



YL9200 V2.00 开发板硬件资源:

中央处理器

—— AT91RM9200（ARM920T）主频 180M(可稳定超频到 240MHZ)，工业级；

外部存储器

—— 16K 字节 IIC 接口的 EEPROM,用于存储上电时的引导程序 BOOT；

—— 64M Bytes NAND Flash（用户可自己更换为 16M、64M 或 128M 的 NandFlash）；

—— 64M Bytes SDRAM(2 片 16 位的 SDRAM 芯片组成 32 位接口)；

—— 16M Bytes Nor Flash（一片 intel E28F128）；

2D 图形加速器

—— 外部扩展的 2D 图形加速引擎，具有独立的 2M 字节显示存储器；

—— 一个标准 VGA 接口，最大支持 16BPP 模式 800×600 分辨率；

—— 一个 TFT 液晶屏接口，最大支持 16BPP 模式 800×600 分辨率；

串口

—— 一个串口为标准三线 RS232 接口；

—— 另一个串口为标准 9 线 RS232 接口，可接 Modem；

RS485 接口

—— 一个 RS485 通讯接口；

网口

—— 一个 100M 网口(AT91RM9200 内部 MAC+外部 PHY)，带发送、接收和联结指示灯；

USB 接口

—— 一个 USB HOST（USB 2.0 Full Speed）接口；

—— 一个 USB Device（USB 2.0 Full Speed）接口；

CAN 总线接口

—— 一个 CAN 总线接口，全面支持 CAN2.0A 和 CAN2.0B 协议；

音频接口

—— 采用 IIS 专用接口芯片，一路立体声音频输出接口可接耳机或音箱；

IDE 接口

—— 40 芯标准连接器，可挂载 IDE 硬盘，板上自带标准 IDE 电源接口；

存储卡接口

—— 一个 MMC/SD 卡接口，可支持 256M 的 MMC 卡；

调试和下载接口

—— 一个 20 芯 Multi－ICE 标准 JTAG 接口，支持 SDT2.51,ADS1.2 等调试 ；

其他

—— RTC 实时时钟；

—— 一个带功率驱动的蜂鸣器；

—— 四个接在 GPIO 口线上的高亮 LED；

—— 四个接在 GPIO 口线上的小按键；

—— 一个 10PIN 标准连接器，可接 4×5 矩阵键盘；

—— 一个复位按键，采用专用芯片进行复位；

—— 一个 50 芯 2 毫米间距用户扩展口，引出了地址线、数据线、读写、片选、中断、IO 口、5V 和 3.3V 电源、地等用户扩展可能用到的信号；

电源接口

—— 开关电源供电，输入直流电压范围是 7～15V（推荐使用 12V，这样 IDE 硬盘可共用电源），带电源指示灯；

BootLoader

—— 预装 BIOSBOX（在 NAND FLASH 中 ）；

操作系统

—— 预装 linux2.4.26 操作系统；

YL9200 的特点：

—— 100M 网口,采用了 RJ45+网络变压器一体化的网口连接器，降低了干扰；

—— 通过串口与网口方式下载程序,调试方便；

—— 带有一个可接 Modem 的 9 针串口；

—— USB HOST 接口可支持 U 盘等 USB 设备；

—— 操作系统: linux 2.4.26,支持 yaffs2，cramfs 以及 fat 文件系统

用户光盘上提供的开发工具和源代码：

1) ADS1.20 安装程序(评估版)；

2）板子上所有硬件的 demo 演示程序,全部提供源代码。

3) Linux_for AT91RM9200 的内核源码包和编译器等；

4) YL9200 核心板和底板电路原理图（pdf 格式）；
5) YL9200 开发板使用手册（pdf 格式）；
6) 开发板上所用到的部分芯片手册、资料；

YL9200 套件包括:

1) 一块已测试好的 YL9200 开发板
2) 一张 YL9200 光盘
3）一条串口线(两边都是母头，交叉串口线)
4）一条网线(交叉网线)
5) 一个＋12V 直流电源
6）一个包装盒

1.2 硬件资源分配

（1）跳线说明

跳线名称	说明
JP1	**在内部启动模式下**，决定是否从 IIC EEPROM 启动程序 **接上跳线帽：**程序从 IIC EEPROM 启动程序，默认 **取下跳线帽：**利用串口来下载启动程序（XMODEM 协议）
JP3	**串口 0 引出线**
JP6	**决定系统是从内部启动还是从外部 NOR FLASH 启动** **1+2：内部启动，默认** **2+3（EXTER_BOOT）:从外部 NOR FLASH 启动**
J9	选择 CAN 总线的匹配电阻，根据用户需要
JP7	决定 CPU 的调试模式，是 JTAG 还是 ICE 模式 1+2（JTAG）：CPU 工作在 JTAG 模式下。 **2+3（ICE）：CPU 工作在 MULTIC-ICE 模式下。默认**

（2）接口说明

接口名称	说明
P1，P2	全功能串口，调试串口

J3	VGA 接口
J2（LCD）	TFT LCD 接口
J5	IDE 接口
CN2	50 针用户扩展接口
J1	MMC/SD 卡接口
CN3	JTAG 接口
JP2	4*5 键盘扩展口
CON2	音频接口
JP4	RS485 接口
CAN	CAN 接口
CN1	USB HOST 接口
CON1	USB Device 接口
J10	电源开关
J12	电源接口
J16	背光电源接口

1.3 Linux 操作系统支持的驱动

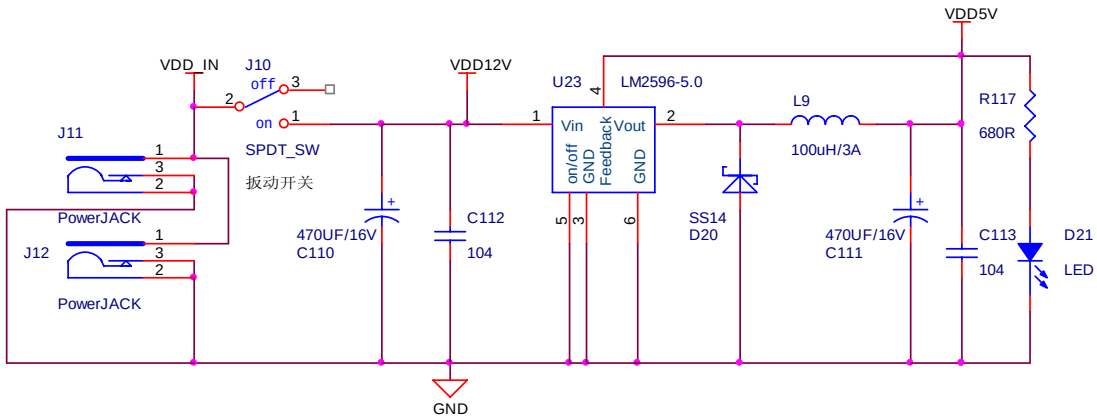
YL9200 开发板提供的嵌入式 Linux 的内核版本是 linux-2.4.26, Linux 下所支持的驱动如下：

- 串口
- 以太网驱动
- USB HOST
- 音频驱动
- MMC 驱动
- VGA 驱动
- IDE 驱动
- MTD,（NAND FLASH，NOR FLASH）
- LED 指示灯驱动
- 支持多种文件系统，如 YAFFS，FAT,CRAMFS 以及 fat 文件系统

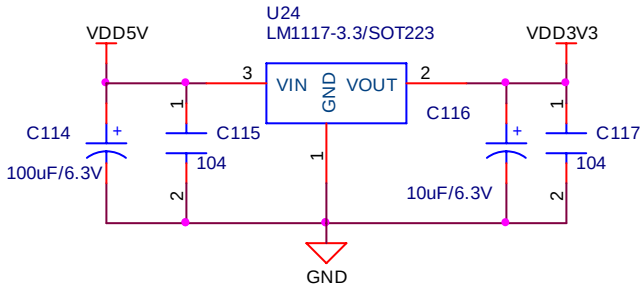
第二章 YL9200 开发板硬件电路说明

2.1 电源电路

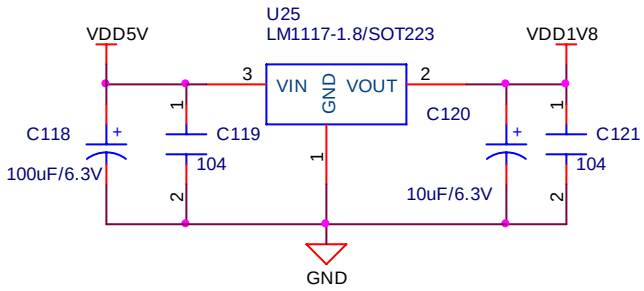
主电源供电电路采用单片开关稳压电源 LM2596-5.0，其最大输出电流可达 3A，具体电路如下：



CPU 及外围 3.3V 器件采用 LM1117-3.3 芯片供电，具体电路如下:

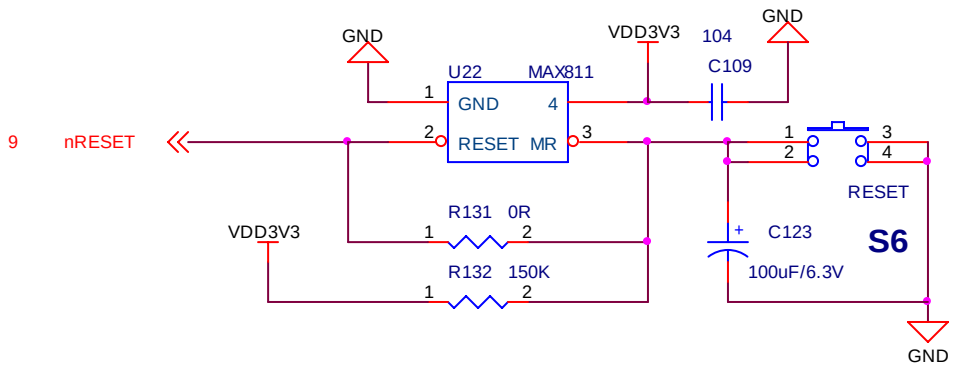


CPU 核心 1.8V 电压采用 LM1117-1.8 进行供电，如下图所示：



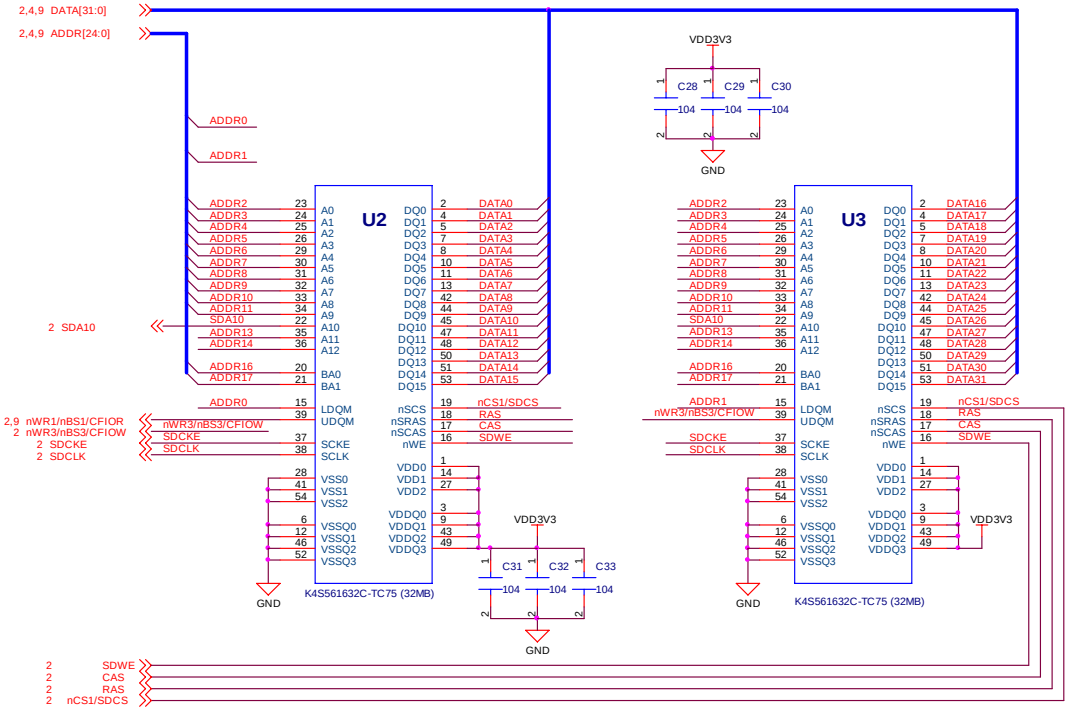
2.2 复位电路

由于 AT91RM9200 的复位时间要求较长，我们采用一个阻容电路来实现上电复位的长复位时间，实际应用过程中可再用专用的复位芯片 MAX811 来实现复位。具体电路如下：(当前开发板上的 MAX811 没有焊接,用 0Ω 电阻短接的)



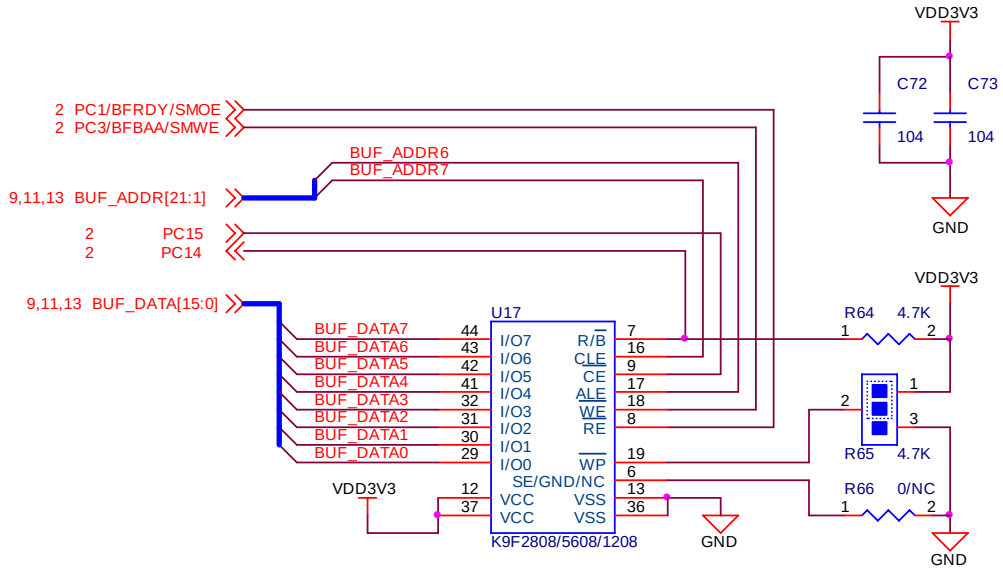
2.3 SDRAM 电路

SDRAM 电路可以使用两片 16M 的 SDRAM 或者是两片 32M 的 SDRAM，具体电路如下:



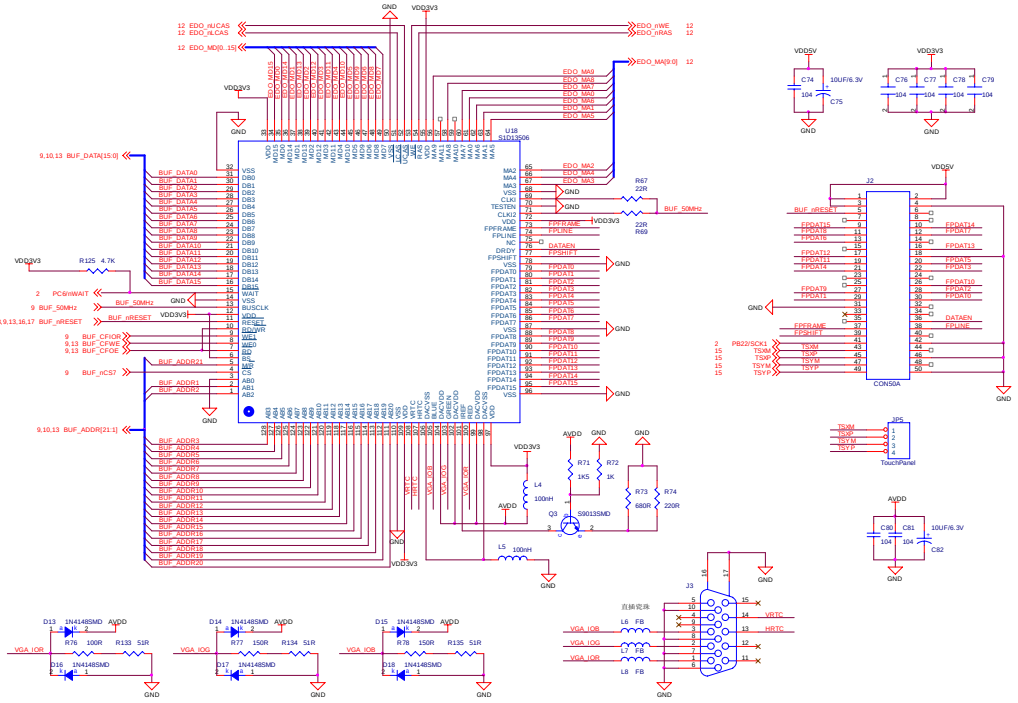
2.4 NAND FLASH 电路

WP#接高电平允许擦除和写入；接低电平禁止擦除和写入。



2.5 LCD 及 VGA 显示电路

由于 AT91RM9200 没有 LCD 控制器,所以我们要使用 LCD 时需要用一个 LCD 控制器完成操作。如下图所示：



其它电路的具体设计请参看光盘中提供的原理图，这里不再多述。

第三章 YL9200 开发板使用和相关资源测试

3.1 启动 Linux

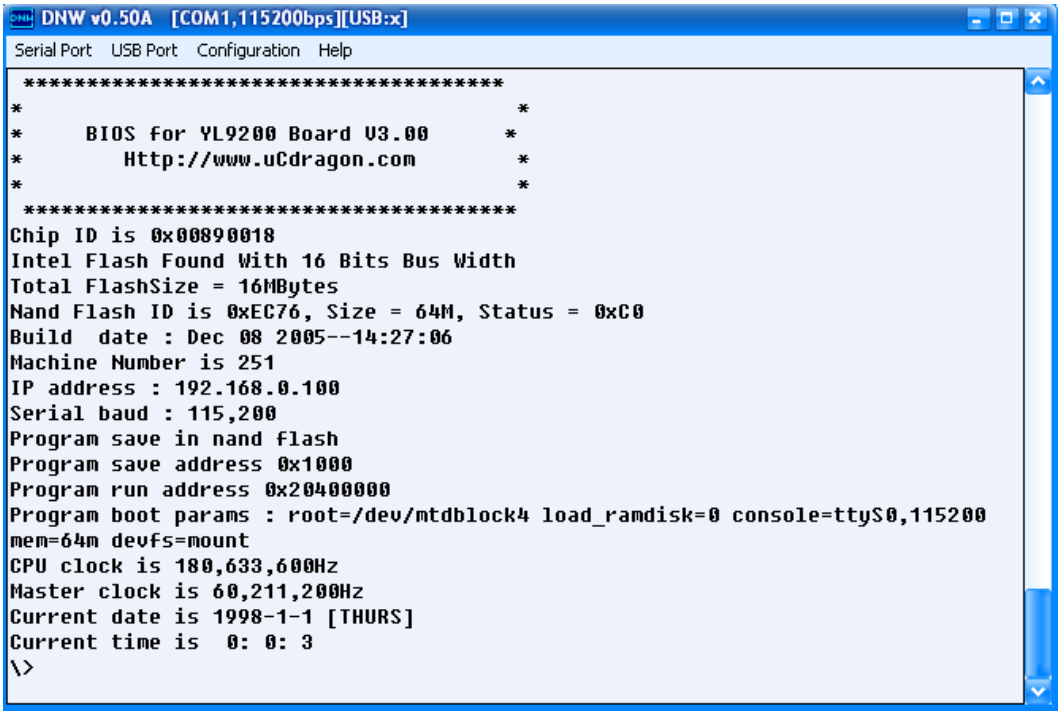
在出厂之前，Linux 已经固化在 YL9200 的存储器中了，下面将做好一些准备工作。先用串口线将开发板的串口 P2 与 PC 机的串口连接起来，打开串口工具超级终端或 DNW.exe（在光盘的“实用工具”文件夹下）。

串口工具的参数：波特率 115200，8 位，无奇偶位，停止位 1，无硬件流。

接着，接好电源，上电，将启动开发板中的 Linux，这时 D2 LED 指示灯不停的闪烁，并且在超级终端或 DNW.exe 里面有 Linux 启动的相关信息。

3.2 在 BIOS 下测试部分资源

在上电后，在蜂鸣器响过一声后，立即按住开发板上的 S3 按键，将停止装载 Linux，进入 BIOS 的命令状态，如下图所示：**(具体的 BIOS 启动及操作过程请参看第四章 BIOS 烧写及相关说明)**



BIOS 跑起来了，说明 IIC EEPROM，串口，NAND FLASH，蜂鸣器（BIOS 的启动的过程中，蜂鸣器会叫一声），按键以及 LED 指示灯是正常的。

（1）网络测试

BIOS 启动后，可以在 BIOS 的启动信息中看到开发板的 IP address:192.168.0.100 信息，说明开发板的 IP 地址为 192.168.0.100，这时要 PC 的 IP 地址与开发板的 IP 地址要在同一网段，我这里的 PC 机 IP 地址为 192.168.0.7。另外一种方法，是将开发板的 IP 地址设置成与 PC 机的 IP 地址在同一网段（命令 ipcfg ***.***.***.***，接着输入命令 senv 来保存设置的参数）。

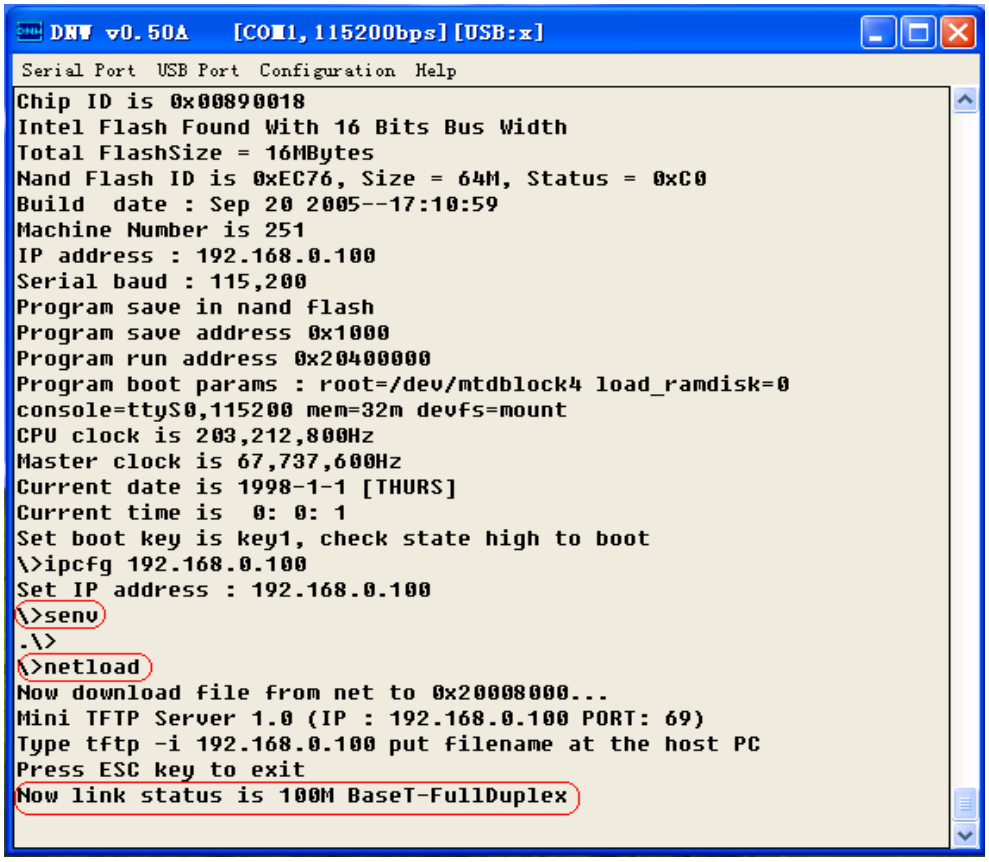
用交叉网线将 PC 的网络接口与开发板的网络接口相连。

IP 地址设置命令： ***ipcfg 192.168.0.100***

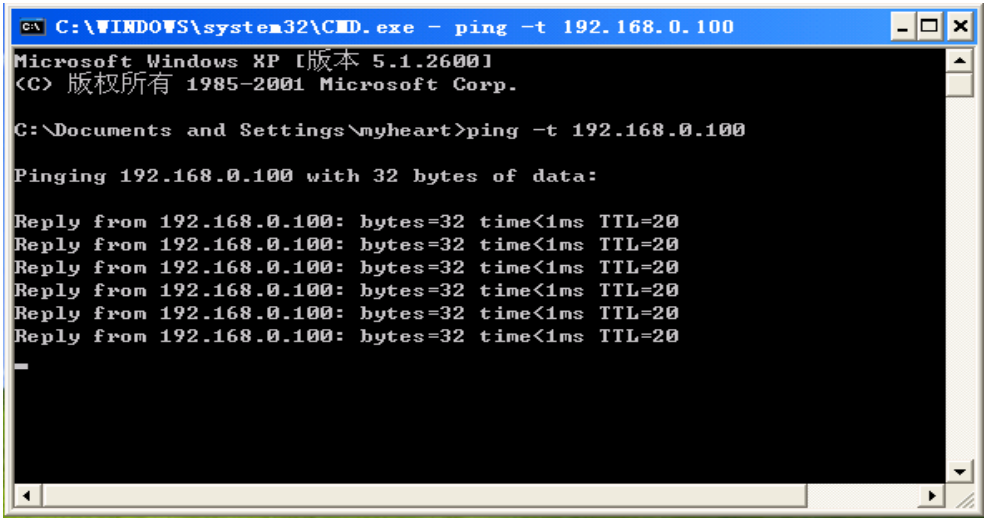
然后输入保存命令：***senv***

接着输入命令：***netload***

界面如下：



这时，在 PC 机端打开命令窗口，在命令窗口输入 ping –t 192.168.0.100 来测试开发板的网络是否是通的，下面是网络成功的信息：



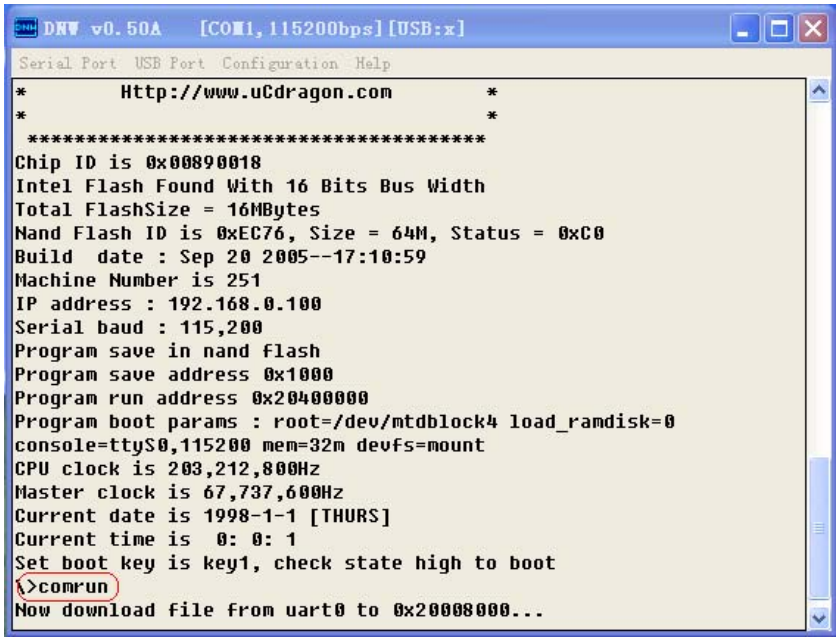
这样，网络接口工作 OK。

（2） USB DEVICE 测试

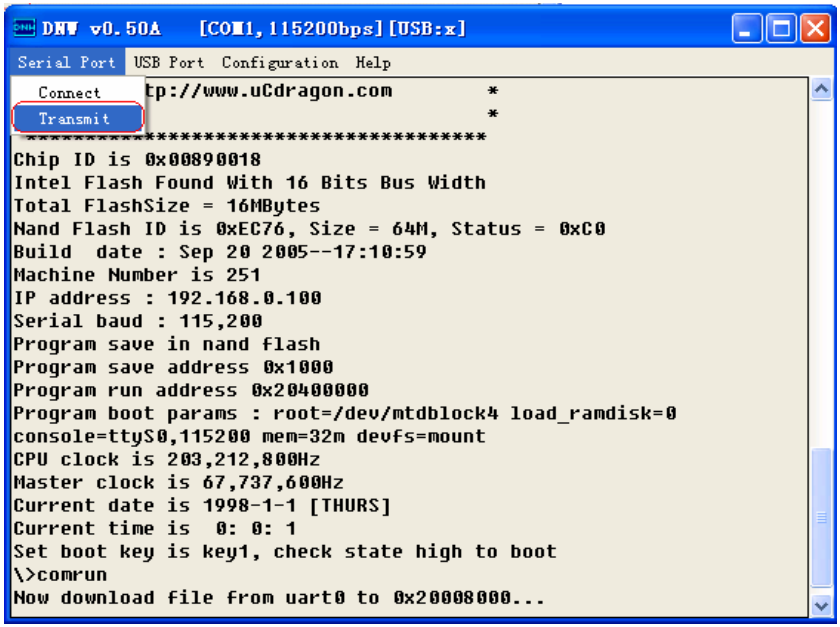
注意在进行 USB DEVICE 测试的时候，串口工具用 DNW.EXE，同时需要从内部启动，即将 JP6 的跳线的位置**设置在 1+2 的位置（丝印是 INTER BOOT）**。

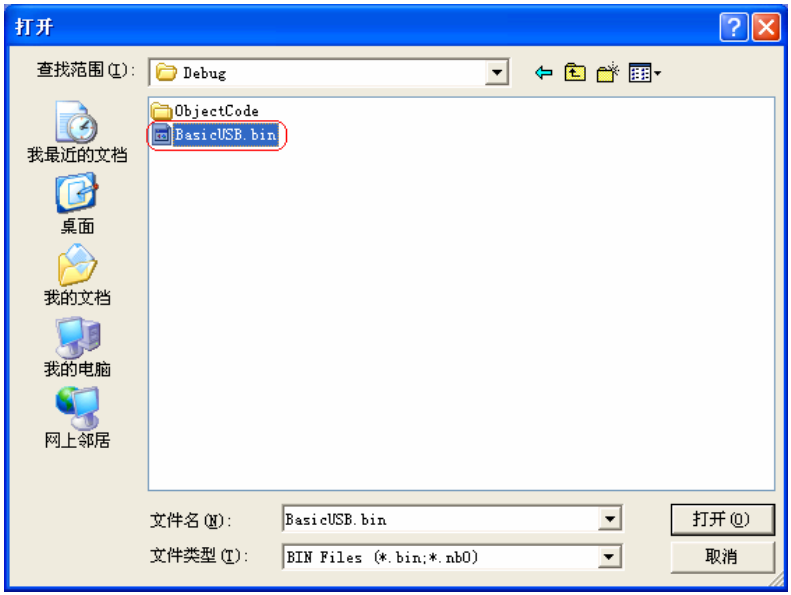
进入 BIOS 状态后，输入命令：

comrun ；并回车，这个命令主要是通过串口来下载程序并运行

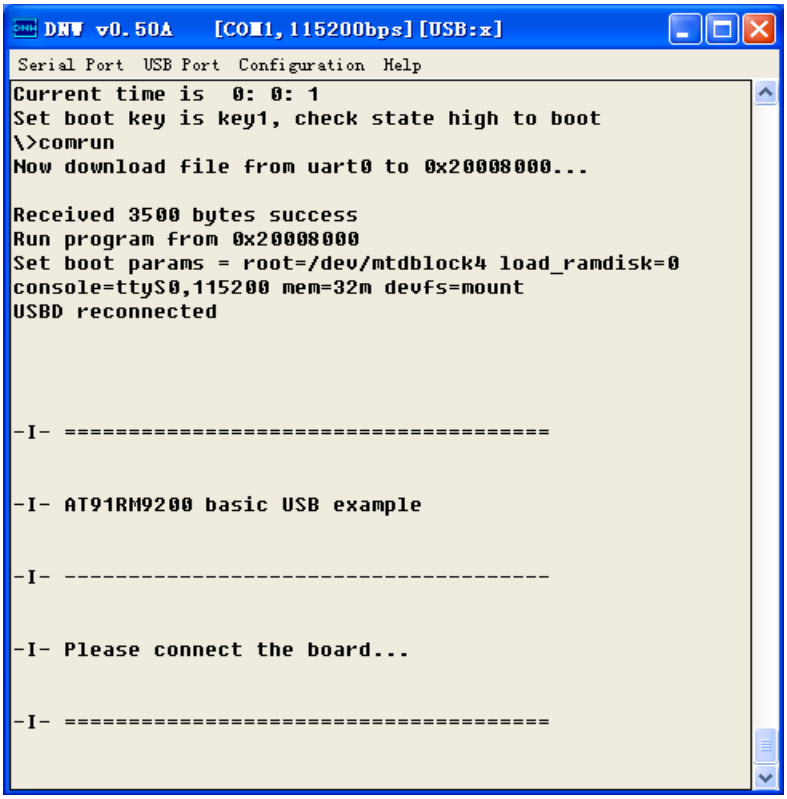


接着点击 DNW.EXE 的“Serial Port”→“Transmit”，接着选择要传输的文件 BasicUSB.bin（.bin 格式的可执行文件），这个文件的位置在光盘的“BIOS 和 Demo 程序源码”下。\BasicUSB\COMPIL\BasicUSB_Data\Debug\BasicUSB.bin





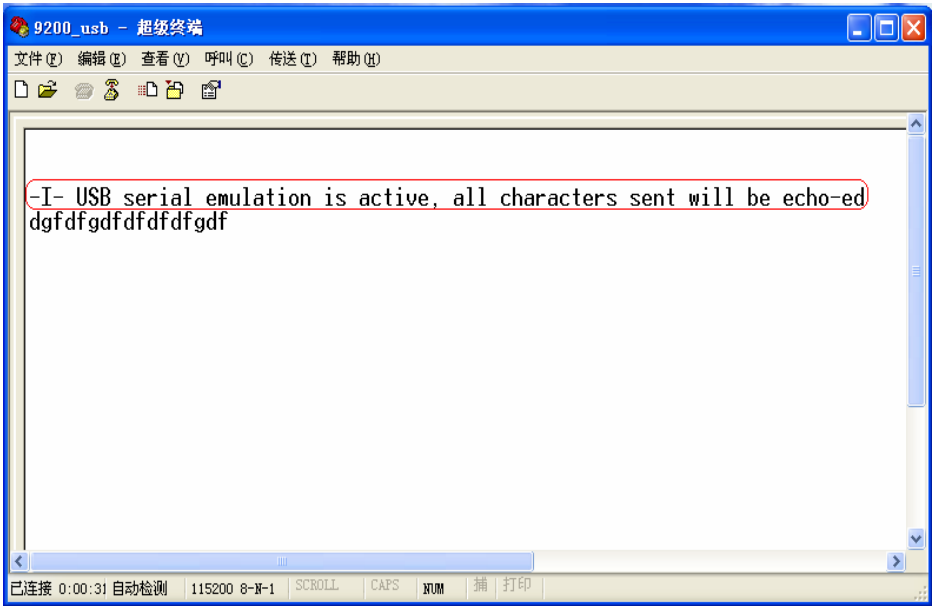
文件传输结束后，会自动执行这个 USB DEVICE 测试程序，程序运行起来，出现如下信息：



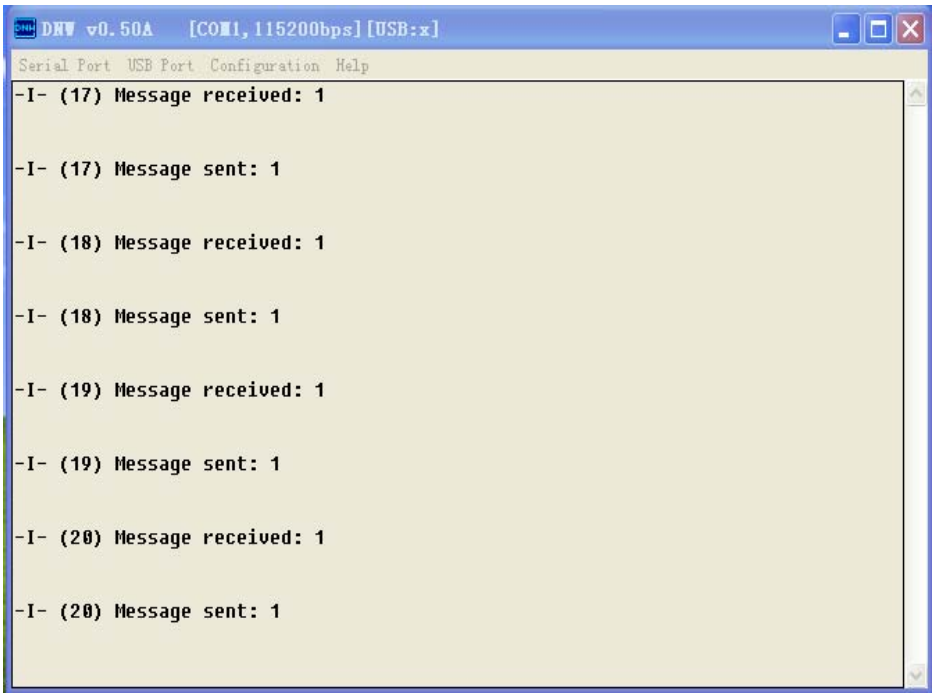
出现上面的信息后，用 USB 线，将 PC 机的 USB 接口与开发板的 USB DEVICE 接口（CON1）连接起来。

如果是第一次运行这个测试程序，PC 机将提示“发现新硬件”，这时候需要安装 USB DEVICE 驱动，驱动文件为\光盘\BasicUSB\atmusb6119.inf 文件（这个驱动文件，主要是将 USB DEVICE 模拟成一个串口，所以访问这个串口就跟访问 USB DEVICE 设备一样），将 USB DEVICE 驱动安装好后，将生成一个新的串口（USB DEVICE 模拟的）COM3（或者是 COM4）。

接着新建一个以 COM3 为接口的超级终端，参数为：波特率 115200，8 位，无奇偶位，停止位 1，无硬件流。



在该超级终端中输入任意字符，这时将在 COM1 口的 DNW 工具中有相应的信息显示，显示信息如下：



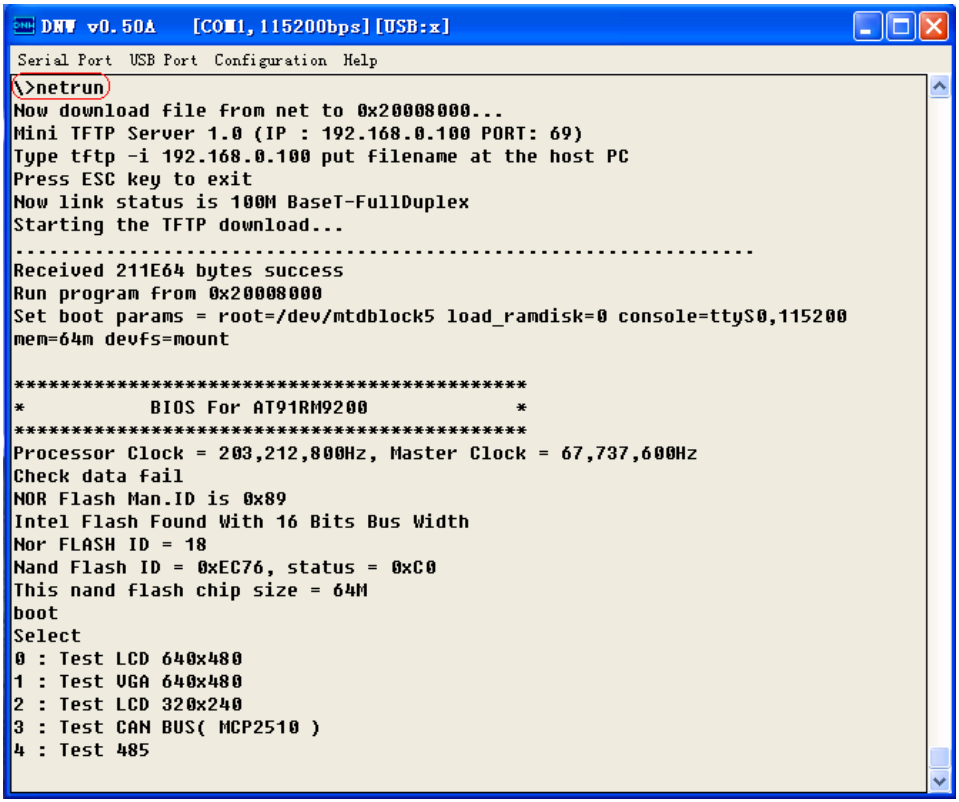
出现上面的信息，说明 USB DEVICE 的测试 OK。

（3）VGA 和 CAN 的测试

进入 BIOS 的命令状态，输入命令：

netrun

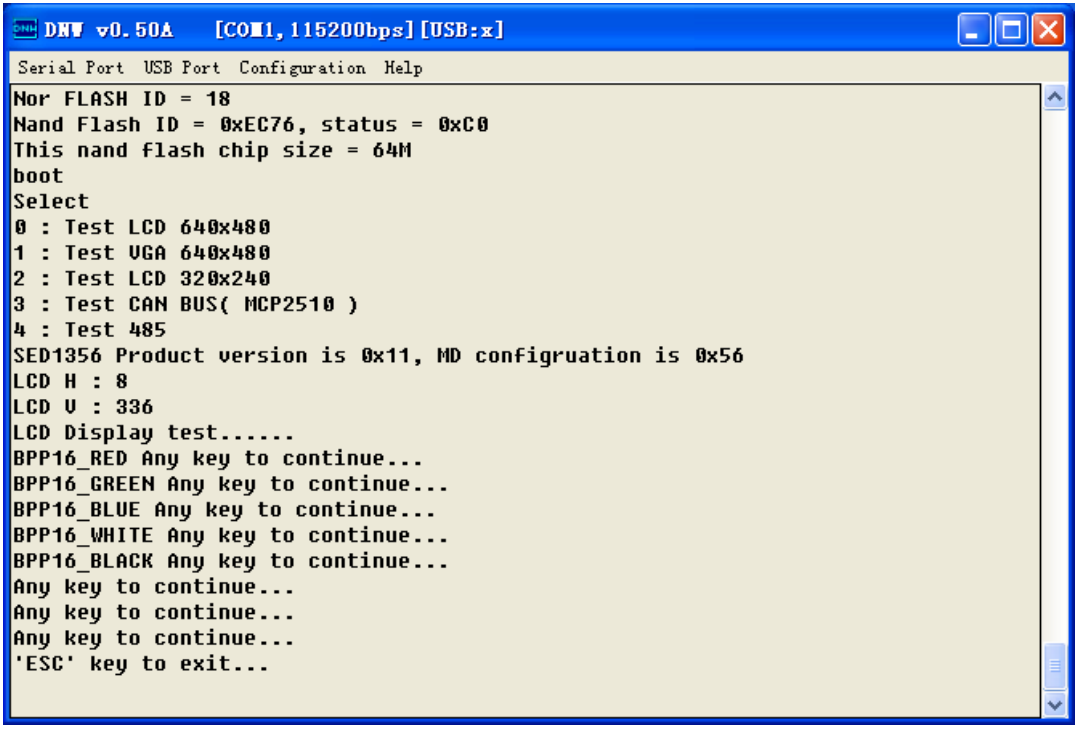
注意：这个命令是网络下载运行命令，再运行这个命令的时候，首先需要将开发板的 ip 地址设置为 192.168.0.100，同时 PC 机的 IP 地址要与开发板的 IP 地址再同一网段。然后回车。接着点击批处理文件 YL9200_Test.bat **\光盘\ 目标代码\YL9200_Test.bat** 传输结束后，将自动执行该程序。



TFT-LCD 640x480 测试

选择“0”，将进行 8 寸屏（该 8 寸 LCD 屏为选配件，TFT）的 LCD 测试，接着将 8 寸 LCD 用 50 针排线与 YL9200 的开发板的 J2 连接起来，这时按任意键可以看到 LCD 屏

上图片的切换。

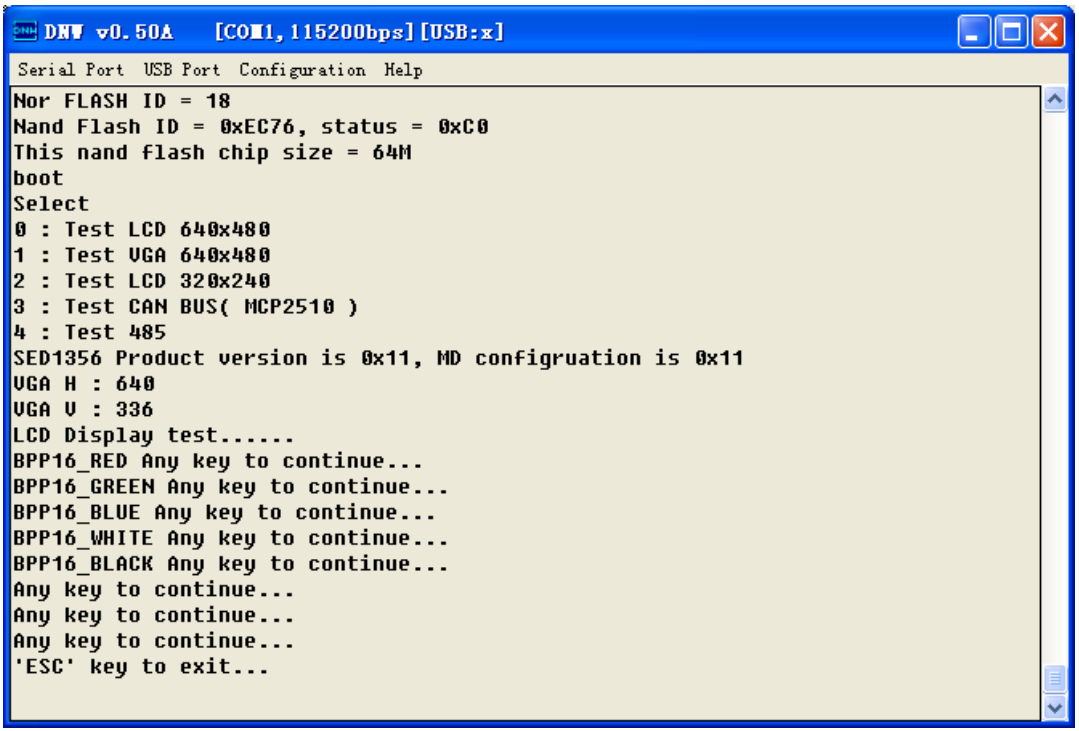


图片显示结束后，按“ESC”键返回主测试菜单。

VGA 640x480 测试

选择**“1”**，将进行 VGA 测试，这时将 VGA 接口连接好，

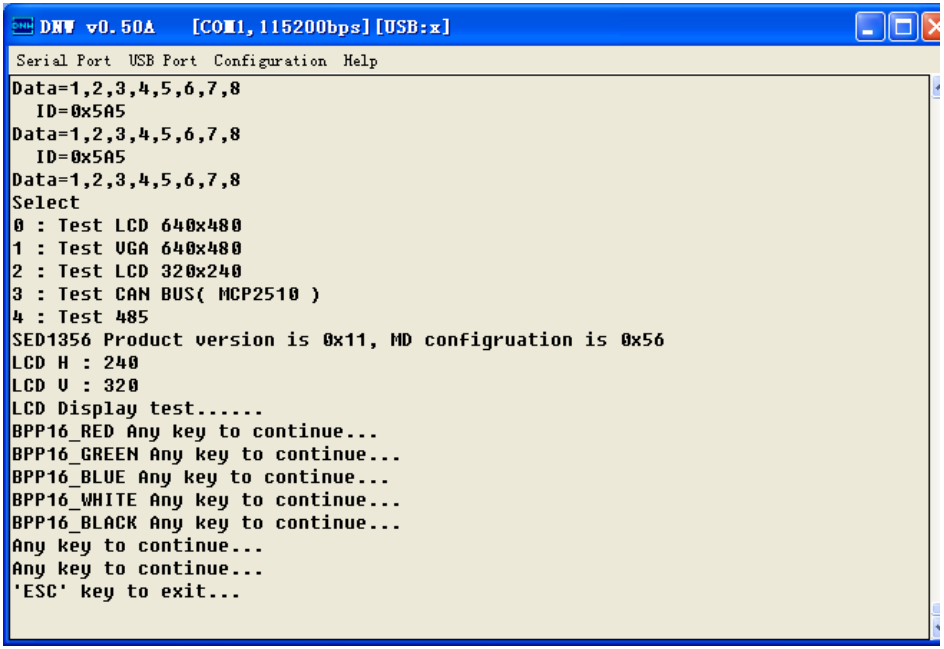
按任意键将切换 VGA 显示的图形界面。



图片显示结束后，按“ESC”键返回主测试菜单。

TFT-LCD 320x480 测试

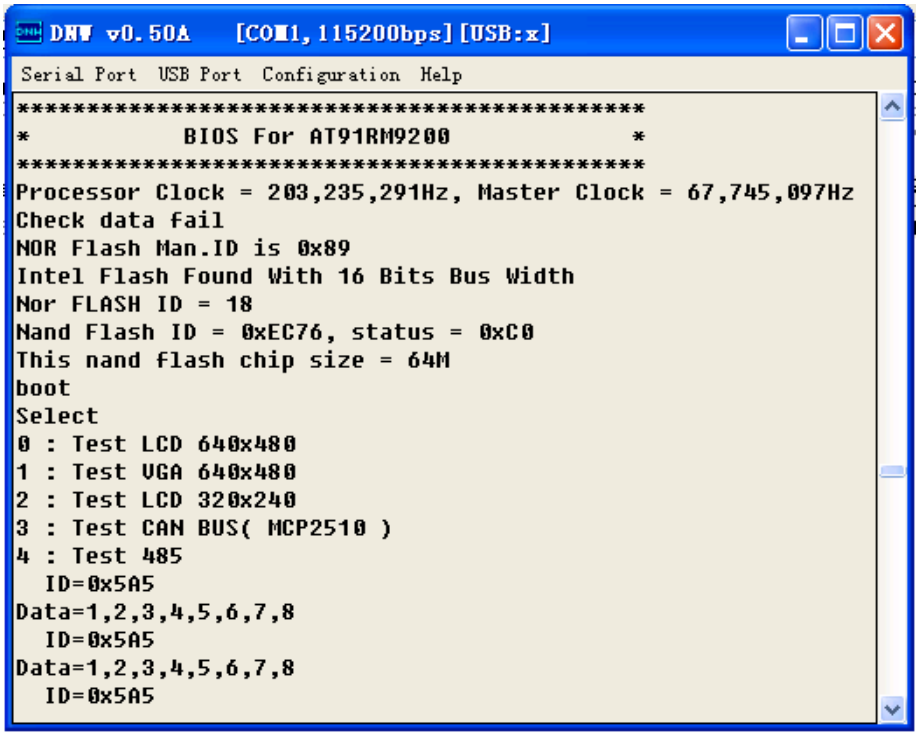
选择“2”，将进行 TFT LCD 320x480(Sharp 3.5 寸)的测试，这时连接好 3.5 寸的 LCD 屏，接着在 DNW 串口按任意键，可以切换在 LCD 屏上显示的图片，



按“ESC”键可以退出该项测试。

CAN 测试

接着输入“3”，将进行 CAN 总线测试，CAN 测试，主要时让 CAN 工作在回环模式下。



可以按“ESC”键退出。

RS485 测试

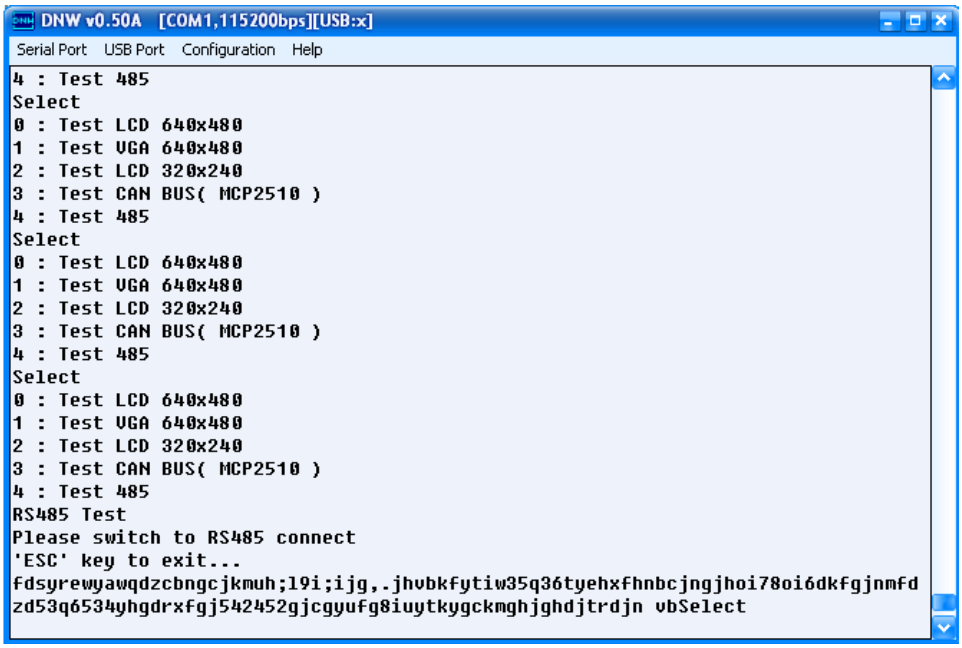
返回主菜单后，接着可以进行 485 测试了

在进行 485 测试前，请用 485 转 232 的转换器，将开发板的 485 接口与一条直连网线连接起。在下面的测试，就是这样前提下进行的。

选择**“4”**，接着将串口线切换到 485 转换器，然后连接 485 接口，我这里对应的仍是 COM1。（波特率 115200，8 位，无奇偶位，停止位 1，无硬件流）

在 COM1 所对应的 DNW 工具中，输入任意字符，这时将会在本 DNW 中显示输入的字符。

显示信息如下：



3.3 Linux 操作系统下的部分资源测试

在 BIOS 状态下，可以通过命令 mrun 来启动 Linux,另一种方法是让 Linux 自启动。

进入 Linux 状态，可以进行 USB HOST 测试，MMC 测试，音频测试，网络测试以及 IDE 硬盘挂接测试。

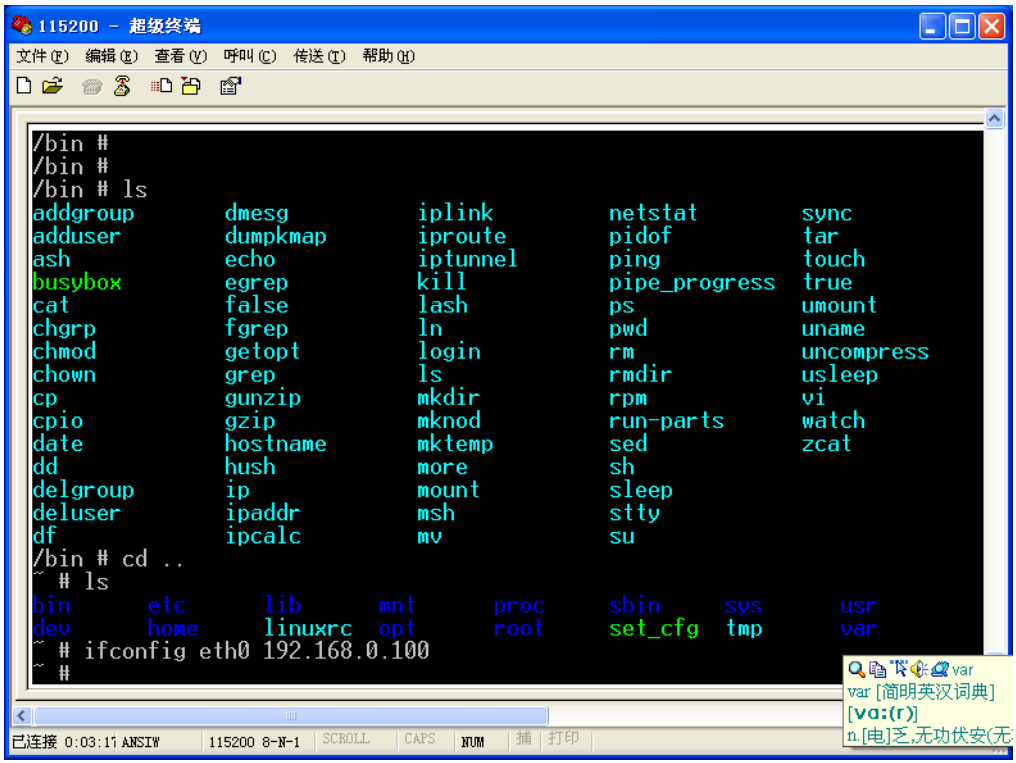
注意在进入开发板的 Linux 环境后，将串口工具换成超级终端。

（1） 网络测试

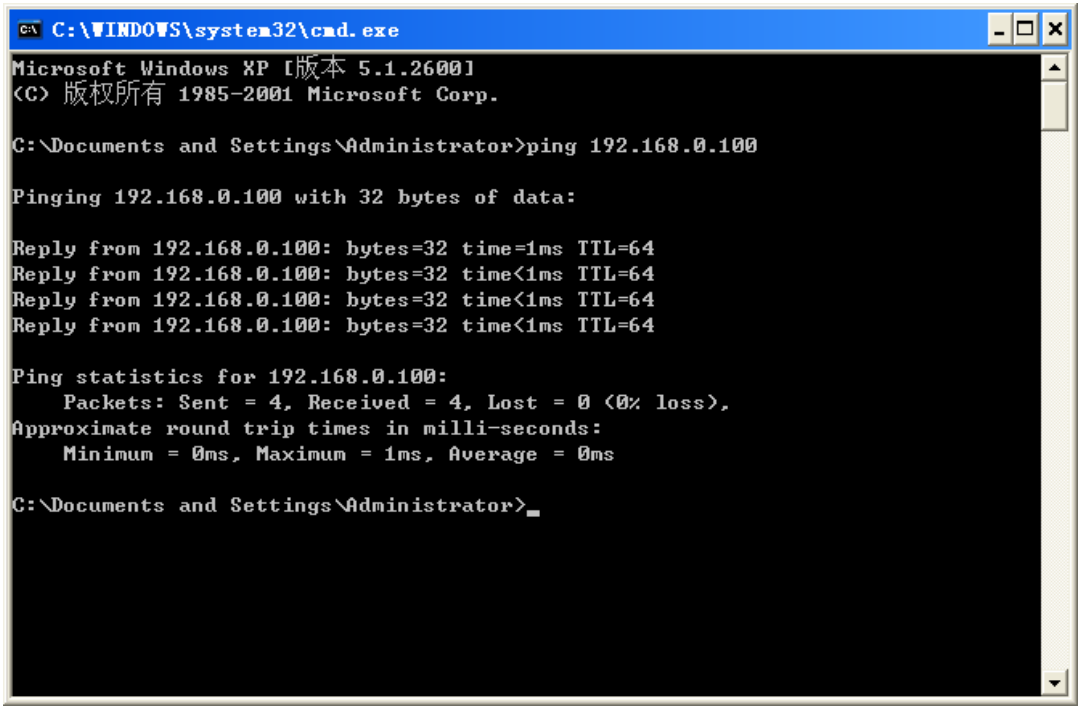
首先用交叉网线将板子的网络接口与 PC 机的接口连接起来。

必须设置好开发板的 IP 地址（IP 地址必须与 PC 机的地址一致），设置命令如下：

ifconfig eth0 192.168.0.100

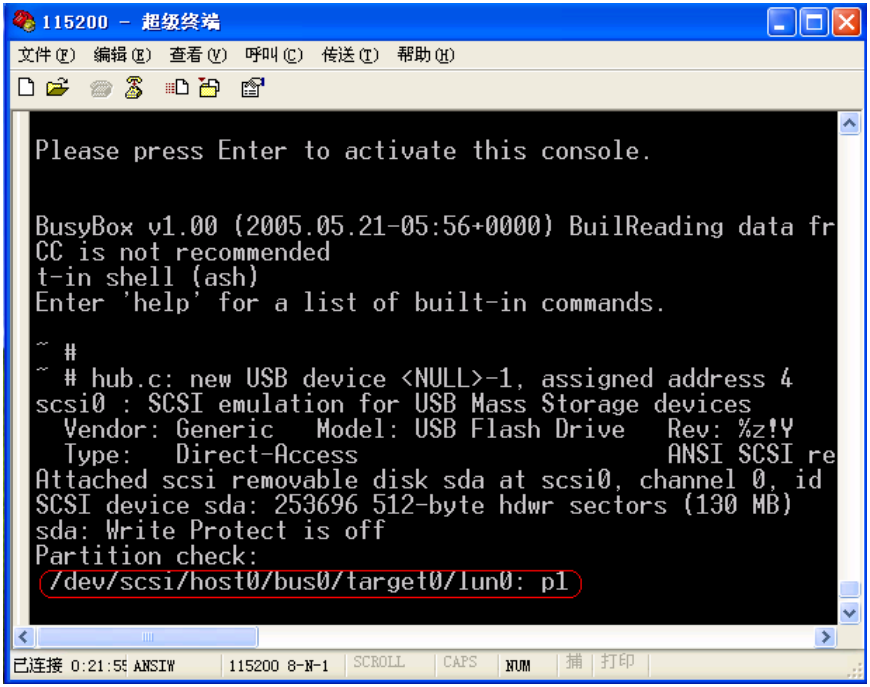


这时在 PC 机端用 ping –t 192.168.0.100 命令来测试 Linux 下是否好的。



（2）USB HOST 测试

Linux 启动后，将 U 盘插入开发板的 USB 接口，如果 Linux 检测到 U 盘，将出现如下信息：



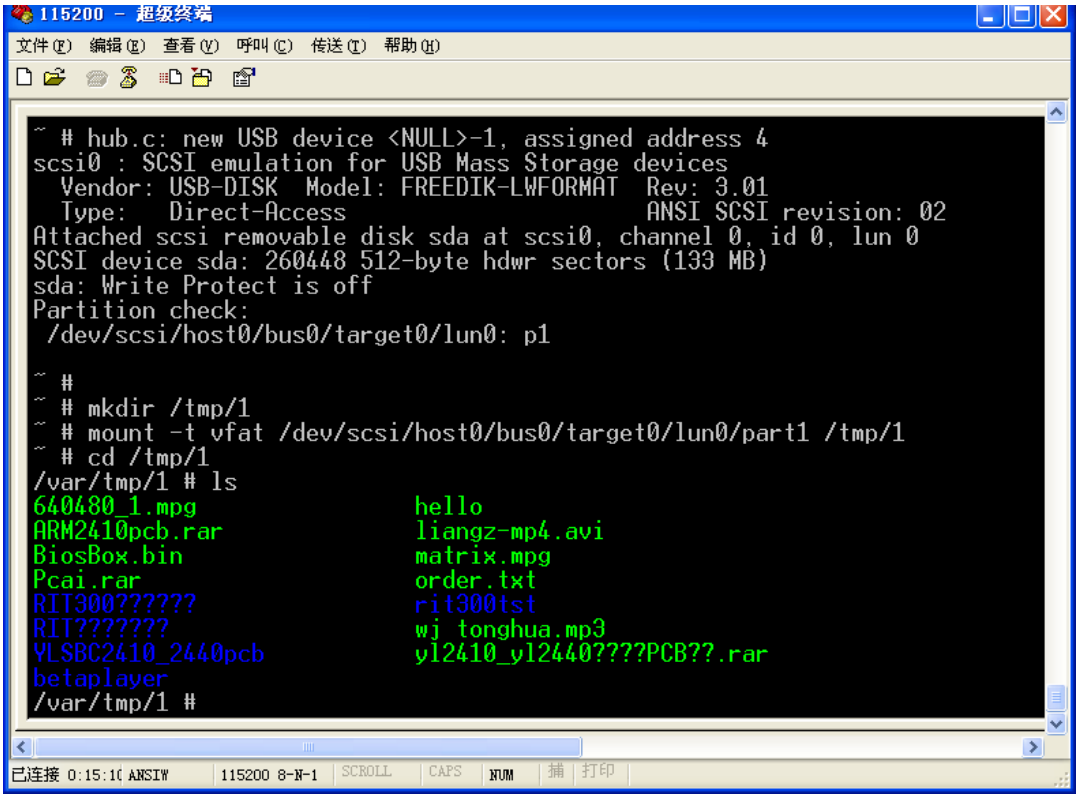
出现“/dev/scsi/host0/bus0/target0/lun0: p1”信息，说明 U 盘被正确检测到，这时就进行 U 盘挂接操作了。

命令如下：

mkdir /tmp/1 ;建立一个用来挂接 U 盘的目录

mount -t vfat /dev/scsi/host0/bus0/target0/lun0/part1 /tmp/1 ;将 U 盘挂接到 /tmp/1 目录下

ls /tmp/1 ； 查看 U 盘下的文件和目录



这样就可以对 U 盘进行操作了，比如，读写，拷贝，删除等等操作。

（3）音频测试

音频测试，接上一步。

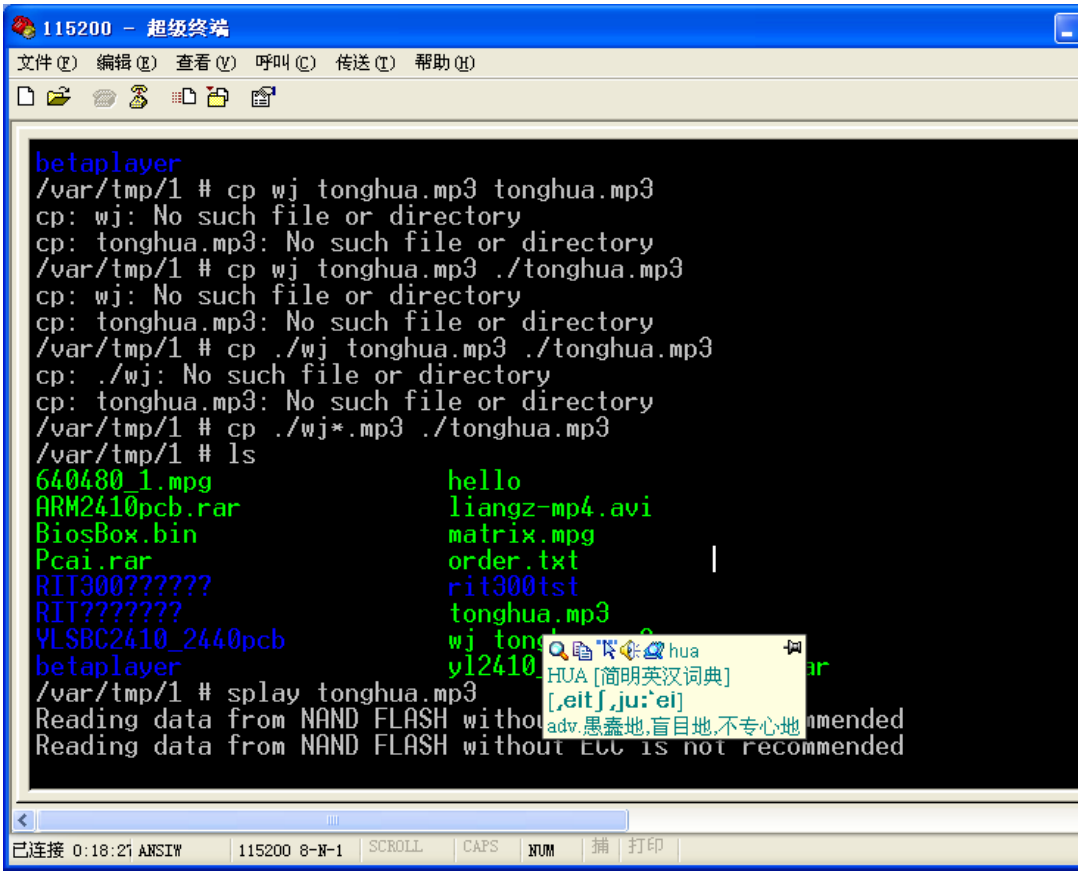
在进行音频测试前，确保 U 盘里放一个.mp3 格式的音频文件。

将耳机或音箱连接到开发板的 J6 接口（音频接口）。

输入命令：

cd /tmp/1 ；进入 U 盘被挂接的目录

splay tonghua.mp3 ；可以是任意的 mp3 音频



（4）MMC 测试

MMC 测试，必须在 Linux 启动前，将 MMC 卡插入 MMC 卡座，Linux 检测到 MMC 卡后，会在/dev/mmc/目录生成一个设备节点/dev/mmc/disc0/part1。

挂接命令如下：

mkdir /tmp/1 ;建立挂接目录(如果前面建了此目录的话要 umout 卸载掉

mount -t vfat /dev/mmc/disc0/part1 /tmp/1 ； 将 MMC 存储卡挂接到/tmp/1 目录下

这样就可以对 MMC 卡的进行相关操作了，比如读写，删除以及拷贝等等。

（5）IDE 硬盘挂接测试

IDE 测试，必须在 Linux 启动前，用硬盘线将 IDE 硬盘与开发板的 IDE 接口连接起来，并给 IDE 硬盘通电，这时再启动 Linux，如果 IDE 硬盘被正确检测到，Linux 将在/dev/ide 目录下生成一个设备节点/dev/ide/host0/bus0/target0/lun0/part1。

IDE 硬盘挂接命令如下：

mkdir /tmp/1 ；;建立挂接目录(如果前面建了此目录的话要 umout 卸载掉建立挂接目录

mount -t vfat /dev/ide/host0/bus0/target0/lun0/part1 /tmp/1；将 IDE 硬盘挂接到/tmp/1 目录下

这样就可以对 IDE 硬盘进行相关操作了，比如读写，删除以及拷贝等等。

3.4 Linux 系统下的一些应用

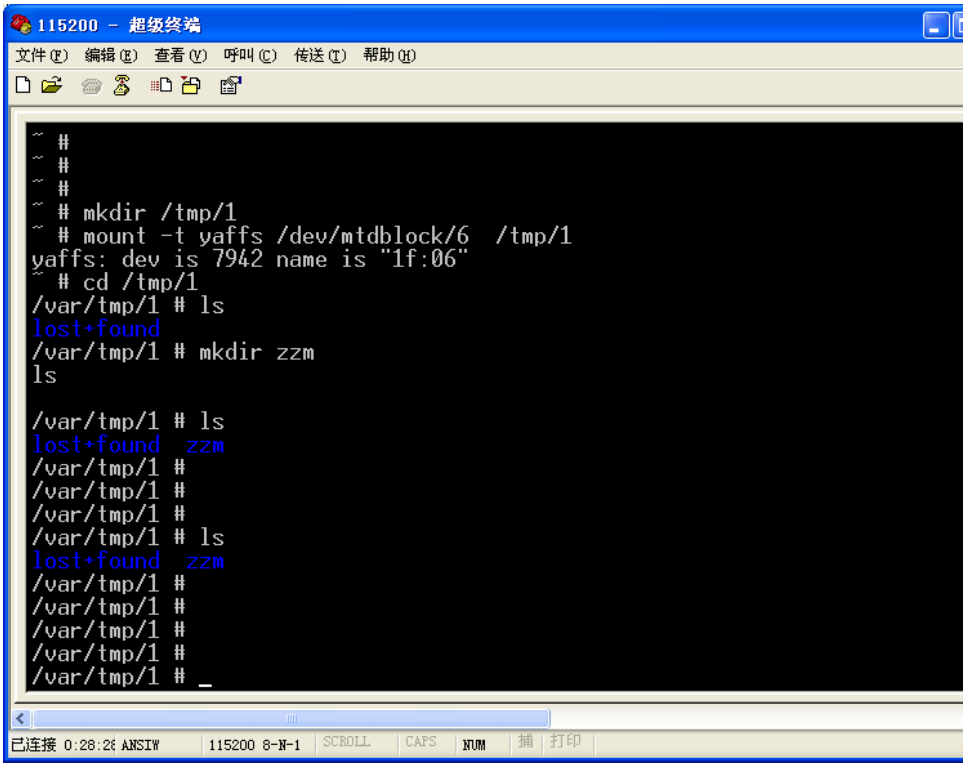
3.4.1 NAND FLASH 的分区挂接

在 Linux 启动的时候，已将 NAND FLASH 分好区了，分区的信息在 Linux 的启动代码里有显示。

NAND FLASH 分区的挂接命令如下：

mkdir /tmp/1 ；建立挂接目录

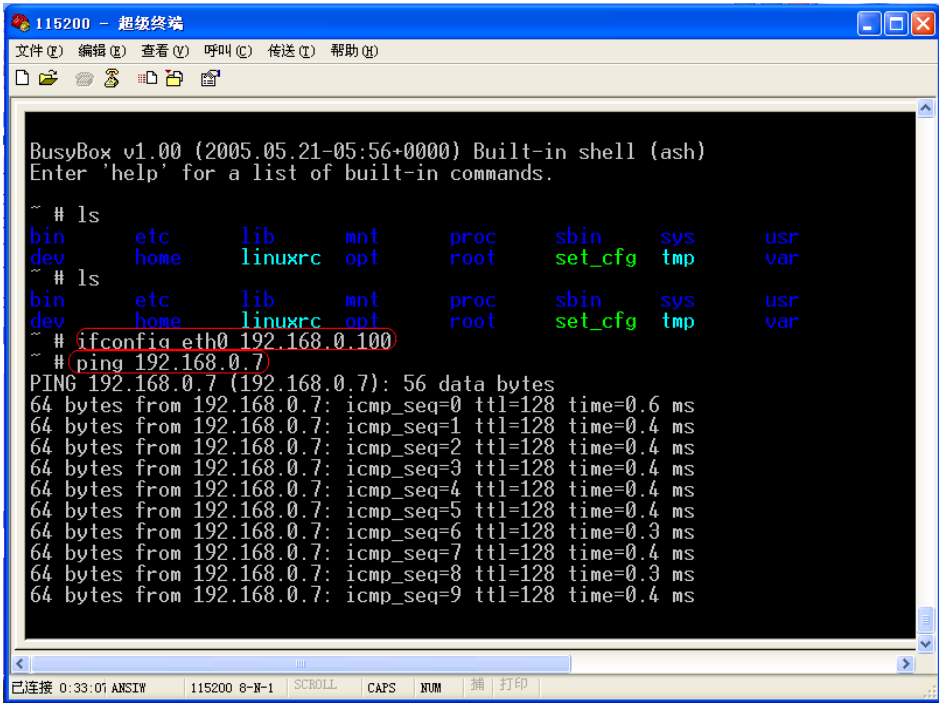
mount -t yaffs /dev/mtdblock/6 /tmp/1 ；将 NAND FLASH 的分区挂接到/tmp/1



3.4.2 网络通讯地址设置

网卡芯片驱动加载成功后，可在 Linux 的 shell 下使用 ifconfig 命令来设置网卡芯片的 IP 地址，之后可 PING 局域网中的其他电脑的 IP 地址。

设置 IP 地址命令:*ifconfig eth0 192.168.0.100*



注意：以后几章节都是工作在 Linux 下，并且 IP 地址为 192.168.0.100。

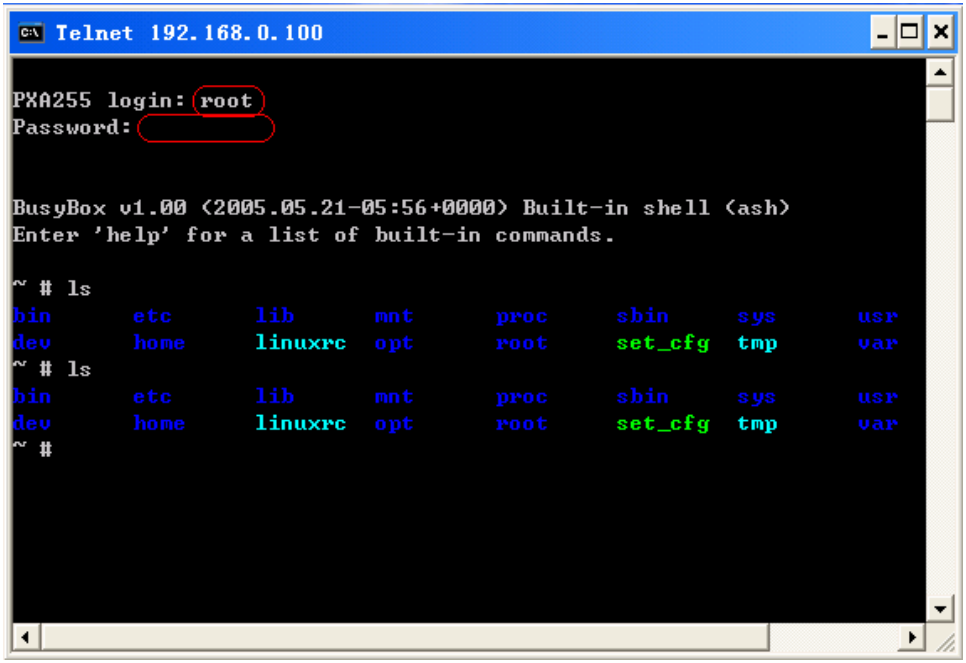
3.4.3 从 PC 机 TELNET 到开发板

在从 PC 机 TELNET 到开发板之前，先要启动 PC 机端的 telnet 服务，这个 TELNET 服务的位置：控制面板－>管理工具－>服务－>Telnet。

接着进入 PC 机的命令窗口，输入命令：

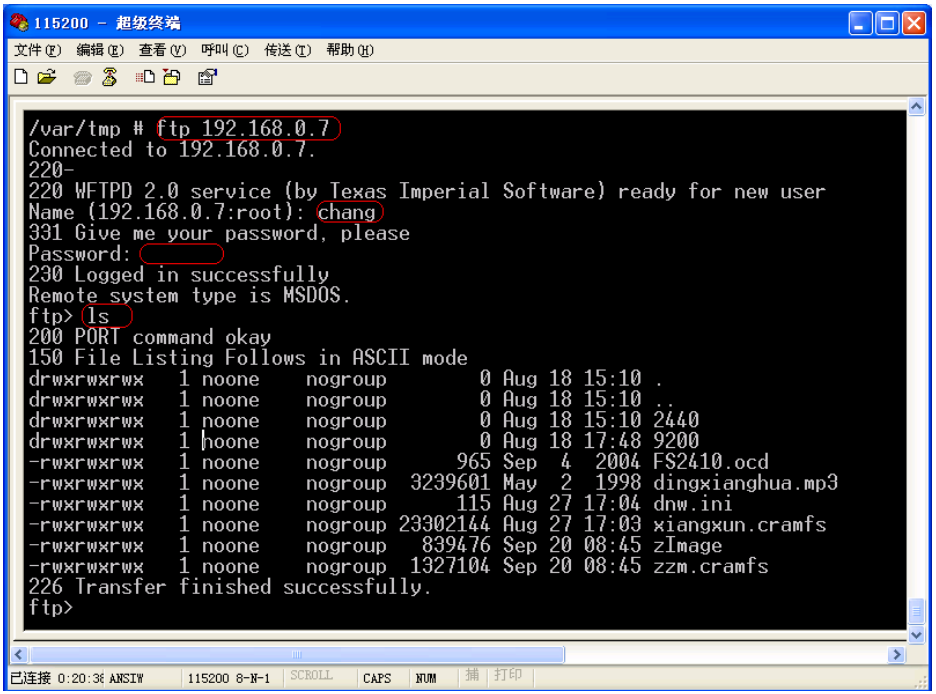
telnet 192.168.0.100

这时会提示要求输入用户名和密码，用户名：root,密码是 linux，可以用 exit 命令退出登陆。



3.4.4 从开发板 FTP 到 PC 机

开发板的根文件系统内带有 FTP 客户端程序，因此也可以在开发板上用 FTP 连接 PC 机，用 bye 指令可断开连接。注意，在运行 FTP 客户端程序的时候，在 PC 机上必须要有 FTP 服务端程序运行，至于 FTP 服务端程序的安装，这里就不说明了。



至于 FTP 服务器用户的密码，是自己在 PC 端设置的。

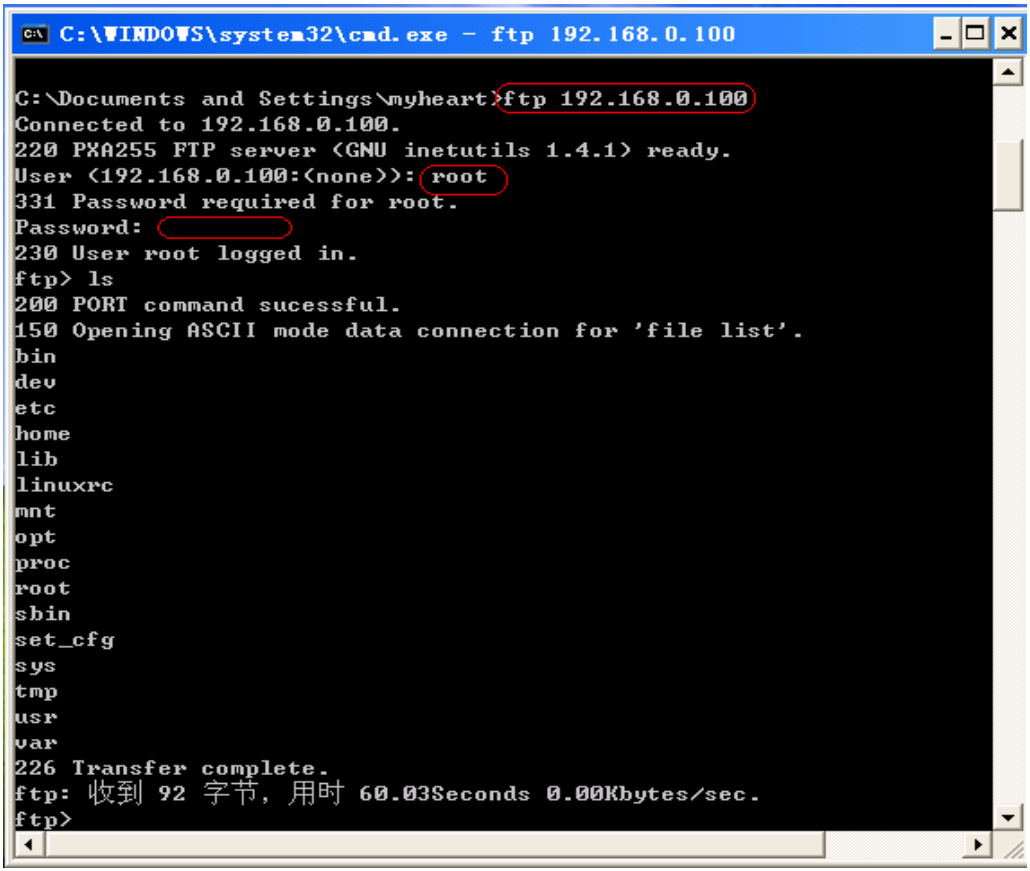
登陆了 PC 机的 FTP 后，就可以从 PC 机上下载和上传文件了，注意，在从 PC 机到开发板的时候要注意，先进入到/tmp 目录，然后在 FTP 登陆到 PC 机，因为/tmp 是在 SDRAM 中建立的空间，可以写的，而其他的目录是只读的。

3.4.5 从 PC 机 FTP 到开发板

开发板的 Linux 有一个 FTP 服务端程序，这样就可以从 PC 机 FTP 到开发板。打开 PC 机的命令窗口，输入命令:

ftp 192.168.0.100

接着回提示要求输入用户名和密码，用户名：root 密码:linux。



3.4.6 在开发板上建立 WEB 服务器

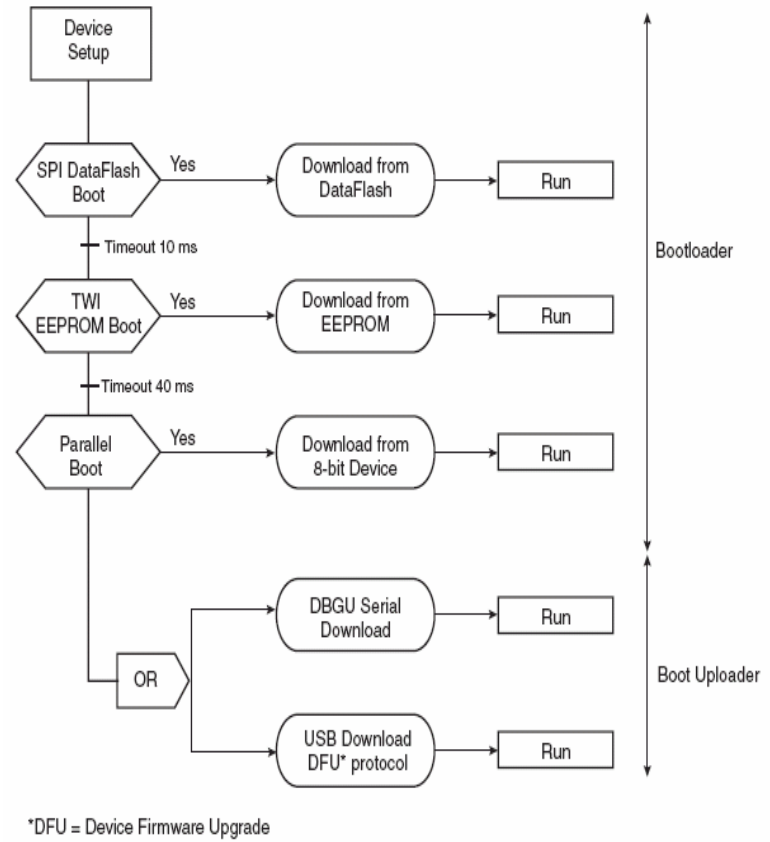
利用 BUSYBOX 所带的 HTTPD 可在开发板上建立一个简单的 WEB 服务器，在 shell 下执行 httpd－h /home/httpd 就可以启动 WEB 服务了，在 PC 上打开网络浏缆器，在地址栏里直接输入开发板的 IP 地址 192.168.0.100 再回车即可看到开发板上的网页。


优龙科技 >> 首页 - Microsoft Internet Explorer
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 收藏夹
地址(D) http://192.168.0.100/
转到 链接
首页 | 优龙产品 | 芯片零售 | 解决方案 | 下载中心 | 如何购买 | 关于我们 | 优龙社区
精品原创
尽在优龙
uCdragon does it quicker
论坛入口
用户名：
密 码：
Cookie：不保存
登 录 清除
新用户注册 忘记密码?
快速搜索
关键字 搜索
解决方案 驱动下载
最新推荐
MORE
FS-PXA255
FS-PXA255核心板：
1.CPU,XSCALE,PXA255,400MHz;2.32M, NOR,FLASH,64M,SDRAM;3.1个电源指示LED,
FS9200
FS9200开发板硬件资源：中央处理器
--AT91RM9200;ARM920T,180M(可稳定超频到240MHZ);工业级外部存储器
--16K,
最新公告
2005年9月24日 星期六
[2004年11月5日]
● 优龙论坛已经开放,敬请光临。
[2004年10月21日]
● 优龙公司将在11月上旬搬迁到玉泉路南山科技创业园
[2004年10月14日]
热门解决方案
(剩下1 项) 正在打开网页 http://192.168.0.100/...
Internet

第四章 烧写 BIOS 及 BIOS 的相关说明

YL9200 可以工作在内部启动（片内启动）模式和外部启动模式。

内部启动模式：通过 CPU 的片内引导程序引导，片内引导程序的流程图如下：



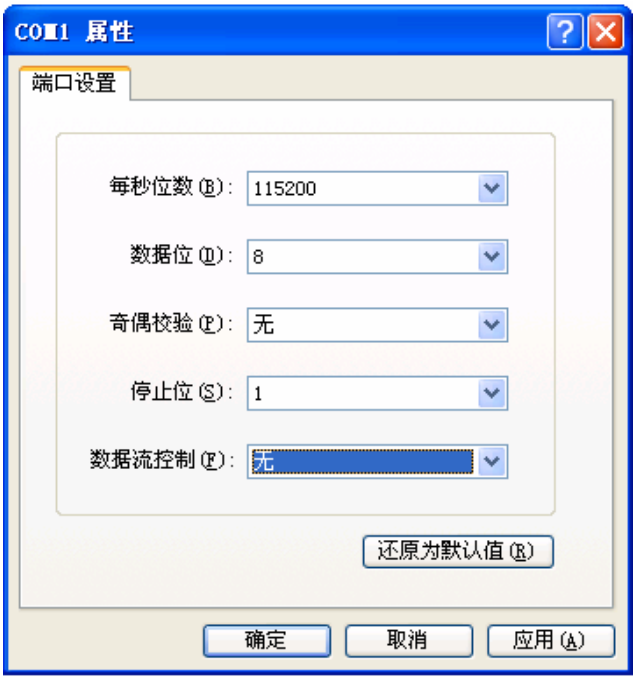
要从内部启动，必须将 JP6 的跳线的位置**设置在 1+2 的位置（丝印是 INTER BOOT）。**

外部启动：主要是从外部 NOR FLASH 启动，要从外部启动，必须将 JP6 的跳线的位置**设置在 2+3 的位置（丝印是 EXTER BOOT）。**

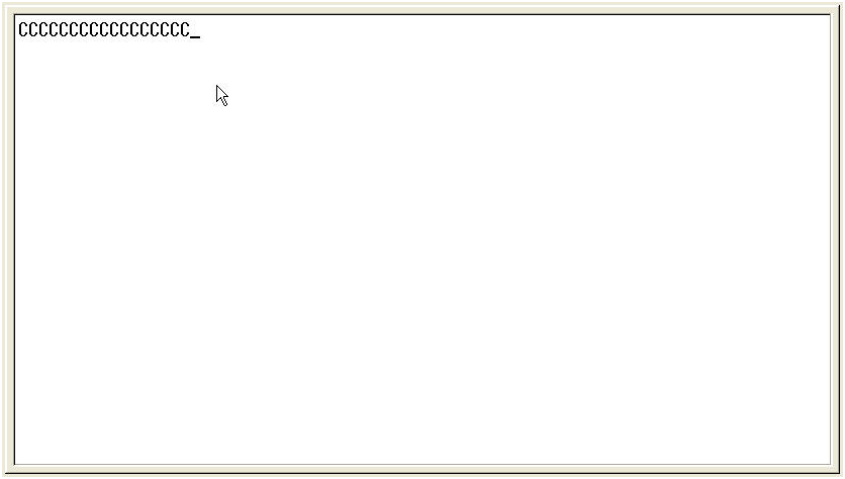
4.1 片内启动程序下载调试

这一章节，工作在内部启动模式下。

用交叉串口线将开发板的 P2 串口和 PC 机的串口连接起来，**将 JP6 的跳线位置设置在 1+2** 的地方，选择从**内部启动**，同时断开开发板上的 JP1 跳线，选择从串口下载程序，打开超级终端，新建一个串口连接，选择连接开发板的串口，进行如下设置



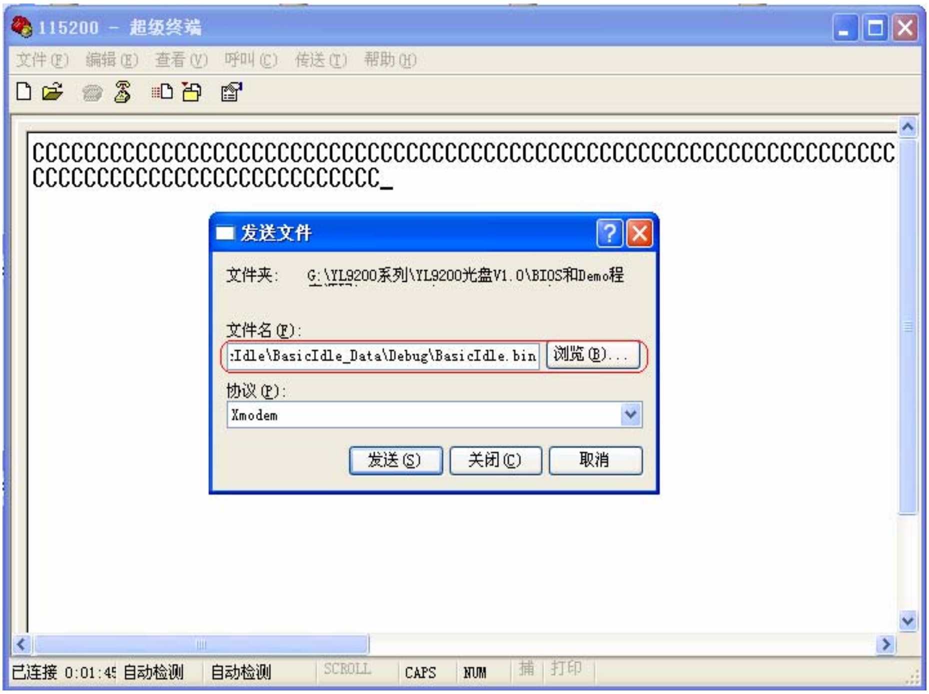
按确定后，再按连接图标。打开开发板电源，然后超级终端会不断显示 C 字符，



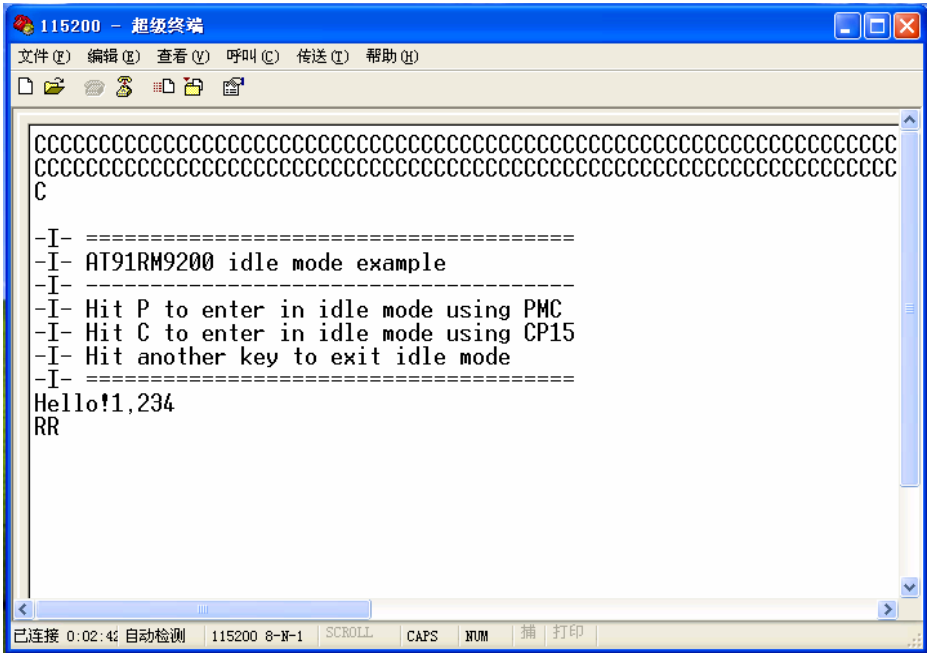
这是 AT91RM9200 片内启动程序在没有检查到任何可用外部存储器带有效启动程序的情况下运行的 XMODEM 方式下载程序，我们可以在超级终端里选择发送一个调试程序到开发板上运行。

这里运行一个简单的 IDLE 的程序，程序的位置：

\光盘\BIOS 和 Demo 程序源码\BasicIdle\BasicIdle_Data\Debug\BasicIdle.bin



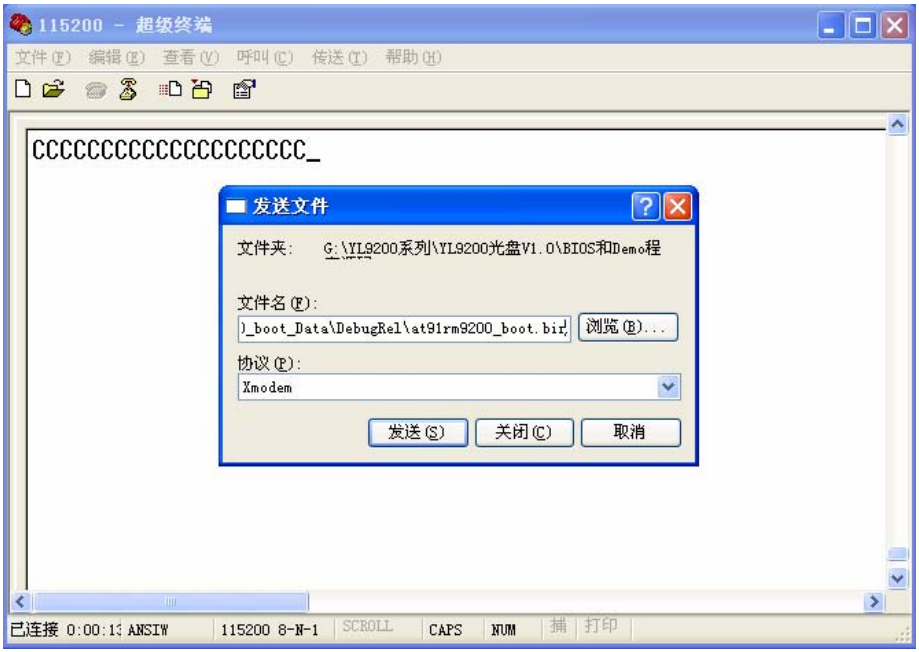
在这里我们选择一个 BasicIdle 的程序进行调试，下载完成后程序自动运行，运行情况如下。



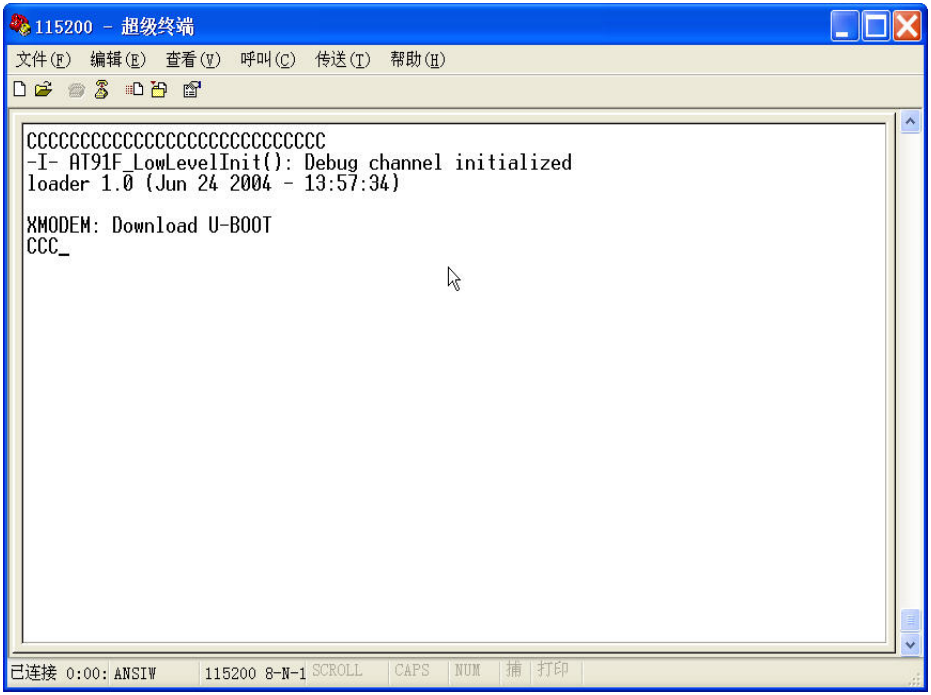
注意：AT91RM9200 的片内启动程序是将调试程序下载到片内 RAM 运行的，其起始地址是 0x200000，大下为 16K，除去 3-4K 片内启动程序占用的部分数据空间，可用空间大概在 12-13K 之间，因此下载调试程序的大小也限制在这么大。另外片内启动程序没有初始化 SDRAM，如果用户编译的调试程序要使用 SDRAM，则要在程序里做相应的初始化，如果程序运行起始地址不在 0x200000，还要在起始代码段做相应的代码搬运工作。

4.2 如何下载运行更大的程序

当要下载调试的程序大于 12K 时，用片内启动程序下载运行已不够用了，需要启动一个二级下载程序 AT91RM9200_BOOT。如 1 步所述，在发送文件时选择 AT91RM9200_BOOT.BIN 进行发送。


115200 - 超级终端
文件(F) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
CCCCCCCCCCCCCCCCCCC_
发送文件
文件夹: G:\YL9200系列\YL9200光盘V1.0\BIOS和Demo程
文件名(F):
)_boot_Data\DebugRel\at91rm9200_boot.bir
浏览(B)...
协议(P):
Xmodem
发送(S)
关闭(C)
取消
已连接 0:00:1
ANSIW
115200 8-N-1
SCROLL
CAPS
NUM
捕
打印

下载完成后运行如下



此程序对 SDRAM 作了初始化，同样运行 XMODEM 下载，下载地址在 SDRAM 的 0X21F00000 处，运行下载程序前初始化内部时钟为 180MHz。

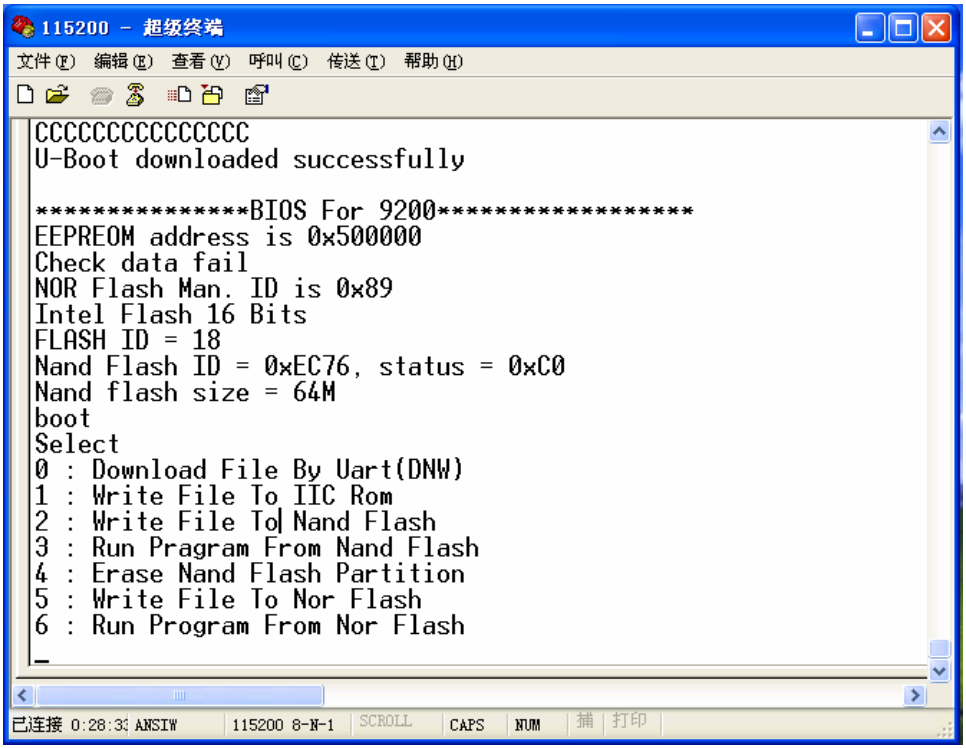
4.3 下载运行 BIOS

注：IIC 启动程序在出售前已烧入开发板上的 AT24C128 芯片内，BIOSBOX 也已烧入 NAND FLASH。

如上步所述，断开开发板上的 JP1 跳线，复位开发板，在超级终端显示出 C 字符后，再将 JP1 跳线连上，这时在超级终端选择 BIOS9200.BIN 进行发送(XMODEM,同 3.3 章节传输文件一样)，bios9200.bin 的位置：

\光盘\BIOS 和 Demo 程序源码\bios9200\bios9200_Data\DebugRel\bios9200.bin

点击文件传送后，接着一直按着开发板上的 S2~S5 中的任意一个键，（因为不按键的话，下载的程序运行后，将会启动 NAND FLASH 里面已固化好的程序），
下载结束后，会自动运行，运行的情况如下



当出现上面的界面后，可以松开按键了。

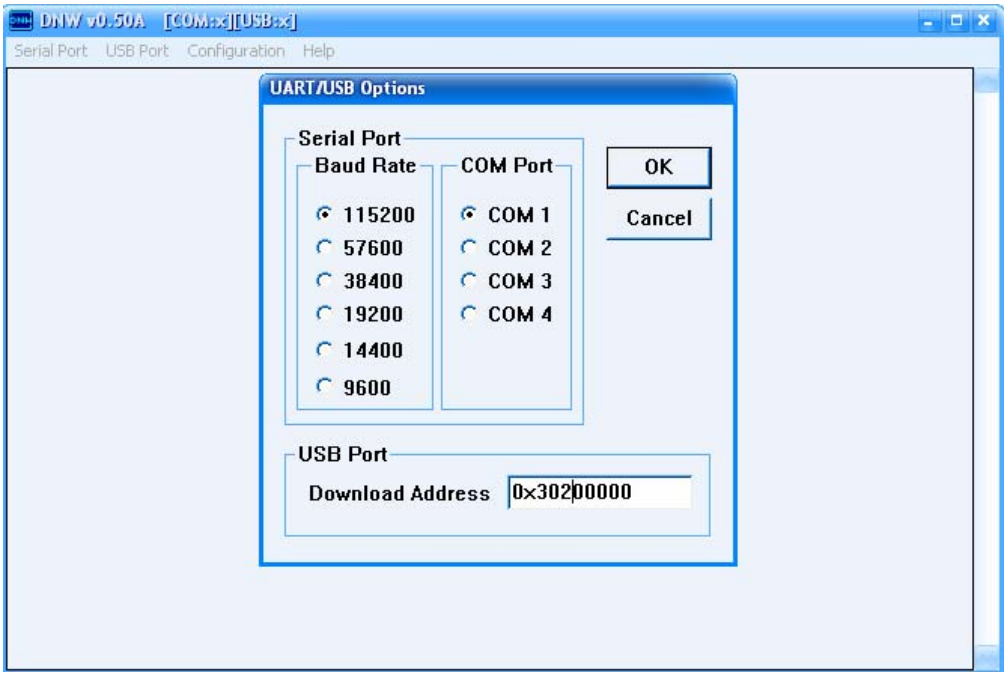
其中 DNW 方式下载是要用 DNW 程序的串口下载方式的，我们可以通过这种方式下载启动程序到 SDRAM 中，再烧写到 IIC ROM 里面去。

注意：下载运行 BIOS9200.BIN 前要保证先连接 JP1。

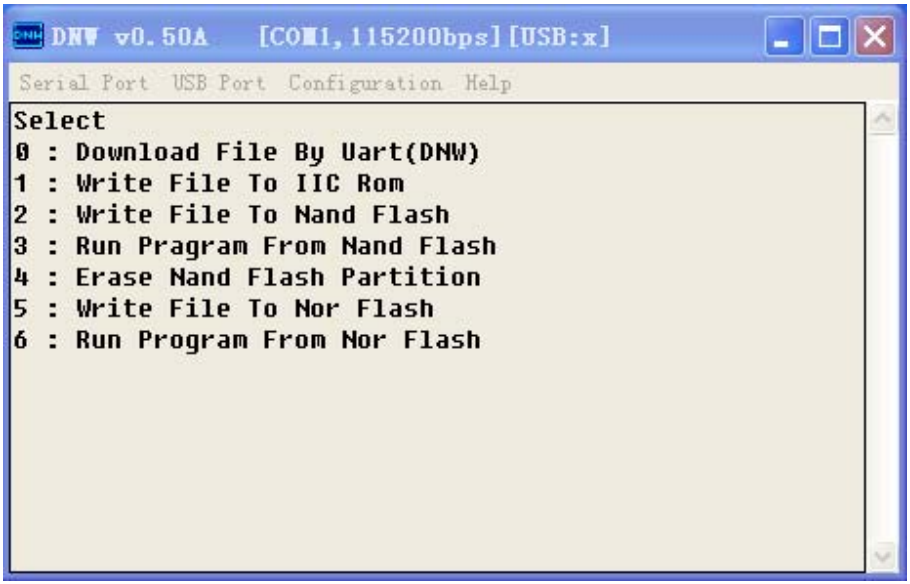
4.4 通过 BIOS9200 下载自己并将其烧写到 IIC ROM

先在超级终端选择断开连接，然后再打开 DNW 程序，在 Configuration 菜单下选择

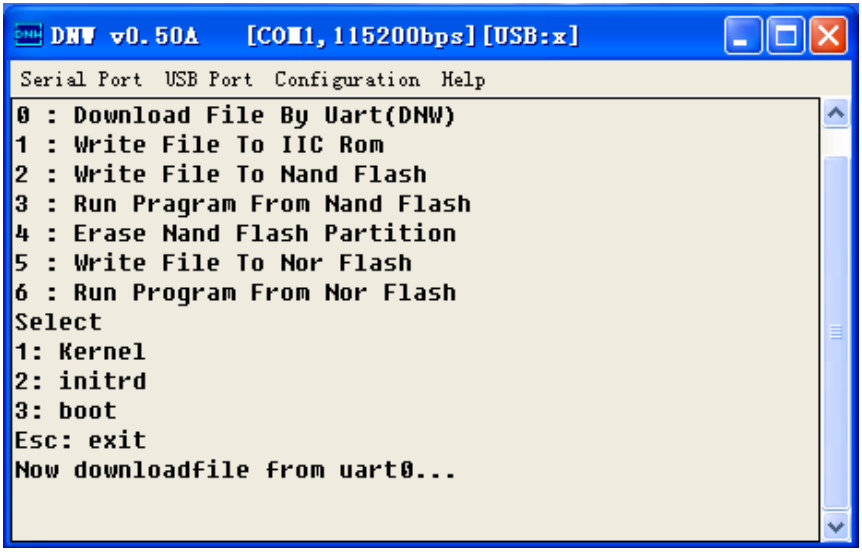
Options，作如下设置



按 OK 后再在 Serial Port 菜单下选择 Connect，敲一下回车键，显示如下

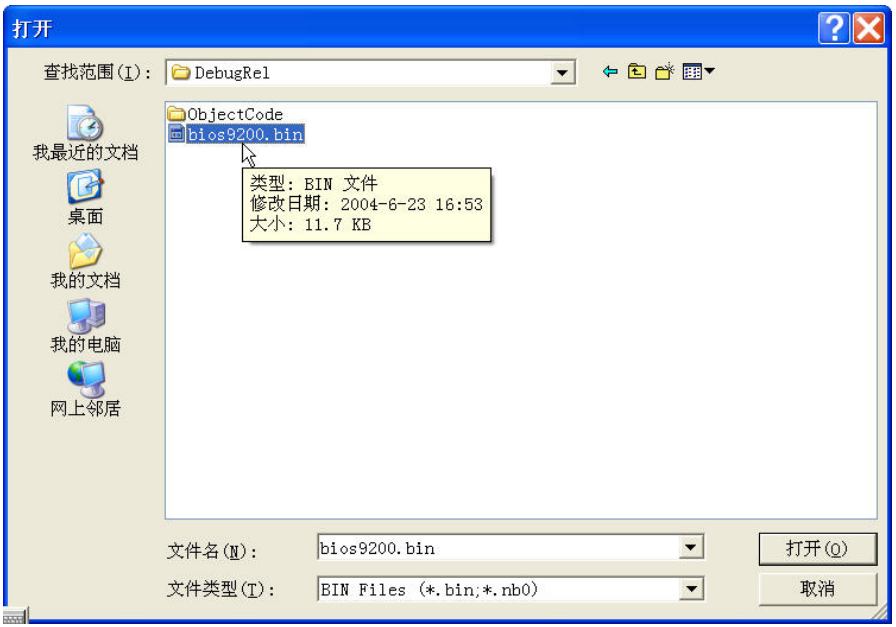


选择 0 : Download File By Uart(DNW)，再选择 2: initrd

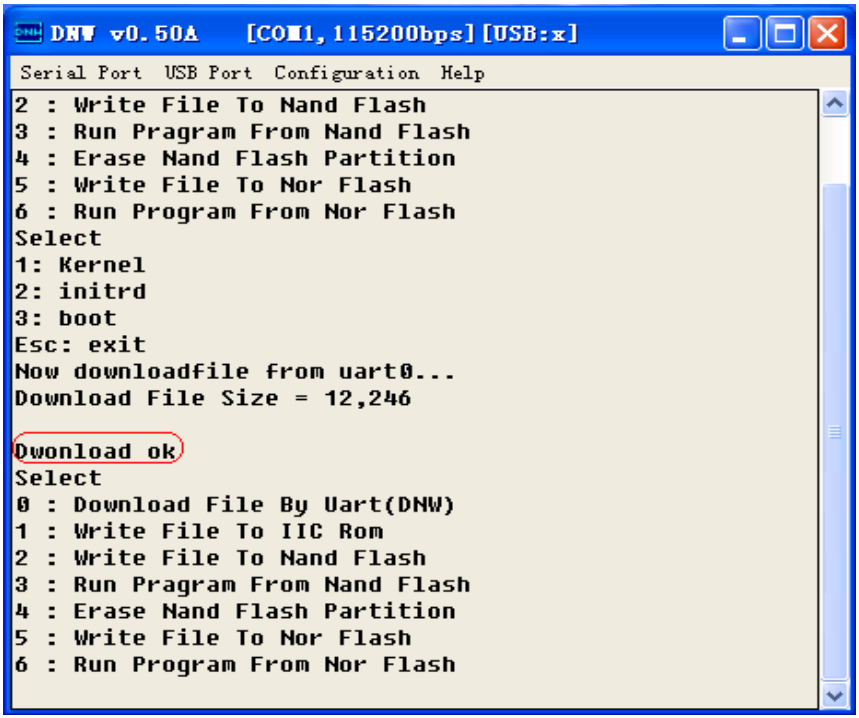


此时可见开发板上 4 个 LED 闪烁，再在 Serial Port 菜单下选择 Transmit，出现文件选择

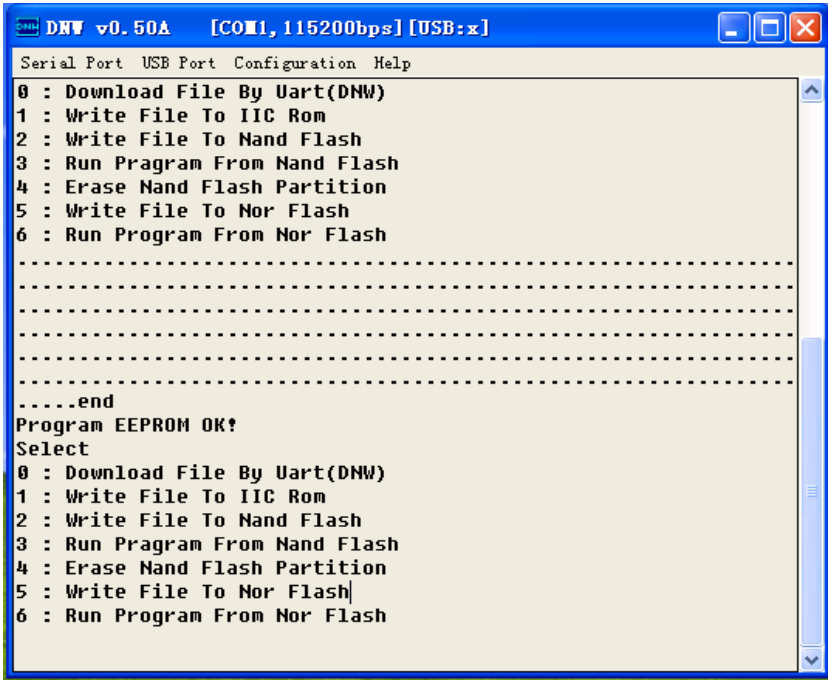
菜单，



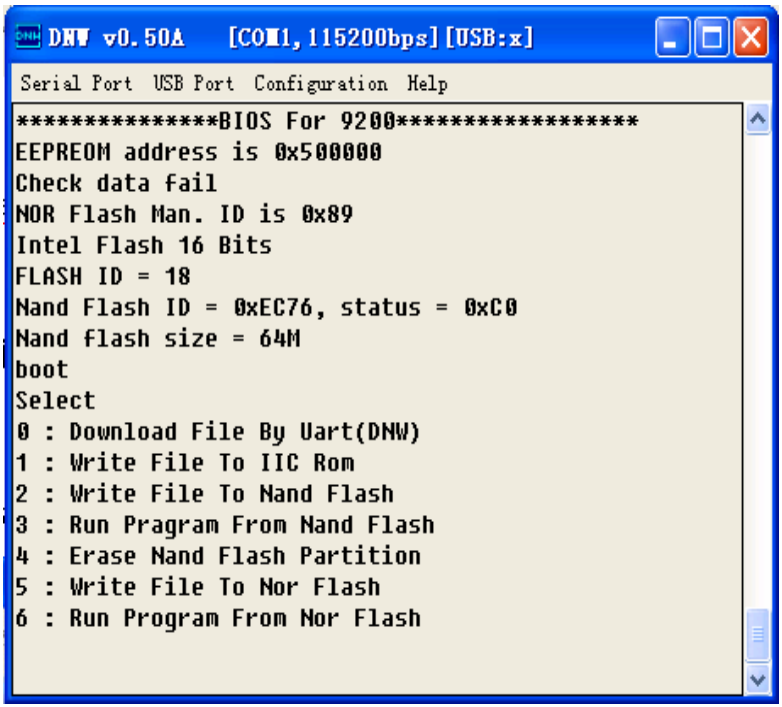
选择 DebugRel 版本的 bios9200.bin 发送，接收成功显示如下



再选择 **1** 将下载的程序烧入 IIC ROM。



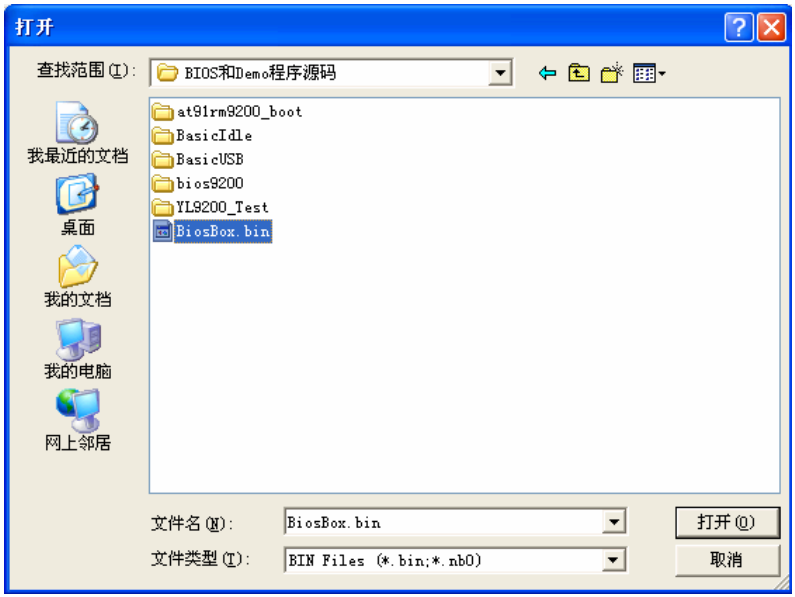
按复位键重启开发板后，可见到 IIC 启动情况



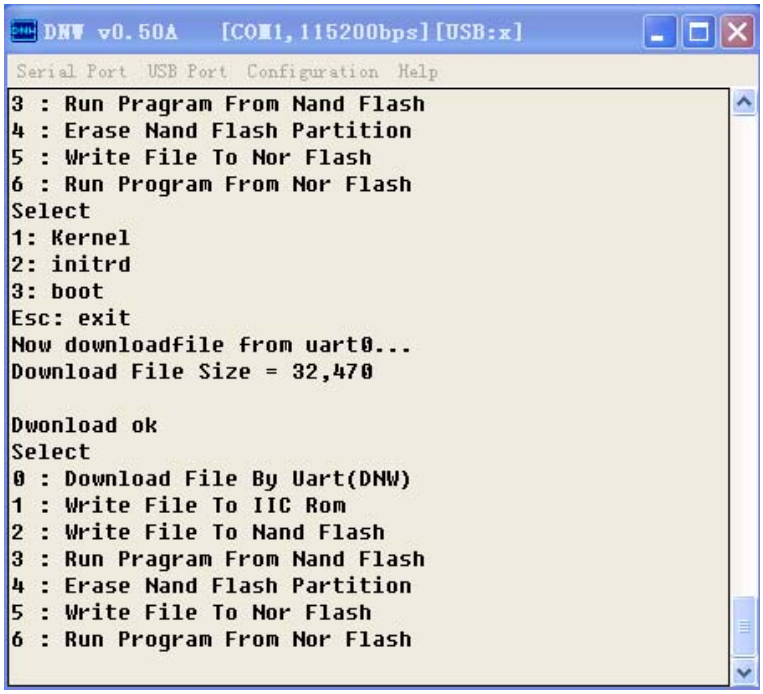
注：在 DebugRel 版的 bios9200 里有 NAND FLASH 支持，Bios 中将 NAND FLASH 的最低 192K 空间视为启动区，Bios 启动后检测到 NAND FLASH 的第一个 PAGE 是否有“boot”标记，有的话会读入此 192K 空间的内容直接运行，用户在编译时可修改源程序（nand.c 中的 FindNandBoot 函数）屏蔽此功能。

4.5 利用 bios9200.bin 将 BIOSBOX.BIN 烧写到 NAND BOOT 分区

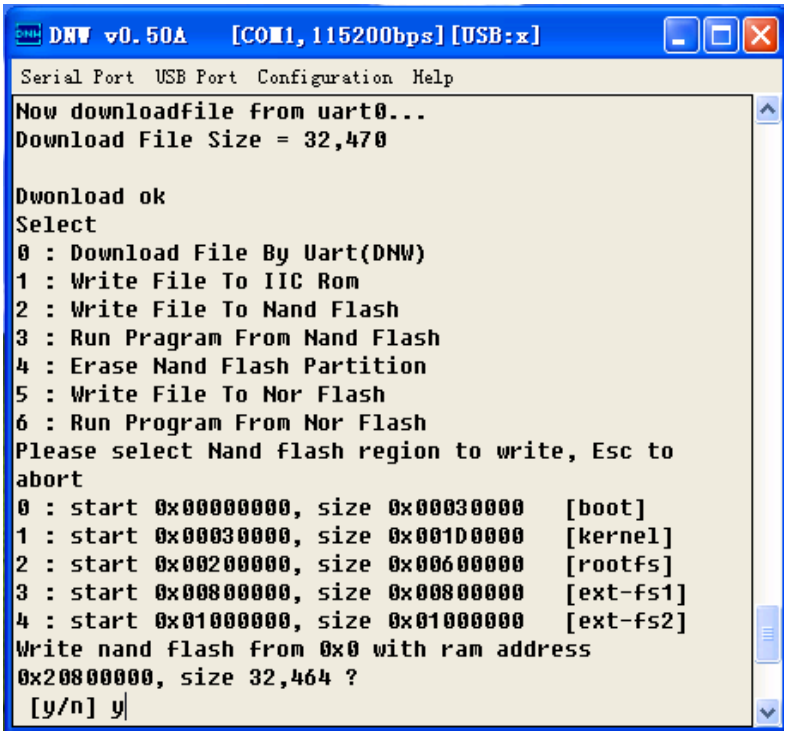
用 IIC ROM 启动后，选择选择 **0 : Download File By Uart(DNW)**，再选择 **2: initrd**，在 DNW 的 Serial Port 菜单下选择 Transmit，在文件选择对话框中选择 BiosBox.bin 文件发送


打开
查找范围(I):
BIOS和Demo程序源码
我最近的文档
桌面
我的文档
我的电脑
网上邻居
at91rm9200_boot
BasicIdle
BasicUSB
bios9200
YL9200_Test
BiosBox.bin
文件名(N):
BiosBox.bin
打开(O)
文件类型(T):
BIN Files (*.bin;*.nb0)
取消

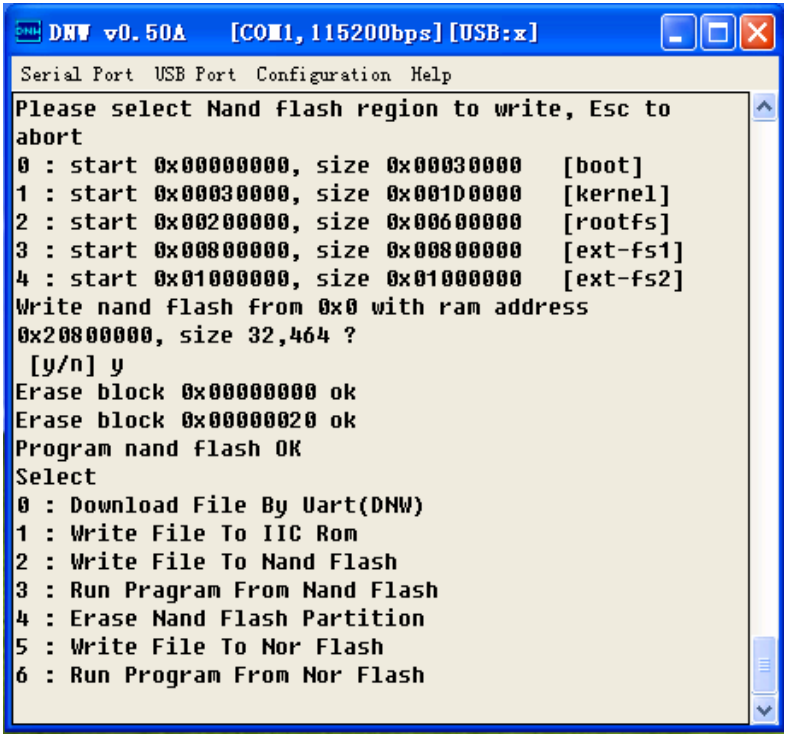
接收成功


DNW v0.50A [COM1,115200bps][USB:x]
Serial Port USB Port Configuration Help
3 : Run Pragram From Nand Flash
4 : Erase Nand Flash Partition
5 : Write File To Nor Flash
6 : Run Program From Nor Flash
Select
1: Kernel
2: initrd
3: boot
Esc: exit
Now downloadfile from uart0...
Download File Size = 32,470
Dwonload ok
Select
0 : Download File By Uart(DNW)
1 : Write File To IIC Rom
2 : Write File To Nand Flash
3 : Run Pragram From Nand Flash
4 : Erase Nand Flash Partition
5 : Write File To Nor Flash
6 : Run Program From Nor Flash

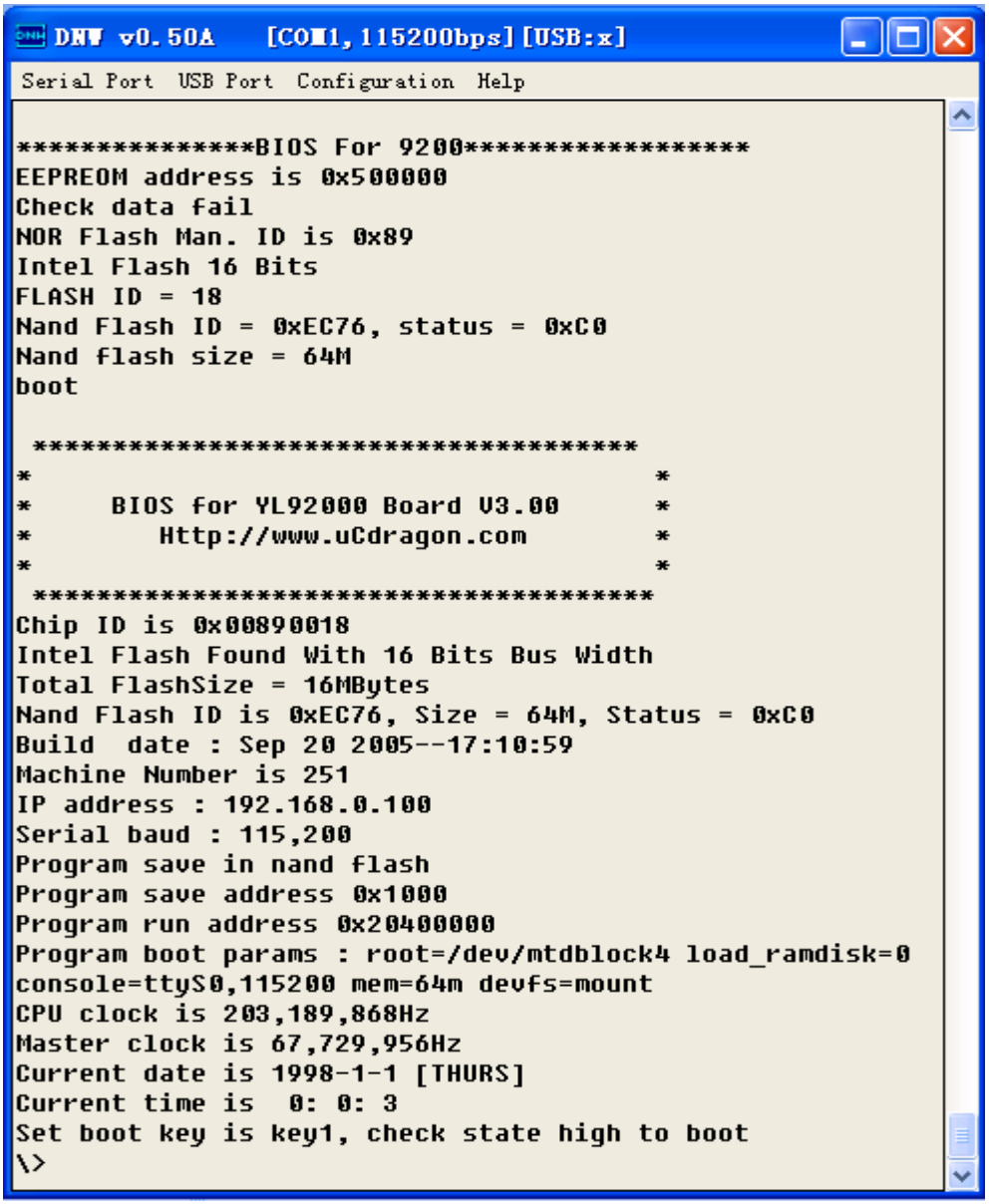
再选择 **2 : Write File To Nand Flash**，再选择 nand flash 分区 0



按 y 确认，会显示烧写成功。



按下开发板复位键，系统重启后，如果在 IIC 启动程序里允许了 NANDBOOT 的功能，会自动运行 NAND FLASH 中的程序。



关于 BIOSBOX 的使用见 BIOSBOX 功能说明。

4.6 利用 BIOSBOX 将程序烧写到 Nor Flash

这一章节，主要利用 BIOSBOX 将 BIOSBOX.bin 烧写到 Nor Flash 中，

（1） 进入 BIOS 状态，输入命令：

comload ;

然后回车。

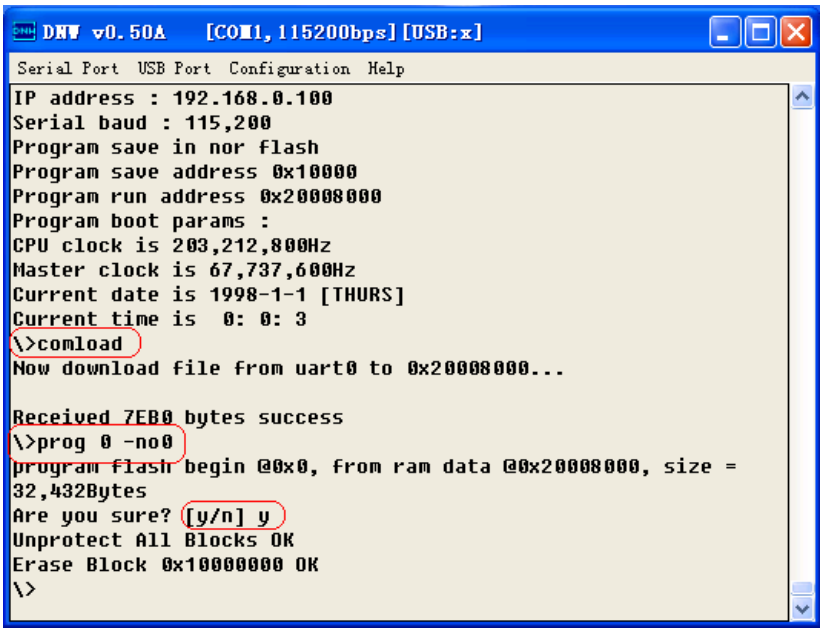
接着选择要传输的文件 BIOSBOX.bin。

文件传输结束后，会自动返回到 BIOS 的命令状态下。

（2）接着，输入命令：

prog 0 –no0 ;这个命令将刚刚下载的 BIOXBOX.bin 烧写到 Nor Flash 中。

在接着出现的[y/n]下，选择”y”，这样就可以将程序烧写到 Nor Flash 中了。



（3）选择外部启动模式

选择外部启动模式，这时将启动刚刚烧写到 Nor Flash 中的 BIOSBOX 程序，要从外部启动，必须将 JP6 的跳线的位置设置在 **2+3（EXTER BOOT）**的位置。

这时在上电复位，按复位键，将工作在外部启动模式下，从 Nor Flash 中启动程序。这时程序 ，也是跑的 BIOS，在这个启动模式下，可以烧写 NAND FLASH.

4.7 BIOS 的功能说明

（注：以下命令所带参数中地址和长度都属 16 进制，不必在前面加 0x）

help 和**?**可以列出所有命令并给出简单的说明；

date 命令可以显示和设置当前日期，只输入 date 命令则显示日期，输入 date 2004-6-8 则设置当前日期为 2004 年 6 月 8 日。

time 命令可以显示和设置当前时间，只输入 time 命令则显示时间，输入 date 14：4：30 则设置当前时间为 14：4：30。

setweek n 可设置星期几，n 从 1 到 7 表示星期一到星期日。

clock 可以显示当前的工作频率。

setmclk 可以改变 CPU 工作频率，具体参数设置可见芯片手册，注意不要使频率超出工作范围。频率参数属于可以保存和调入的参数，这次设置和保存后下次复位 BIOS 会自动调入这写参数初始化 CPU。

setbaud 可改控制串口的波特率，改完后要在 PC 上相应改变串口通讯波特率后再敲回车。

ipcfg 可显示和修改 TFTP 下载时所用的 IP 地址，只输入 IPCFG 则显示当前 IP 地址，输入 IPCFG192.168.2.223 则将 IP 地址改为 192.168.2.223。

netload 启动 TFTP 接收，若没带地址参数，则使用缺省下载地址 0x0c008000,若指定地址，下载数据保存到指定地址开始的 SDRAM 中去，如 netload c300000。启动 tftp 接收后，要在 PC 端执行 tftp 下载程序，在 WIN2000 或 WINXP 下，直接输入 tftp -i xxx.xxx.xxx.xxx put 文件名即可；在 WIN98 下，使用 CDROM 里所带的 TFTP 程序；在 LINUX 下，使用 CDROM 里所带的 TFTPMD 程序。注意进行 TFTP 传输时要保证 PC 机和开发板处于同一个 IP 段内。

netrun 或者是 **g** 启动 TFTP 接收完数据后会自动运行下载到的程序，缺省下载地址和指

定参数同 netload。

comload 启动串口下载（DNW 程序的串口下载），缺省下载地址和指定参数同 netload.

comrun 启动串口下载（DNW 程序的串口下载）并在接收完数据后自动运行下载的程序，缺省下载地址和指定参数同 netload。

rx 启动 XMODEM 方式下载，可在超级终端内选择 1K XMODEM 或 XMODEM 发送数据到开发板上，缺省下载地址和指定参数同 netload。

rxrun 在启动 XMODEM 方式接收完数据后自动运行下载到的程序，缺省下载地址和指定参数同 netload。

prog 可以烧写 NOR FLASH，目前支持 SST39VF160。prog 命令完整的参数是 prog addr1 addr2 length [-no0],其中 addr1 是要烧写的 FLASH 的地址，大于等于 0，小于 200000，字对齐，addr2 是 sdram 中要烧进 flash 的数据区起始地址，length 是要烧写的长度，-no0 表示要把数据烧进 Nor Flash 0 地址开始的地方时，是否修改 0 地址的指令，因为 CPU 复位总是从 0 开始执行的，当用 Nor Flash 启动时，若用 prog 命令将下载到的程序烧入 Nor Flash 0 地址开始的地方并在命令最后指定-no0,那么在复位后，就不会再运行 Bios 而直接启动用户程序了，若不在 prog 命令最后加-no0, 则 BIOS 可以烧写 NOR FLASH 0 地址的数据前，将 0 地址的指令改为直接跳转到 0x1f0000 处即 Bios 的驻留地址，并保存原程序 0 地址将要跳转到的地址，以后在执行 boot 指令时再跳转过去执行用户烧入的程序。运行 BIOS。在运行 BIOS 下载完数据后，也可不带参数直接执行 prog 命令，缺省的 NORFLASH 地址是用户程序存储地址 prog_s_addr（见后面 setpa 命令），sdram 中数据起始地址和数据长度在接收成功后自动设定了。

ap 指令自动下载完数据并将数据烧写到 NORFLASH 的 0 地址处，缺省为 TFTP 下载，指定-c 表示串口下载（DNW 方式），-x 表示 XMODEM 下载，-b 表示不修改 0 地址的指令。

backup 可用在第一次烧写完 BIOS 到 NORFLASH 0 地址后上电执行时将 BIOS 本身拷贝到 0x1f0000 处。

copy 将 NORFLASH 某地址的数据拷贝到另一地址。

boot 可运行用户通过 BIOS 下载烧写到 0 地址并修改过 0 地址跳转地址的程序，见 prog.

run 可运行存储器中的程序，缺省地址就是缺省下载地址，也可指定运行地址。

move addr1 addr2 size 可将存储器中 addr1 开始的长度为 size 的数据拷贝到 addr2 开始的地址去。

mrun 可自动执行 move 的过程并运行程序，比如在 FS44B0 中我们将 UCLINUX 内核保存在 Nor Flash 的 0x10000 开始的地方，长度为 800K，它的运行地址是 0x0c300000,那么 MRUN 就可以完成拷贝的操作并直接运行。MRUN 内部使用的参数见 setpa 命令。

md 显示存储器中的数据，可以带地址参数。

memd 可显示单个存储器单元中的内容，-c 参数表示 8 位数据，-s 参数表示 16 位数据,-l 参数表示 32 位数据，后面跟存储器地址。

mems 可修改单个存储器单元中的内容，-c,-s,-l 参数同上，后面跟存储器地址和要写入的内容。

machine 可设置机器号，适用于 LINUX 此参数可保存。

setpa 有几个参数

Usage : setpa -s[-r][-i][-ni][-nor][-nand] [address]

-s save address

-r run address

-i initrd save address

-ni disable initrd

-nor use nor flash to save

-nand use nand flash to save

其中-s 表示用户程序在 FLASH 中的存储地址，如上面所说的将 UCLINUX 内核保存到 Nor Flash 的 0x10000 处，为使 mrun 正确运行，我们就要设置 setpa -s 10000

-r 表示用户程序的运行地址，如上面所说的将 UCLINUX 内核的运行地址是 0x0c300000,为使 mrun 正确运行，我们就要设置 setpa -r c300000
-i 表示使用 initrd(对于 linux 或 uClinux),它的存储地址是多少。
-noi 表示取消 initrd。
-nor 表示用户程序存储在 Nor Flash 中，-nand 表示用户程序存储在 Nand Flash 中。注意使用 Nand Flash 存储时，前述保存地址 1000 表示 Nand 分区 1，2000 表示 Nand 分区 2，依此类推，Nand 分区见 nfpart 命令。
setpa 设置的参数都是可以保存的。

setbp 可以设置启动命令（对于 UCLINUX 和 LINUX）,可以保存。
Usage : setpa -s[-r][-i][-ni][-nor][-nand] [address]
-s save address
-r run address
-i initrd save address
-ni disable initrd
-nor use nor flash to save
-nand use nand flash to save

-s 表示 mrun 运行的程序是存储器在 flash 的什么位置,
对于 nor flash 是 nor flash 中的地址,
对于 nand flash 1000 表示分区 0,2000 表示分区 2.
-r 表示存储的程序要读到 sdram 中什么位置再运行.对于 uClinux 是 c300000.
-i 表示 initrd 存储在 flash 的什么位置,如同-s.
-ni 表示取消 initrd
-nand 表示用 NAND FLASH 作为内核及 initrd 的存储介质.

bootkey 可设置 BIOS 复位运行后检查哪个按键状态来自动启动存储在 Flash 中的用户程序，即自动调用 mrun 指令，按键编号 1~4，状态 0 表示低启动，1 表示高启动。比如要在复为后检测到按键 3 为低时启动，可执行 bootkey 3 0。此参数也可保存，注意实现自动启动的前提是先烧写好 Flash 和用 setpa 命令设置好各个参数，bootkey 命令最后可带-b 参数，表示自动运行 boot 指令，缺省情况下是运行 mrun 指令。

nfpart 可在 Bios 中对 Nand Flash 简单分区，比如 Nand Flash 大小是 32M，要分为

0~0x30000,0x30000~0x200000,0x200000~0x800000,0x800000~0x1000000,0x1000000~0x2000000 这样 5 个分区，可以执行 nfpart 30000 200000 800000 1000000 2000000，分区最多为 8 个，分区参数可以保存。

nferase 可以擦除 NAND FLASH 分区，块有错误时会有提示。

nfprog 可以将下载的数据写入 NAND FLASH 分区,也可指定烧入数据的起始地址和长度，烧写有错误也会有提示。

nfload 可以将 NAND FLASH 分区的数据全部读入 sdram 中，可以指定 sdram 地址和 NAND FLASH 分区。

senv 命令可以保存所有保存的参数到 FLASH 中，下次复位运行 BIOS 后会调入这些参数。

defset 命令用来设置一些默认参数，比如 NAND FLASH 的分区，Linux 的启动参数等等之类的。

注意：有些命令不能在 YL9200 上使用，比如 usbload 之类的。

第五章 烧写和启动 Linux

这一章节，主要介绍如何烧写 Linux 内核和 Linux 的根文件系统，主要是利用 BIOS 来进行烧写。

串口使用 DNW 工具。

5.1 通过 BIOS 烧写 Linux 内核

（1）进入 BIOS 后，输入命令：

defset ;主要用来设置默认参数

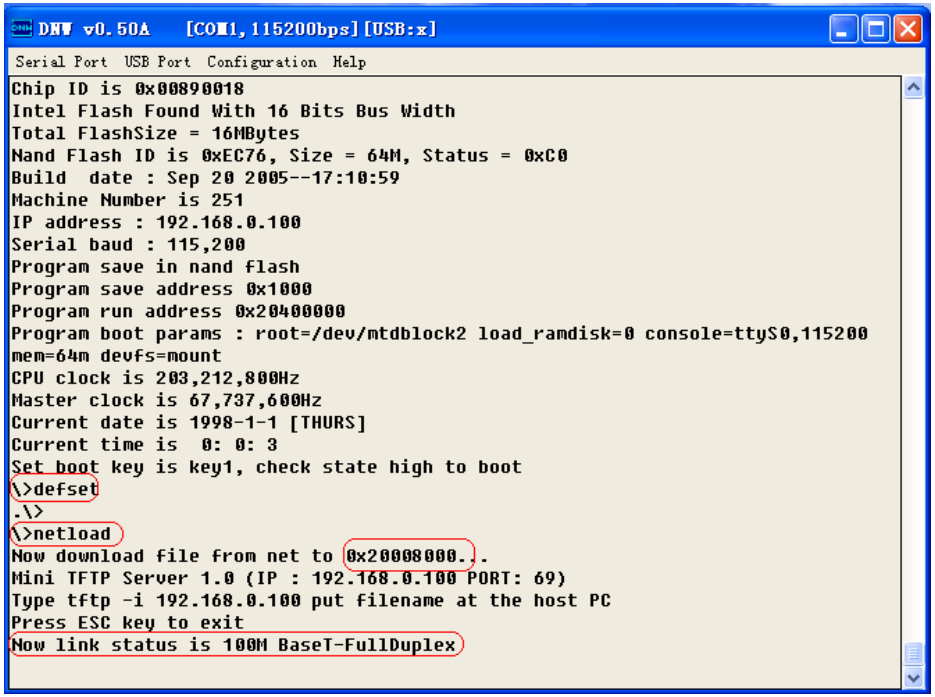
接着连好交叉网线，PC 机的地址要与开发板的地址在同一网段，（开发板的 IP 地址：192.168.0.100，PC 机的 IP 地址：192.168.0.7，当然你可以设置成其他的，最好与我们这里设置一样）。

（2）接着输入命令：

netload ;启动 TFTP 来传输 Linux 内核文件

然后回车。

输入上面命令后，信息如下：

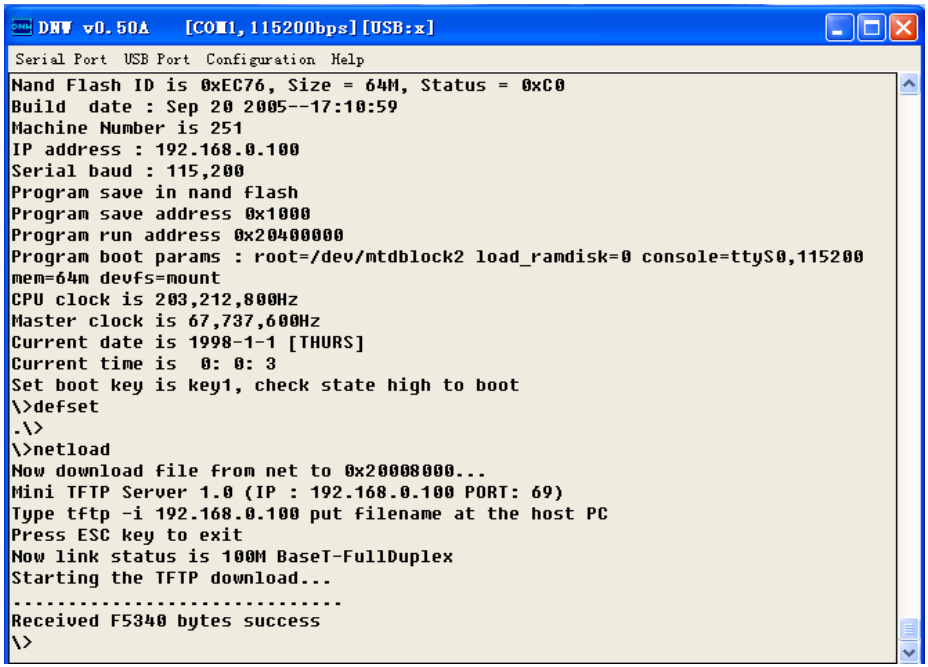


上面的信息显示，程序将下载到 0x20008000 地址处（可以在 netload *****，来改变下载的地址）。IP 地址为 192.168.0.100。

（3）点击光盘下的“目标代码”文件夹下的 zImage.bat 批处理文件，

注意：在进行 TFTP 传输操作前，要将防火墙关闭掉

点击 zImage.bat 批处理文件后，将会启动 TFTP 文件传输，传输成功后，将自动返回到 BIOS 的命令状态下。



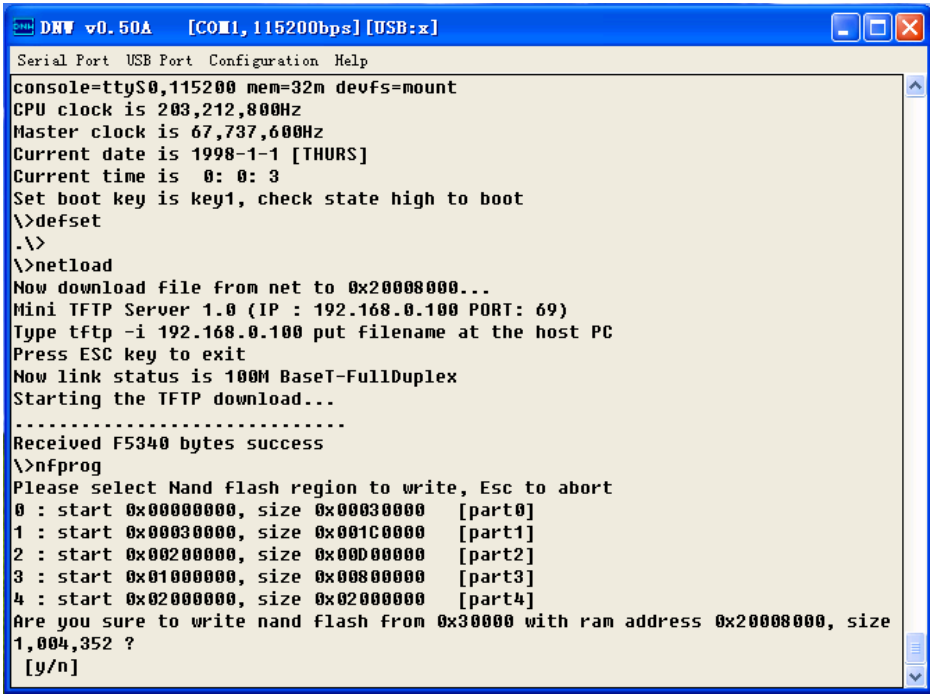
（4）接着，输入命令：

nfprog ;用这个命令将下载的 Linux 内核（可以是其他程序）烧写到 NAND

; FLASH 分区

然后回车。

输入上述命令后，会提示选择要烧写的 NAND FLASH 分区，这时选择“1”，将 Linux 的内核烧写到 NAND FLASH 的分区 1 中。



接着在“[y/n]”提示下输入“y”，烧写结束后，会自动返回到 BIOS 的命令状态下。

5.2 通过 BIOS 烧写根文件系统

这一章节，主要介绍如何烧写根文件系统。

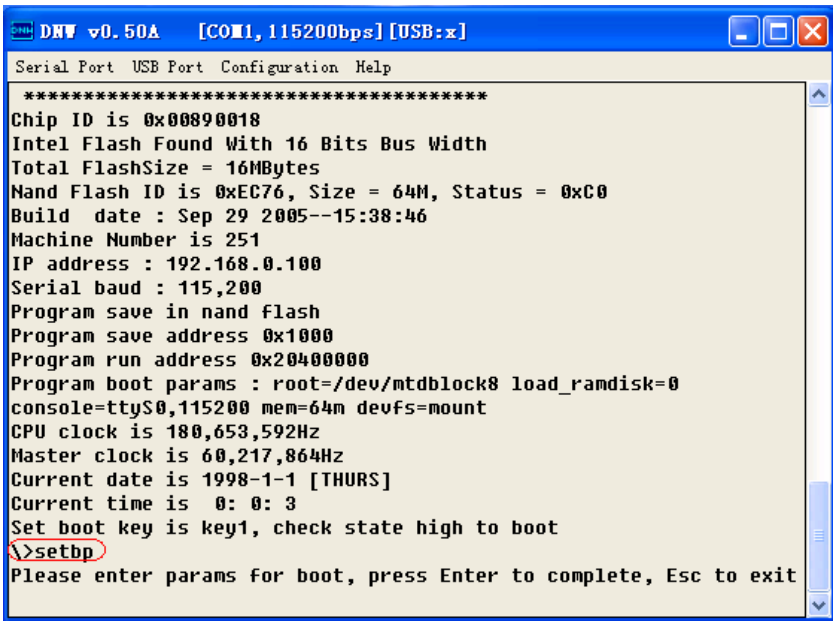
步骤与烧写 Linux 内核是一样的，不同的地方是点击的批处理文件是 cramfs.BAT，烧写的分区是 **2**，其他过程是一样的。**(注意，根文件系统烧写的区域不能有坏块，如果有可以换其他的分区，同时内核的启动参数（/dev/mtdblockX）也需要改动！[X=分区号＋3])**

这样内核和根文件系统烧写好，就可以启动 Linux 了。

设置内核传递参数，命令序列如下：

setbp ;

输入上面这个命令后，会出现如下的界面：

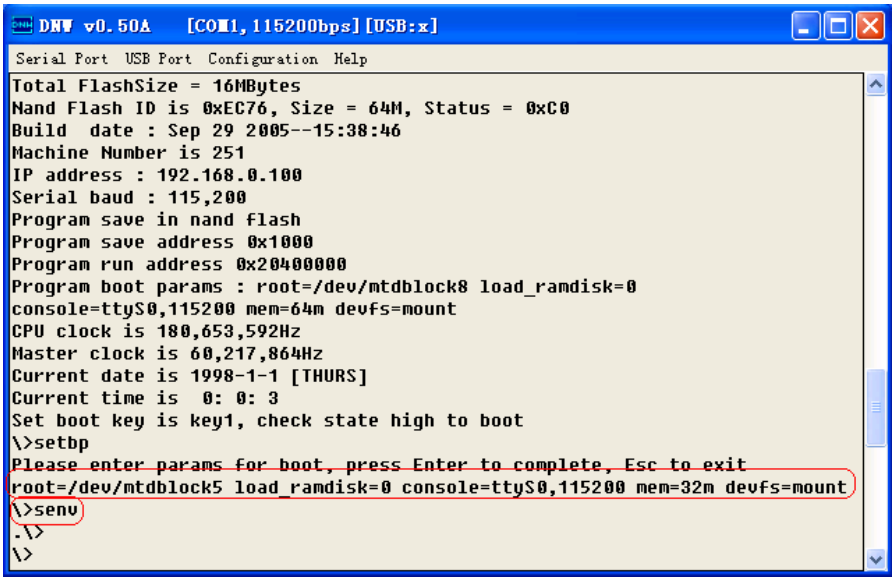


接着输入参数：

root=/dev/mtdblock5 load_ramdisk=0 console=ttyS0,115200 mem=64m devfs=mount

接着输入命令：

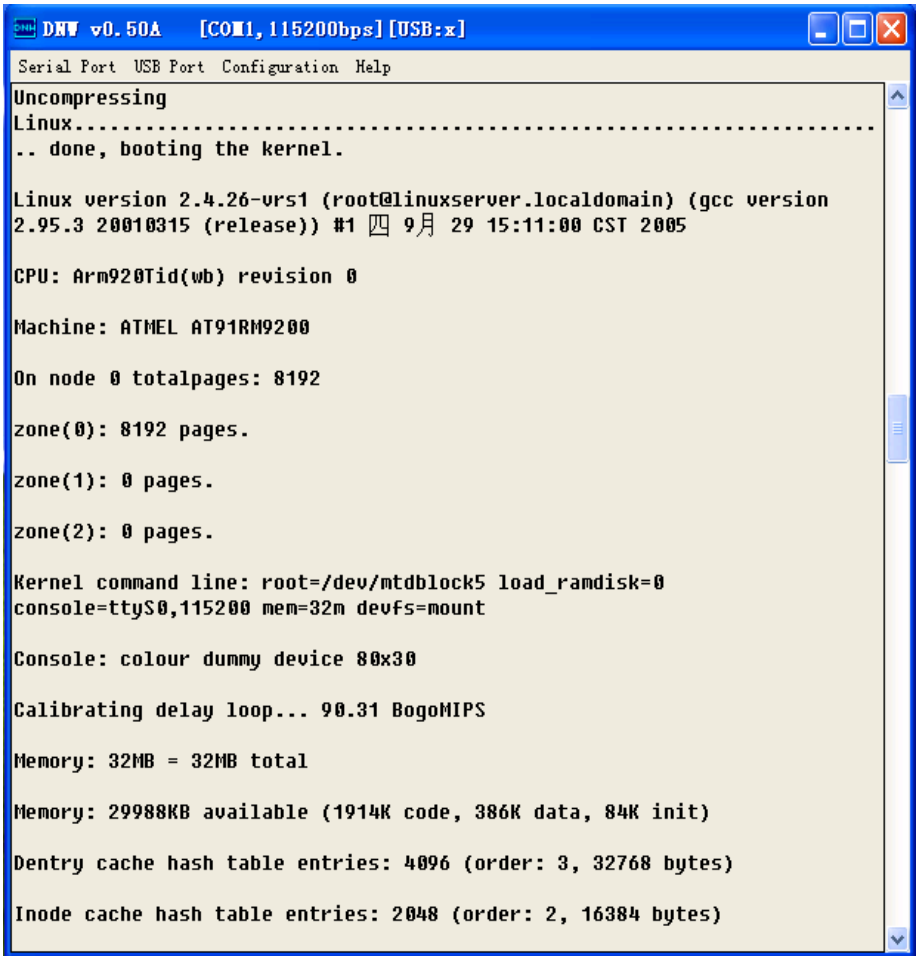
senv ;保存设置的参数



在 BIOS 命令状态下，可以通过输入命令：

mrun ；这个命令可以在 BIOS 下运行 Linux

这时将启动 Linux 操作，启动信息如下：



5.3 设置 Linux 自启动

将 Linux 内核和根文件系统烧写好后，现在来设置 Linux 自启动参数，这样的话，上电就可以自启动 Linux，或者自己的应用程序。

首先进入 BIOS 的命令状态，输入命令：

bootkey 1 1 ；这个命令就可以设置 Linux 的自启动了

接着输入命令：

senv ；保存设置的参数

当上述参数设置好，只要一上电，将会启动 Linux 系统，如果要重新回到 BIOS 的命

令状态，需要上电听到蜂鸣器响一声的时候，按住开发板的 S3 按键。

设置 Linux 自启动后，接上显示器，可以看到显示器的右上角有一个器器企鹅的图标。

说明在 Linux 下就只有支持 frambuffer。

第六章 Linux 内核的编译及根文件系统的制作

注意：这一章节的操作是在 PC 的 Linux 操作系统下进行的
PC 机的 Linux 操作系统是 RED HAT 9.0。

6.1 建立交叉编译环境及编译 Linux 内核

在编译 LINUX 内核前先要在 PC 机的 LINUX 操作系统下安装 ARM 的交叉编译工具。在光盘上提供的编译工具是 cross-2.95.3 的编译器。安装方法是先进入/usr/local 目录，

cd /usr/local

建立一个 arm 的目录，

mkdir arm

再进入 arm 目录，

cd arm

将光盘下的“Linux 源码和工具”下的 cross-2.95.3.tar.bz2 编译工具拷贝到/usr/local/arm 目录（也就是刚刚建立 arm 的目录下）

在/usr/local/arm 目录下执行解压操作，

tar jxvf cross-2.95.3.tar.bz2

解压完成后会在 arm 目录下生成一个 2.95.3 的子目录，编译工具都解压在此目录下。要想把 arm 编译器的路径加在系统命令搜索路径中去，可以编辑/etc/bashrc 文件，在最后加上一行

export PATH=/usr/local/arm/2.95.3/bin:$PATH

安装好编译工具后，在某个分区内解压内核压缩包，比如在/home 分区内。

cd /home

然后将光盘下的“Linux 源码和工具”下的 linux-2.4.26-vrs1-hzh.tar.gz 源码包拷贝到/home 目录下。

然后执行解压命令：

tar zxvf linux-2.4.26-vrs1-hzh.tar.gz

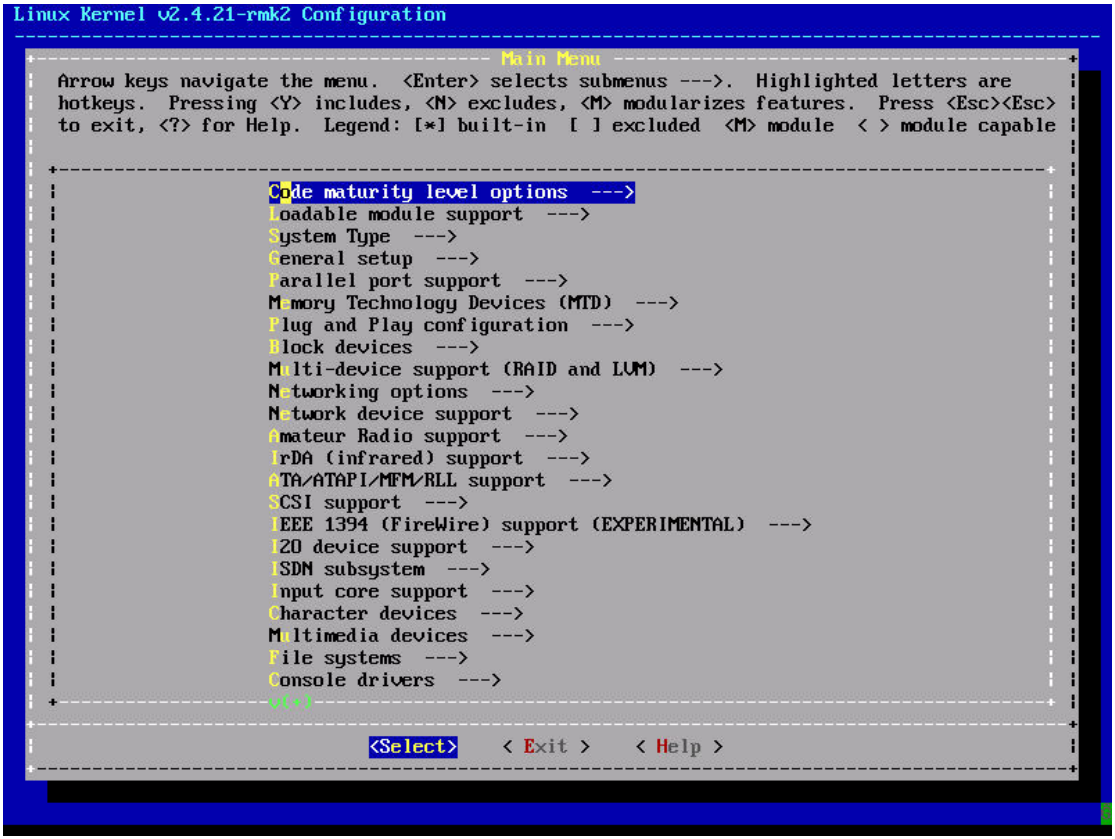
解压完后会在/home 目录下生成 linux-2.4.26-vrs1-hzh 的目录，LINUX 内核都包含在此目录下。再进入此目录

cd linux-2.4.26-vrs1-hzh

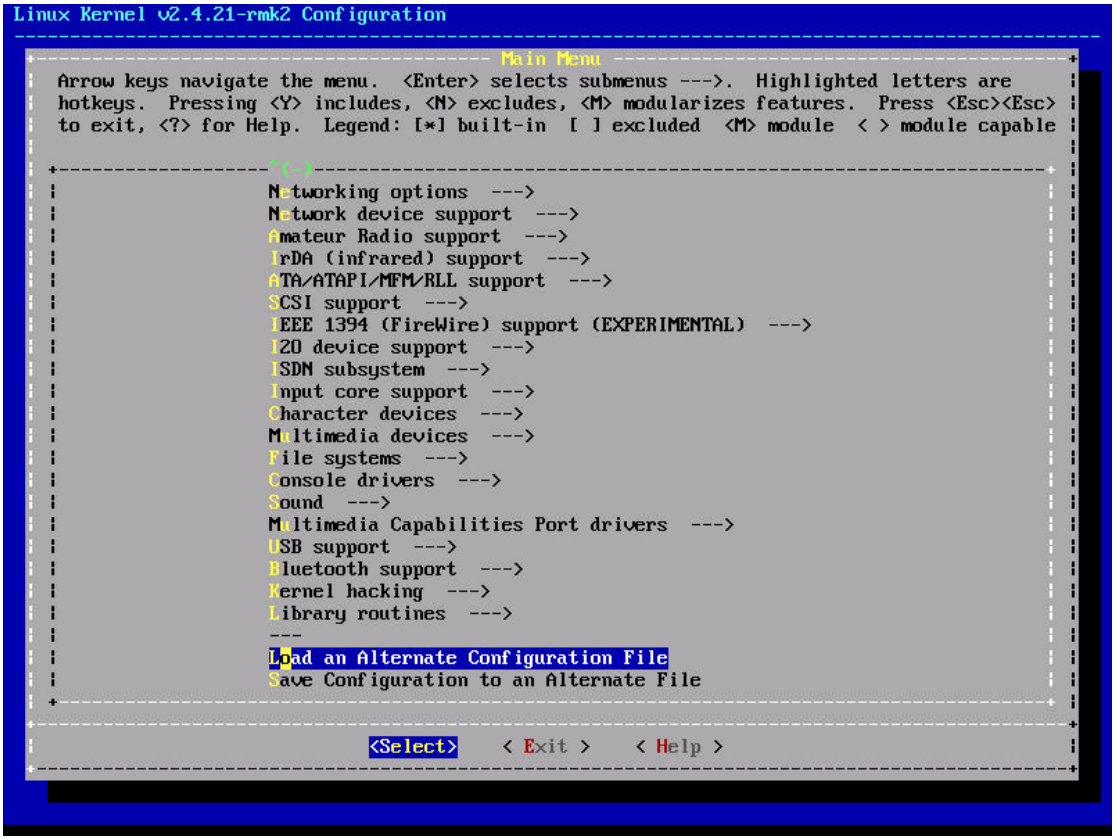
执行

make menuconfig

出现如下图的配置菜单

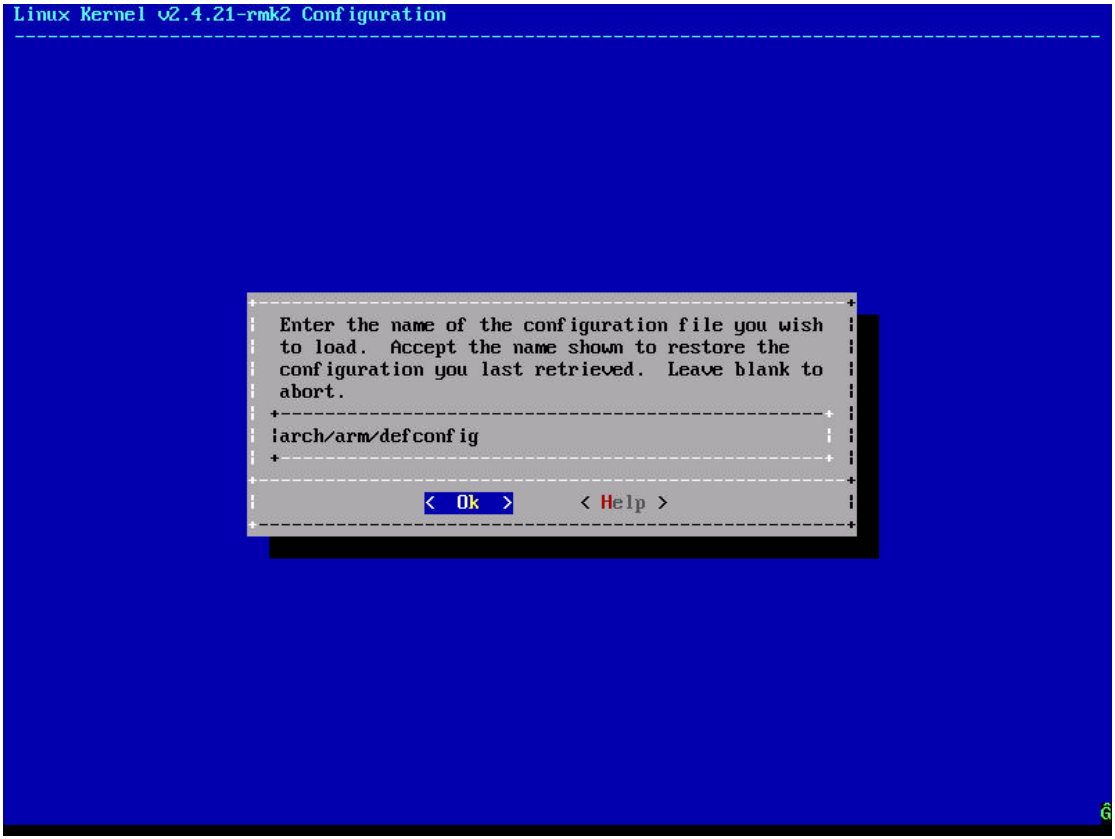


移到最下面，选择 Load an Alternate Configuration File

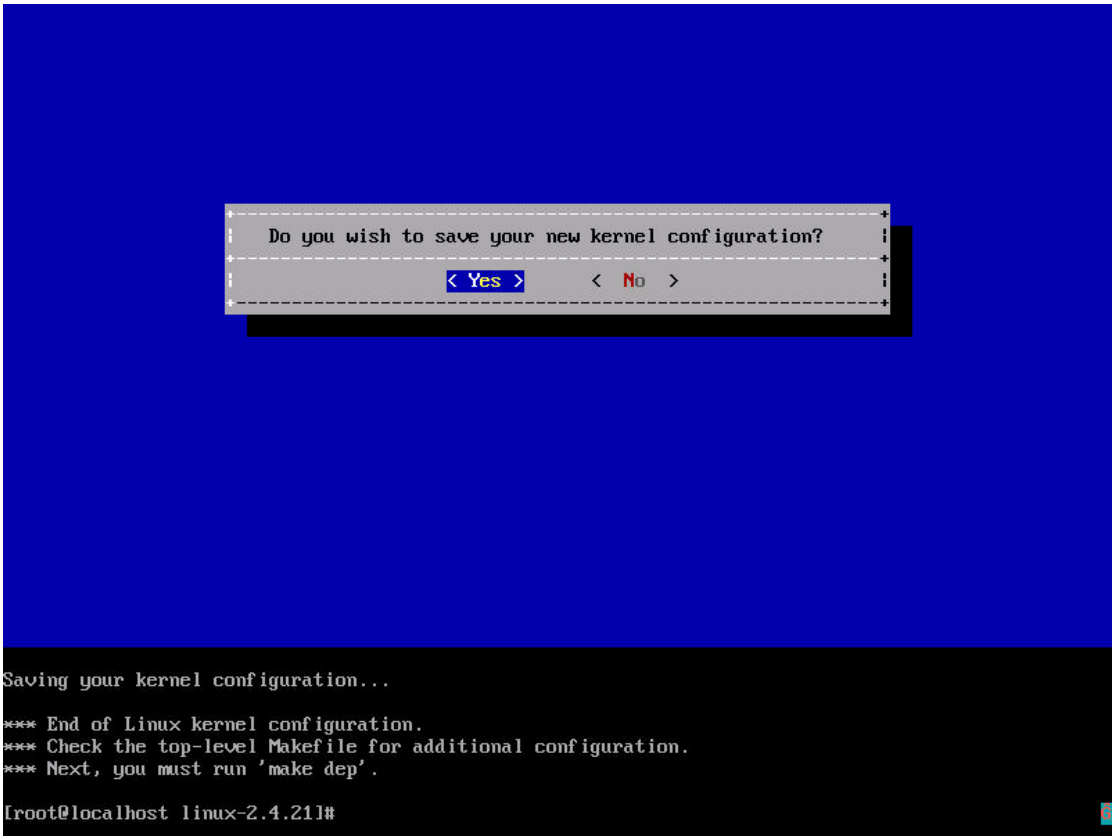

Linux Kernel v2.4.21-rmk2 Configuration
Main Menu
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help. Legend: [*] built-in [ ] excluded <M> module < > module capable
Networking options --->
Network device support --->
Amateur Radio support --->
IrDA (infrared) support --->
ATA/ATAPI/MFM/RLL support --->
SCSI support --->
IEEE 1394 (FireWire) support (EXPERIMENTAL) --->
I2O device support --->
ISDN subsystem --->
Input core support --->
Character devices --->
Multimedia devices --->
File systems --->
Console drivers --->
Sound --->
Multimedia Capabilities Port drivers --->
USB support --->
Bluetooth support --->
Kernel hacking --->
Library routines --->

Load an Alternate Configuration File
Save Configuration to an Alternate File
<Select>
< Exit >
< Help >

出现如下文件名输入框

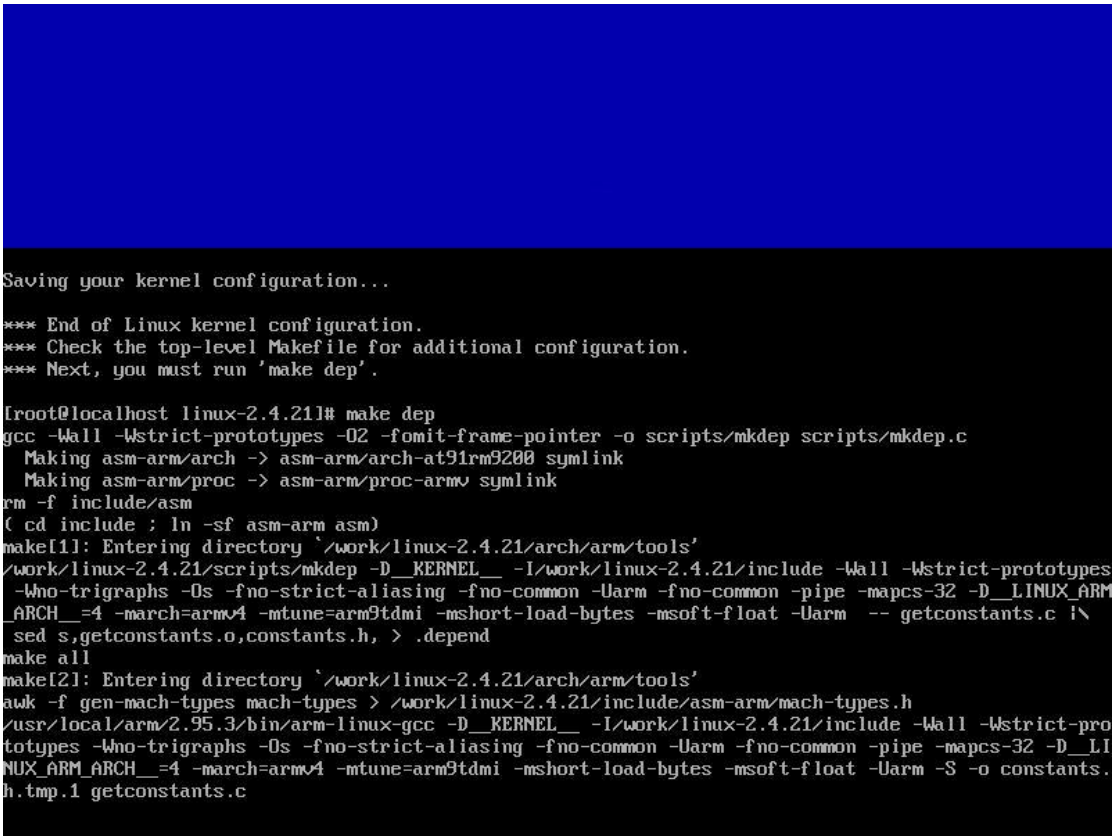


将 arch/arm/defconfig 改为 kernel_9200.cfg，再按 Ok 退出
回到主菜单后再按右键移动到 Exit 退出配置，出现保存配置的确认框

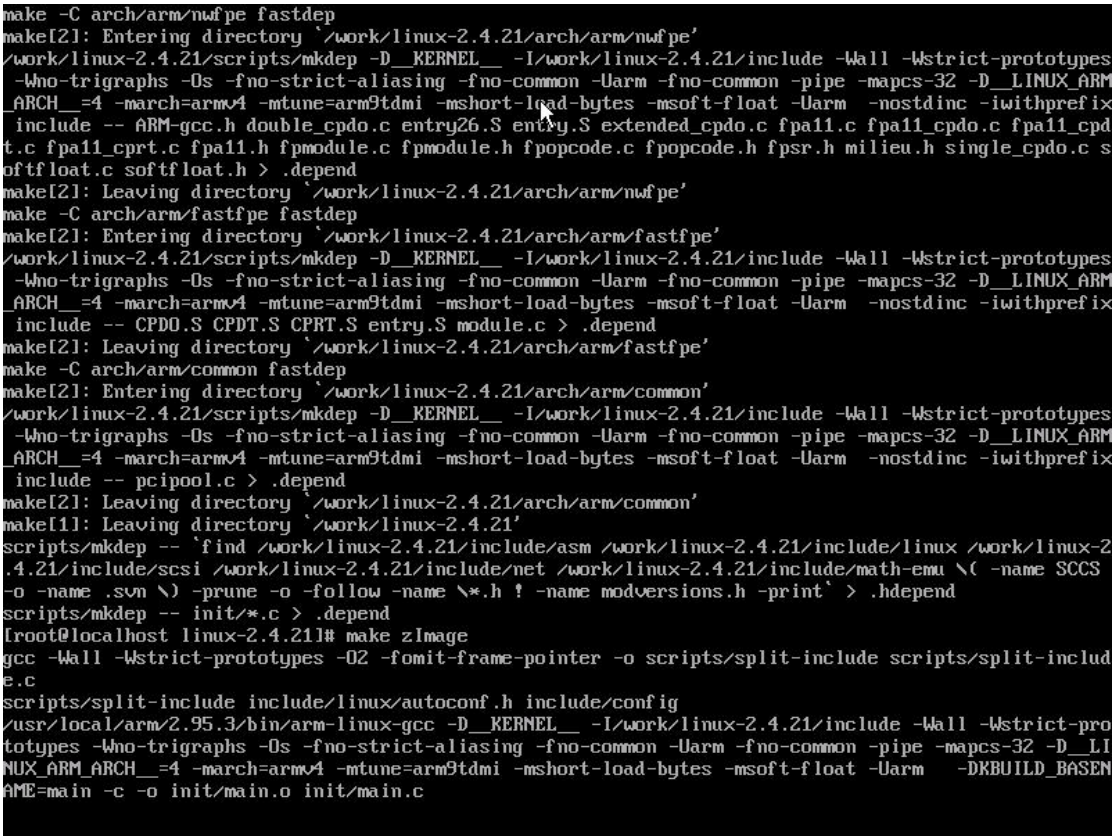


选到 Yes 后回车就保存配置了。

接下来执行 make dep，编译文件依赖关系



make dep 结束后执行 make zImage



编译结束后在 arch/arm/boot 目录下得到压缩内核文件 zImage，

```
h/arm/kernel/init_task.o init/main.o init/version.o init/do_mounts.o \
        --start-group \
        arch/arm/kernel/kernel.o arch/arm/mm/mm.o arch/arm/mach-at91rm9200/at91rm9200.o kernel/kerne
l.o mm/mm.o fs/fs.o ipc/ipc.o arch/arm/common/nopci.o \
         drivers/i2c/i2c.o drivers/serial/serial.o drivers/char/char.o drivers/block/block.o drivers
/misc/misc.o drivers/net/net.o drivers/ide/idedriver.o drivers/scsi/scsidrv.o drivers/cdrom/driver.o
 drivers/mtd/mtdlink.o drivers/usb/usbdrv.o drivers/media/media.o drivers/input/inputdrv.o drivers/a
t91/at91drv.o \
        net/network.o \
        arch/arm/nwfpe/math-emu.o arch/arm/lib/lib.a /work/linux-2.4.21/lib/lib.a \
        --end-group \
        -o vmlinux
/usr/local/arm/2.95.3/bin/arm-linux-nm vmlinux | grep -v '\(compiled\)\|\(\.o$\)\|\( [aUw] \)\|\(\.\
.ng$\)\|\(LASH[RL]DI\)' | sort > System.map
make[1]: Entering directory `/work/linux-2.4.21/arch/arm/boot'
make[2]: Entering directory `/work/linux-2.4.21/arch/arm/boot/compressed'
/usr/local/arm/2.95.3/bin/arm-linux-gcc -D__ASSEMBLY__ -D__KERNEL__ -I/work/linux-2.4.21/include -ma
pcs-32 -D__LINUX_ARM_ARCH__=4 -march=armv4 -msoft-float -traditional -c head.S
/usr/local/arm/2.95.3/bin/arm-linux-gcc -D__KERNEL__ -I/work/linux-2.4.21/include -O2 -DSTDC_HEADERS
 -mapcs-32 -D__LINUX_ARM_ARCH__=4 -march=armv4 -mtune=arm9tdmi -mshort-load-bytes -msoft-float -Uarm
 -fpic -Uarm -D__KERNEL__ -I/work/linux-2.4.21/include  -c -o misc.o misc.c
/usr/local/arm/2.95.3/bin/arm-linux-gcc -D__ASSEMBLY__ -D__KERNEL__ -I/work/linux-2.4.21/include -ma
pcs-32 -D__LINUX_ARM_ARCH__=4 -march=armv4 -msoft-float   -c -o head-at91rm9200.o head-at91rm9200.S
/usr/local/arm/2.95.3/bin/arm-linux-objcopy -O binary -R .note -R .comment -S /work/linux-2.4.21/vml
inux piggy
gzip -9 < piggy > piggy.gz
/usr/local/arm/2.95.3/bin/arm-linux-ld -r -o piggy.o -b binary piggy.gz
rm -f piggy piggy.gz
/usr/local/arm/2.95.3/bin/arm-linux-ld -p -X -T vmlinux.lds head.o misc.o head-at91rm9200.o piggy.o
/usr/local/arm/2.95.3/lib/gcc-lib/arm-linux/2.95.3/libgcc.a -o vmlinux
make[2]: Leaving directory `/work/linux-2.4.21/arch/arm/boot/compressed'
/usr/local/arm/2.95.3/bin/arm-linux-objcopy -O binary -R .note -R .comment -S compressed/vmlinux zIm
age
make[1]: Leaving directory `/work/linux-2.4.21/arch/arm/boot'
[root@localhost linux-2.4.21]# ls arch/arm/boot/zImage -l
-rwxr-xr-x    1 root     root       828536 Jun 29 14:23 arch/arm/boot/zImage
[root@localhost linux-2.4.21]#
```

可以在 BIOSBOX 里将此文件下载后烧入到 NANDFLASH 里去或直接运行，注意下载后要直接运行时指定地址为 20400000，如 comrun 20400000、rxrun 20400000、netrun 20400000 等。

6.2 制作 cramfs 根文件系统

上面的过程主要用于调试，掉电之后不能保存，要将程序固化，需要将该程序和模块添加到根文件系统中。这里我们将以 YL9200 光盘附带的 myroot.cramfs（在“目标代码”文件夹下）根文件系统的添加为例。

（1） 将 myroot.cramfs 拷贝到任意目录下

（2） 在该目录下建立两个文件：

mkdir chang

mkdir guo

（3） 将 myroot.cramfs 挂接到 chang 目录

mount chang myroot.cramfs -o loop

（4） 将 chang 目录下的内容压缩

cd chang

tar -cvf /chang 的上一级目录的绝对路径/1.tar ./

这样将在 chang 的上一级目录产生一个 1.tar 的包

（5） 将包解压到 guo 目录下。

umount chang ；卸载挂接

cd .. ；进入上一级目录

mv 1.tar guo ；

cd guo ；

tar –xvf 1.tar ；将打包的根文件系统的里的内容解压

rm 1.tar

（6）经过上面的步骤就可以将自己的驱动和应用程序添加到 cramfs 根文件系统中了

现在将开始制作 cramfs 根文件系统

先将 mkcramfs 文件拷贝到 guo 所在的目录

在这个目录下运行命令：

mkcramfs guo myroot1.cramfs

运行成功后，会在该目录下生成 myroot1.cramfs 根文件系统

（7）根文件系统制作成功后，就可以将 myroot1.cramfs 烧写到相应的地方，关于根文件系统的烧写，在“YL9200 使用手册”中有如何烧写根文件系统的说明。

（8）上面的步骤教你如何将自己的驱动和引用程序添加到根文件系统中。

6.3 编译一个 HELLO 的 DEMO 程序

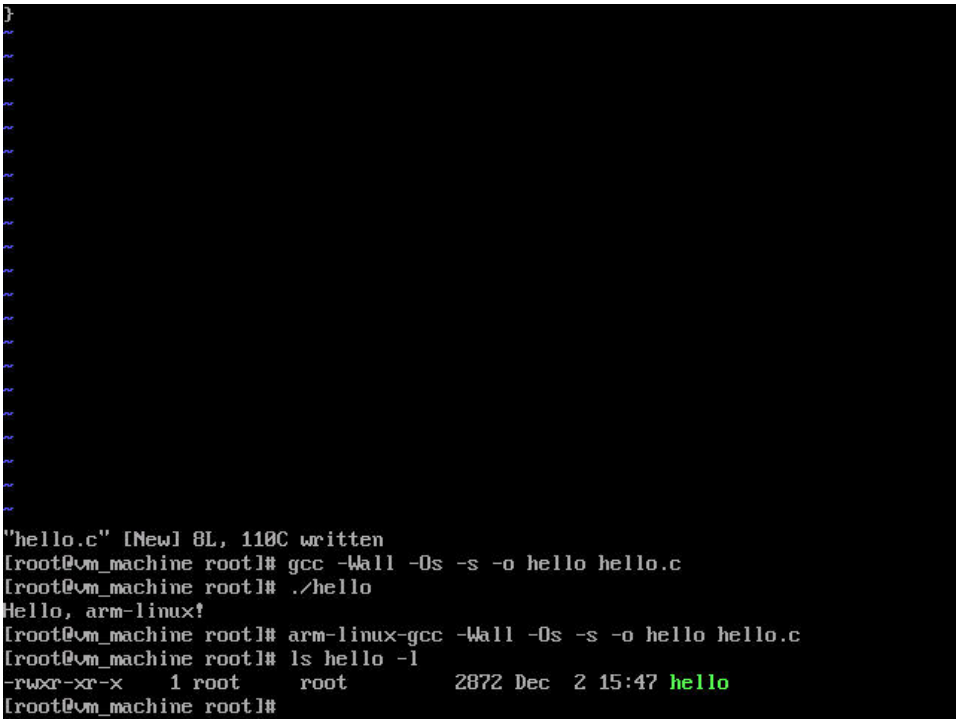
先在 Linux 下用文本编辑器编辑一个简单的 C 程序。

```
#include <stdio.h>
int main()
{
        printf("hello \n");
        return 0;
}
```

保存退出后，先用 GCC 编译并运行一下，之后再用 arm-linux-gcc 编译此程序，最后生成的可执行程序可以下载到开发板上运行。

编译 hello 的命令：

```
arm-linux-gcc -Wall -Os -s -o hello hello.c
```

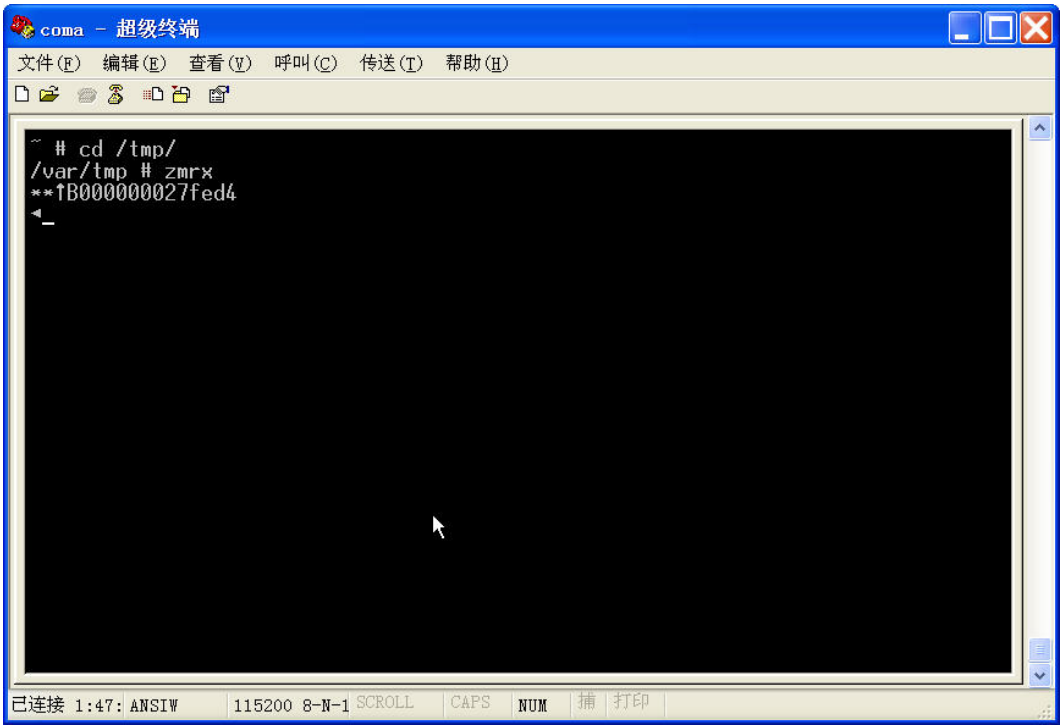


生成的 hello 程序，可以通过 FTP 传到开发板的 Linux 下的/tmp 目录下。
但对于这个小程序，下面一章节将介绍利用串口来进行传输。

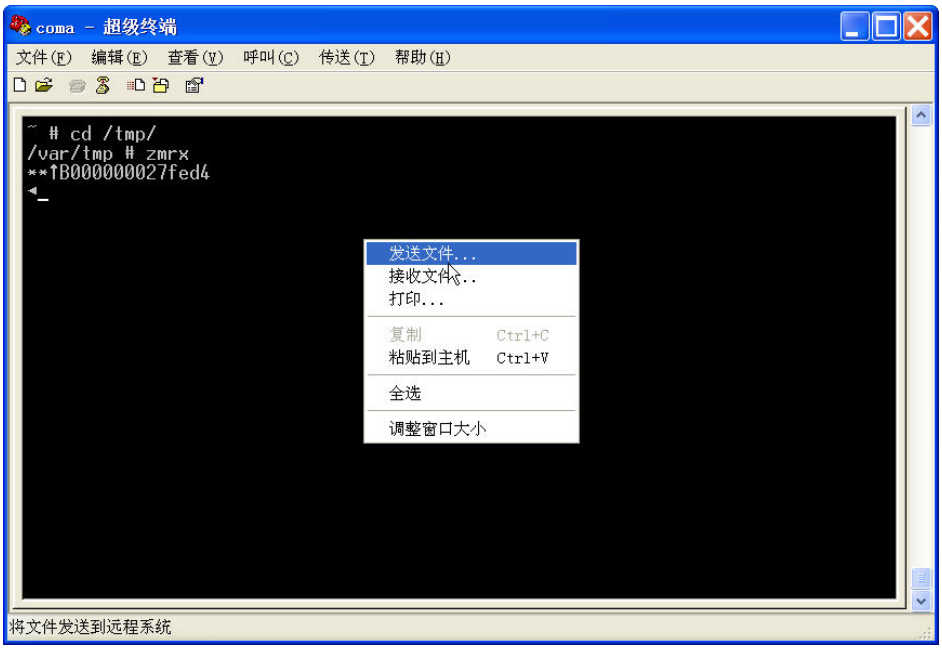
6.4 利用串口的 ZMODEM 协议进行文件下载

如果用户在开发板的 LINUX 系统与 PC 之间传送一个小文件，可以使用控制台所在的串口与超级终端利用 zmodem 协议来通讯。

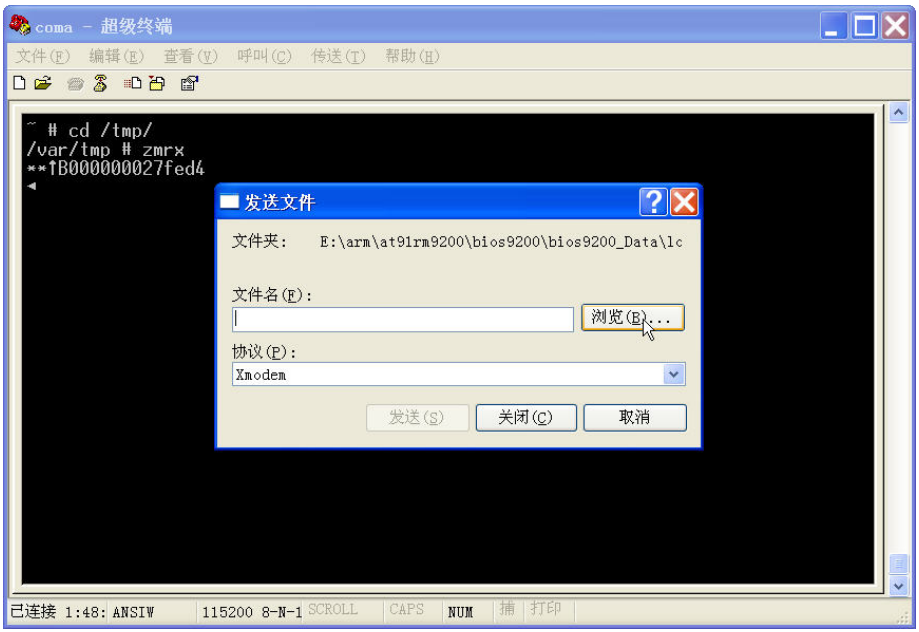
进到可写的目录/tmp，执行 zmrx 命令



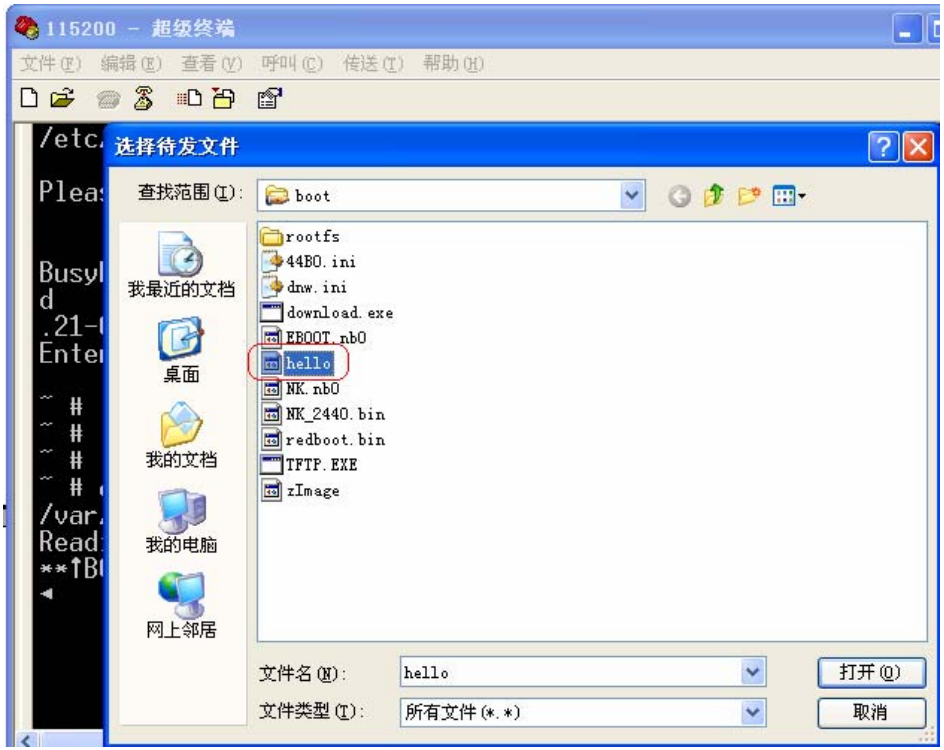
在超级终端里点击鼠标右键



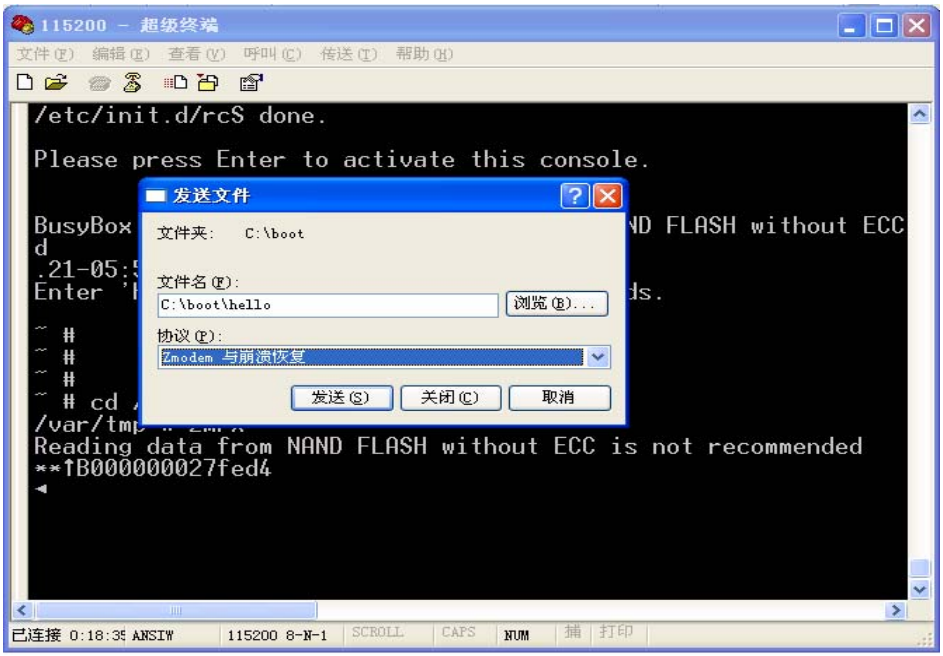
选择发送文件



再选择浏览找到要发送的文件 hello



回到上一个对话框，选择以 Zmodem 与崩溃恢复方式发送，设置好后按发送


115200 - 超级终端
文件(F) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
/etc/init.d/rcS done.
Please press Enter to activate this console.
发送文件
文件夹: C:\boot
文件名(F):
C:\boot\hello
浏览(B)...
协议(P):
Zmodem 与崩溃恢复
发送(S)
关闭(C)
取消
Reading data from NAND FLASH without ECC is not recommended
已连接 0:18:35 ANSIW
115200 8-N-1
SCROLL
CAPS
NUM
捕
打印

下图为发送界面

发送完毕后，它会自动结束。返回到 shell 状态。

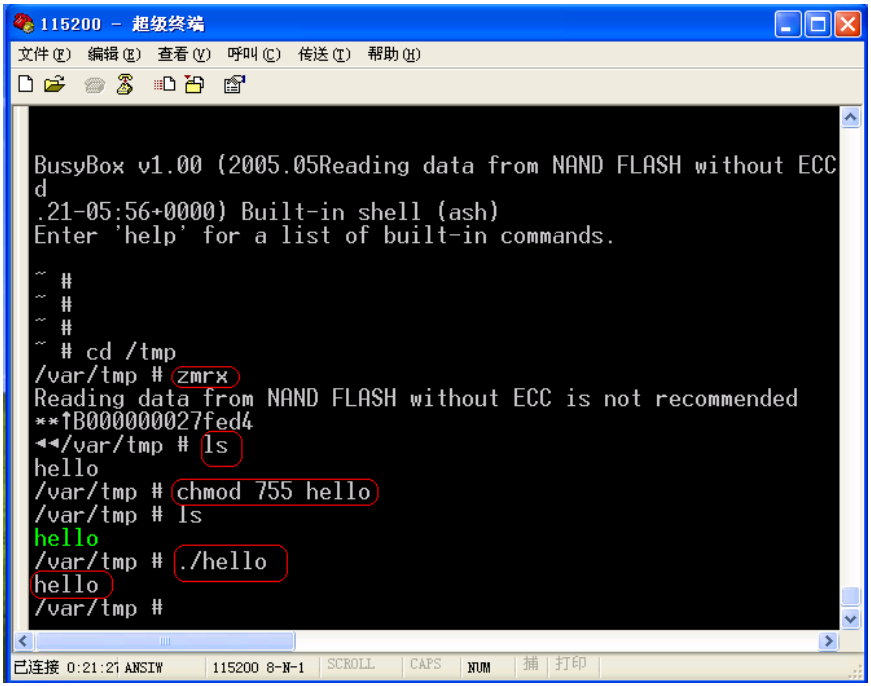
Hello 下载完成后，现在可以执行它了。

执行命令如下：

chmod 755 hello

./hello

这时可以看到 hello 运行的情况。



注意，接收文件的目录必须是可写的，并且此目录下不能存在与要接收的文件名称相同的文件，有的话先删除或改名。另外 BUSYBOX 自带的 rx 命令也是一个接收程序，但是使用 xmodem 协议。也可以使用 ftp 的方法进行传送文件,或用 U 盘。