FFT-RM9200 说明书

西安傅立叶电子科技 ARM 技术研发部

2004 年 9 月 23 日

目　录

第六章 嵌入式 Linux 在 FFT-RM9200 上的开发

- 应该对 Linux 的开发有个大致的了解
- 嵌入式 Linux 在 FFT-RM9200 上的移植
- 到此时，你所想和需要的是什么？

第七章 嵌入式 Linux 的内核开发

- 傅立叶给你提供的内核开发资源
- 开发 Linux 内核的步骤
- 如何配置和裁剪 Linux 的内核
- 如何添加新的内核配置和驱动
- 嵌入式 Linux 内核的编译
- 对于嵌入式 Linux 的内核开发，你还需要什么？

第八章 嵌入式 Linux 的文件系统开发

- 傅立叶给你提供的文件系统开发资源
- 开发 Linux 文件系统的一般开发步骤
- 如何开发 FFT-RM9200 的文件系统
- 图形界面和上网
- U 盘的访问
- 对于嵌入式 Linux 的文件系统开发，你还需要什么？

第九章 如何开发和运行你的顶层程序

- “Hello World”程序的编译与调试
- 如何使得你的应用程序在目标板上运行
- 对于顶层程序的开发，你还需要什么？

第十章 图形界面的开发

- RM9200 的图形界面介绍
- 在 Microwin 下开发图形界面的顶层程序
- 在 MiniGUI 下开发图形界面的顶层程序
- 你还需要什么？

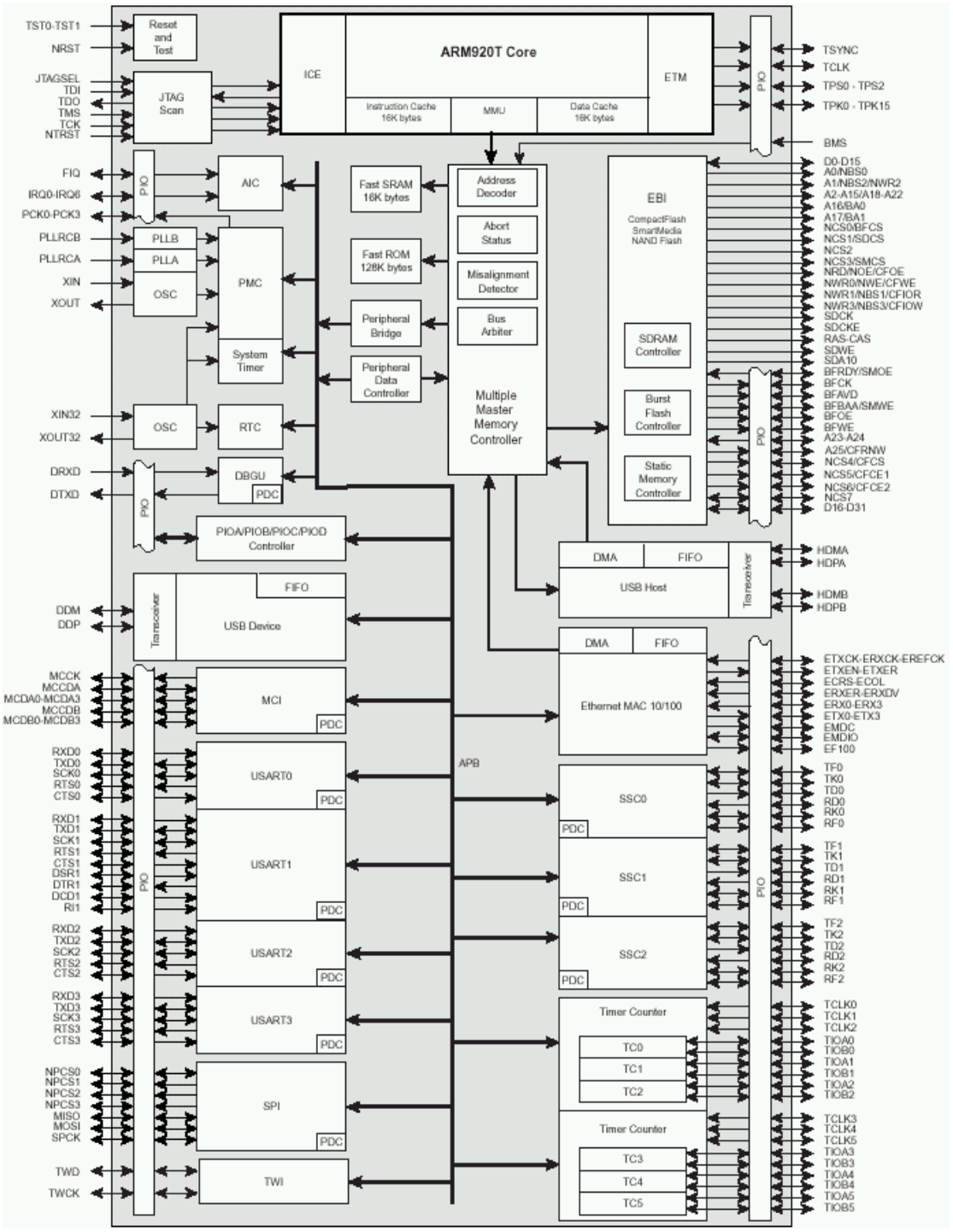
附录：

- FAQ
- JTAG 接口电路
- FFT 提供的 Linux 下的例程

第一章　　了解 FFT--RM9200

1．AT91RM9200 处理器介绍

- 内部集成了一个ARM920T—ARM Thumb处理器
– 在180 MHz时运行速度高达200 MIPS
– 内带16KB的数据Cache，16KB指令Cache，写缓冲区
– 全功能的MMU
– 片内带有Debug通信通道的Emulator
– 嵌入式 Trace Macrocell (仅对于256-ball BGA 封装模式)
- 内部Memories
– 16KB的SRAM 和128KB的ROM
- 外部总线EBI接口
– 支持SDRAM, Static Memory, Burst Flash, Glueless Connection to CompactFlash®, SmartMedia™ and NAND Flash
- 系统设备的增强功能:
– 增强型的时钟产生器和电源管理控制器
– 带有两个PLL的在片振荡器
– 慢速的时钟操作模式和软件电源优化能力
– 四个可编程的外部时钟信号
– 包括周期性中断、看门狗和第二计数器的系统定时器
– 带有报警中断的实时时钟
– 调试单元，两线UART 和支持Debug调试通道
– 带有8个优先级、可单个屏蔽中断源、Spurious中断保护的先进中断控制器
– 7个外部中断源和一个快速中断源
– 四个32位的PIO控制器可以达到122个可编程I/O引脚（每个都有输入控制、可中断及开路的输出能力）
– 20通道的外部数据控制器 (DMA)
- 10M/100M网卡
– Media Independent Interface (MII) or Reduced Media Independent Interface (RMII)
– 集成28字节的FIFOs和直接用于收发 DMA 通道
- USB 2.0 主口2个(12 M-bits/秒
– 两个在片的收发器 (208脚的PQFP封装只有一个)
– 集成的FIFOs和DMA通道
- USB 2.0 从口1个
– 在片收发器, 2-KB可配置集成的FIFOs
- Multimedia Card Interface (MCI)
– Automatic Protocol Control and Fast Automatic Data Transfers
– MMC and SD Memory Card-compliant, Supports Up to Two SD Memory Cards
- 3个异步的串行控制器(SSC)
– 对于每个收发器的独立的时钟和帧信号
– I2S Analog Interface Support, Time Division Multiplex Support



AT91RM9200**的内部结构图**

– High-speed Continuous Data Stream Capabilities with 32-bit Data Transfer

• Four Universal Synchronous/Asynchronous Receiver/Transmitters (USART)

– Support for ISO7816 T0/T1 Smart Card

– Hardware and Software Handshaking

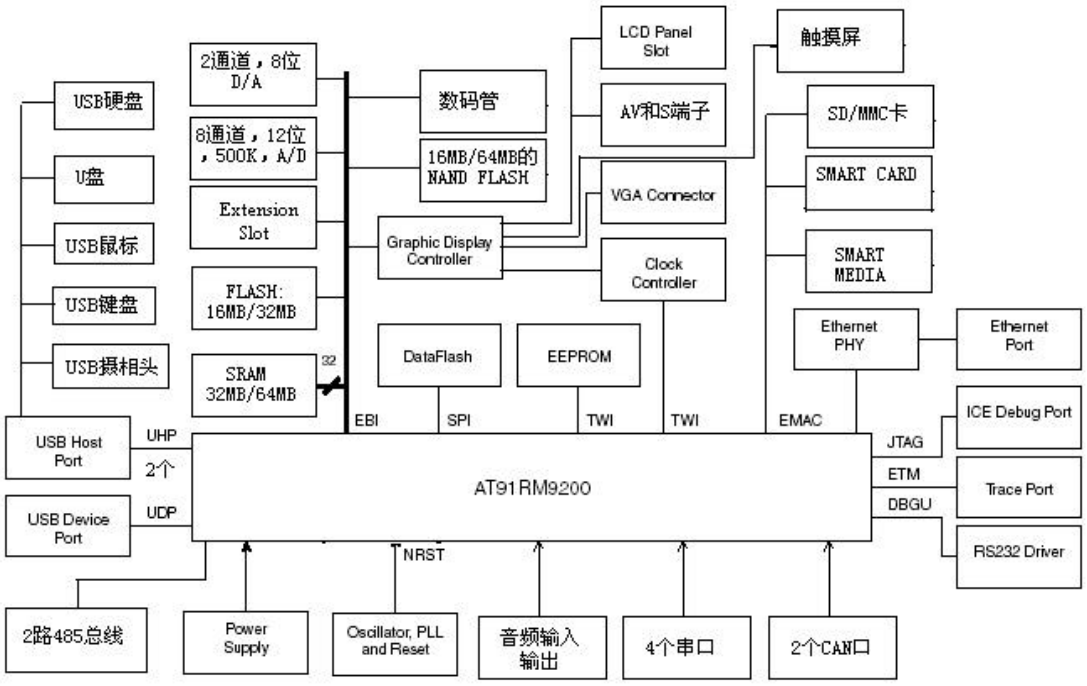
– RS485 **支持**, IrDA **速度可达**115 Kbps

– Full Modem Control Lines on USART1

• **SPI接口**
– 8/16**位可编程数据长度**, 4 **个外设芯片选择**
• **2个 3通道的定时/计数器(TC)**
– 3**个外部时钟输入**，**每个**I/O**通道有两个可以复用的**I/O**口**
– **栓**PWM**生成**，**捕获/波形模式**，**向上和向下计数兼容**
• **Two-wire Interface (TWI)**
– **主模式支持**，**所有**2**线的**Atmel EEPROMs **支持**
• **IEEE 1149.1 JTAG Boundary Scan on All Digital Pins**
• **Power Supplies**
– 1.65V **到** 1.95V for VDDCORE, VDDOSC and VDDPLL
– 1.65V **到** 3.6V for VDDIOP (**外设** I/Os) and for VDDIOM (**内存** I/Os)
• **Available in a 208-lead PQFP or 256-ball BGA Package**

2． FFT-RM9200 的资源介绍

2．1 FFT-RM9200 的结构图形如下



2．2 主要功能

整体概述

- **基于** AT91RM9200 **的** FFT-RM9200 **评估板**
- **处理器运行频率** 180 MHz，**速度为** 200MIPS
- **总线运行频率** 60 MHz
- **标准的** ATX **电源，类似一台小工控机**
- **在板的** SDRAM, DataFlash® and EEPROM、NAND FLASH **内存**

- USB (6 个主口和 1 个从口)
- 快速网络，串口接口
- 音频接口
- CAN2.0 总线接口：2 路
- RS485 总线接口，2 路
- SD/MMC 卡：2 个
- SMART MEDIA CARD：1 个
- CF 卡：1 个
- 显卡扩展：可同时接 VGA、LCD、AV 端子、S 端子
- JTAG/ICE, DBGU 调试接口
- 扩展插槽

AT91RM9200处理器

_ 集成ARM920T™ ARM® Thumb® 处理器
– 在180MHz时，速度高倒200 MIPS
– 16-KB数据Cache, 16-KB指令Cache, 写缓冲区
– 虚拟内存管理单元MMU
– 带有Debug调试的在片 Emulator
– Mid-level Implementation Embedded Trace Macrocell
– 16KB的内部SRAM 和128K B的内部ROM
_ 外部总线接口 (EBI)
– 支持 SDRAM, SRAM, Burst Flash, 和CompactFlash®, SmartMedia™ and NAND Flash 的无缝连接

系统设备：

– 增强型的时钟产生器和电源管理单元
– 带有两个PLL的两个在片振荡器
– 慢速的时钟操作模式和软件电源优化能力
– 4个可编程的外部时钟信号
– 包括周期性中断、看门狗和第二计数器的系统定时器
– 带有报警中断的实时时钟
– Debug Unit, Two-wire UART and Support for Debug Communication Channel
– 带有8个优先级、可单个屏蔽中断源、Spurious中断保护的先进中断控制器
– 7个外部中断源和一个快速中断源
– 四个32位的PIO控制器可以达到122个可编程I/O引脚（每个都有输入控制、可中断及开路的输出能力）
– 20通道的外部数据控制器 (DMA)
_ 10/100M的以太网接口
_ 6个全速的USB 2.0 主接口和一个从口
_ Four Universal Synchronous/Asynchronous Receiver/Transmitters (USART)
_ Two3通道16位的定时/计数器(TC)
_ 两线接口(TWI)
_ IEEE 1149.1 JTAG 标准扫描接口

存储器

_8MB/16MB/32MB的Nor Flash内存

_ 32MB/64MB的SDRAM

_带有两线接口的512KB EEPROM

_ 最大为8M字节的串行DataFlash接口

_ 16MB/64MB的NAND FLASH接口

_ SD/MMC存储卡接口

_ CF存储卡接口

_ SMART MEDIA CARD存储卡接口

时钟电路

_ 32.768 kHz 标准时钟为AT91RM9200

_ 18.432 MHz 针对AT91RM9200的标准晶振

_ 50 MHz 针对显示控制器的标准晶振

_ 16 MHz CAN控制器的标准晶振

复位电路

_ Reset 控制器

电源供应电路

_标准的ATX电源

_5V单电源供电可选

远程通信

_单片收发的快速Ethernet物理层

_ 4个RS-232 DB9 接口

_ 1个RS-232 DB9 连接器的Debug端口

_ 2路RS485总线端口

_1路红外端口

_ 2路独立的CAN2.0总线接口

_ 主和从USB socket包

用户接口

_ 图形显示控制器

_ TFT/SNT LCD接口

_ 标准VGA显示器接口

_ 标准的S端子接口

_ 标准的AV端子接口

_ 8路，12位，500K的A/D

_ 2路，8位的D/A

_ 2个数码管

_ IRQ和FIQ中断接口

扩展槽

_ 两个标准的96针欧氏插槽

_外部可以扩充：Atmel原版的四种卡、可以更换显示芯片接口、DAA接口、指纹识别接口

_总线接口、片选接口、众多的PIO接口全部扩充

Debug 接口

_ 20脚JTAG接口器件

_ 串口调试单元

2．3 FFT-RM9200 的基本跳线和按键设置

下面是 FFT-RM9200 的基本跳线设置，关于详细的跳线设置，请根据你所用到的功能，参考原理图

表1 FFT-RM9200底版的重要跳线设置

插座	缺省设置	特征描述
SW1		电源开关
BP1		复位开关
JP1	接3V3	接3V3，由ATX电源的3.3V直接供电 接CPU 3V3，由5V转换的3.3V供电
JP2	Closed	1.8V供电
JP3	Closed	5V转换的3.3V控制跳线
JP4	Closed	当此跳线Closed时，系统从并行FLASH中启动
JP7	Closed	当Closed时，系统不从EEPROM启动
J9	接JTAG	接JTAG， Jtag仿真机使用，包括FFT-ICE及兼容设备接ICE， Trace等仿真机使用

表2 FFT-RM9200 CPU板的跳线设置

插座	缺省设置	特征描述
JP1	Closed	1.8V电压的跳线设置，CLOSE表示供电
JP2	Closed	16MB的FLASH，详细参见原理图

表3 FFT-RM9200 13806掀卡板的跳线设置

插座	缺省设置	特征描述
JP1	Closed	1.8V电压的跳线设置，CLOSE表示供电
JP2	Closed	16MB的FLASH，详细参见原理图

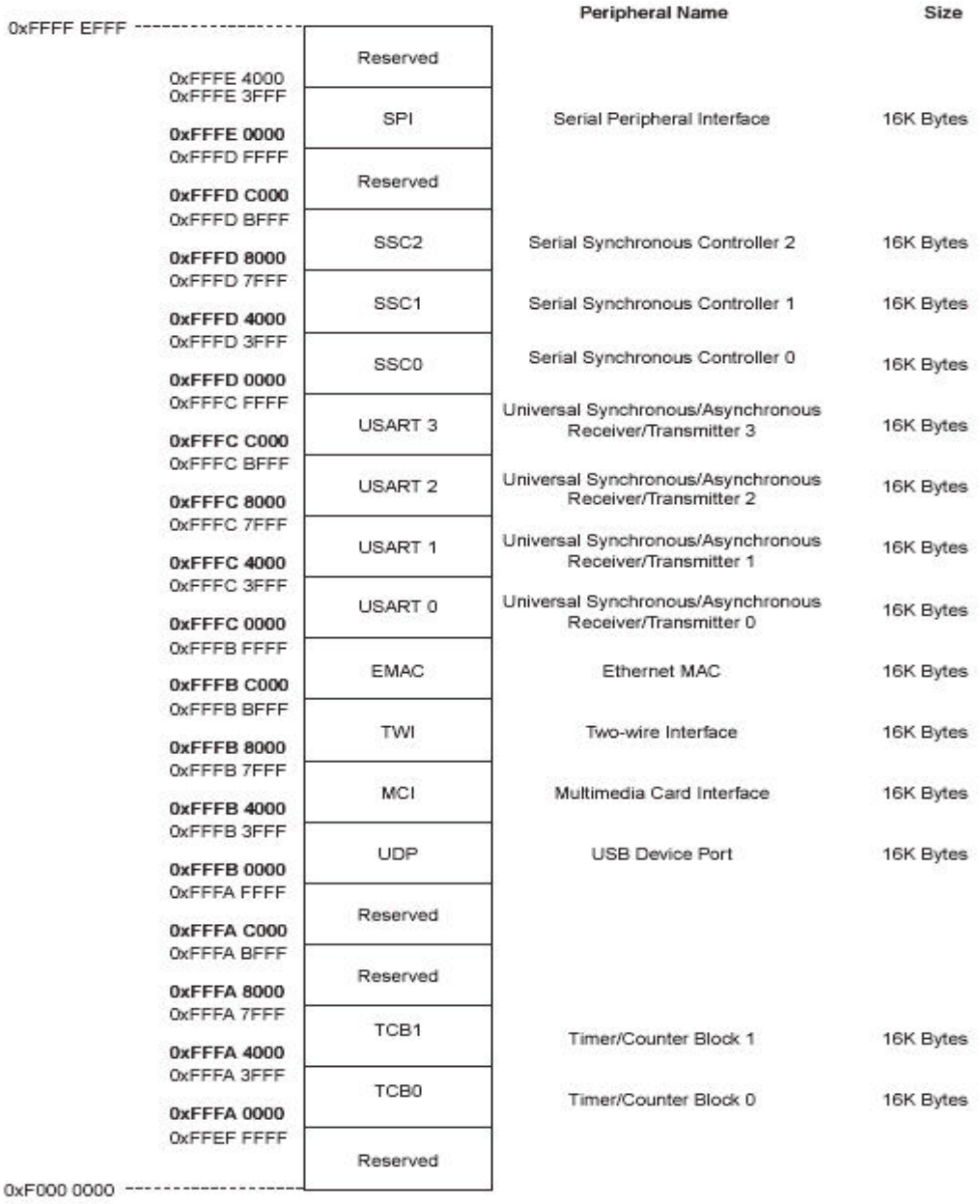
2．4 FFT-RM9200 的主要 memory 资源分配

资源名字	接口方式	内存分布	备注
SDRAM	NCS1	20000000—21FFFFFF	32MB
并行 FLASH	NCS0	10000000—10FFFFFF	16MB
		10000000---10005FFF	Fft-boot.bin
		10010000---1001FFFF	Fft-Uboot.gz
		10020000---1003FFFF	环境变量
EEPROM	I^2C	512K 字节	AT24C512（512KB）
串口 FLASH	SPI		可选配置
NAND FLASH			可选配置

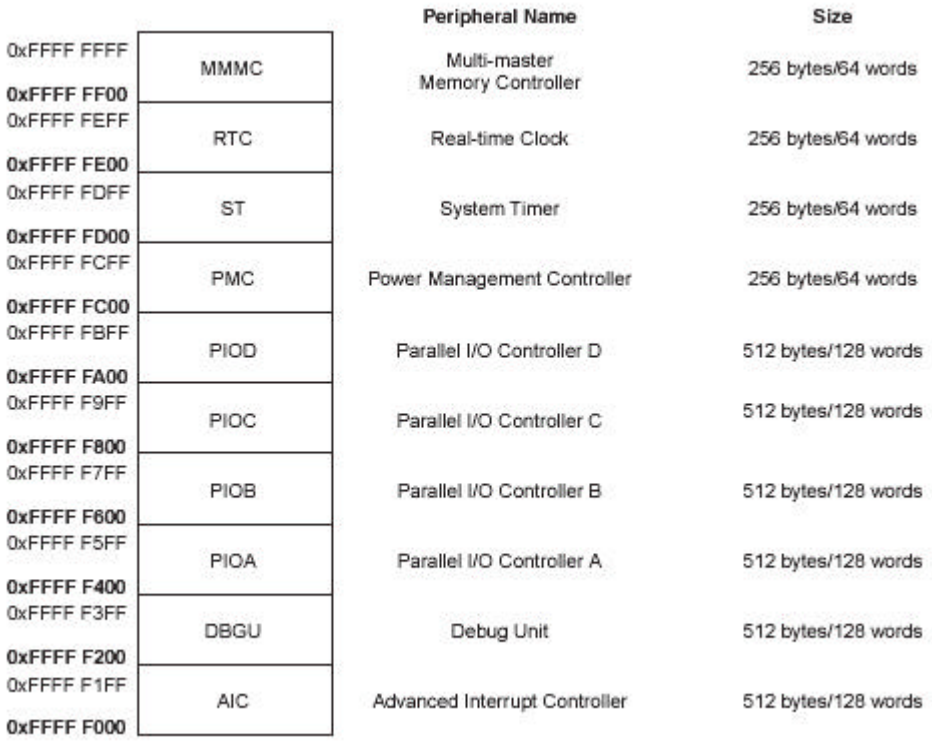
2．5 内部资源

Internal SRAM ：16KB，REMAP 之前 0x200000，REMAP 之后 0x0
Internal ROM ：128KB，地址为 0x10 0000；当从 bootROM 启动时，REMAP 之前为 0x0
USB Host Port： 地址为 0x30 0000.

2．6 用户设备影射



2．7 系统设备映射



		Peripheral Name	Size
0xFFFF FFFF **0xFFFF FF00**	MMMC	Multi-master Memory Controller	256 bytes/64 words
0xFFFF FEFF **0xFFFF FE00**	RTC	Real-time Clock	256 bytes/64 words
0xFFFF FDFF **0xFFFF FD00**	ST	System Timer	256 bytes/64 words
0xFFFF FCFF **0xFFFF FC00**	PMC	Power Management Controller	256 bytes/64 words
0xFFFF FBFF **0xFFFF FA00**	PIOD	Parallel I/O Controller D	512 bytes/128 words
0xFFFF F9FF **0xFFFF F800**	PIOC	Parallel I/O Controller C	512 bytes/128 words
0xFFFF F7FF **0xFFFF F600**	PIOB	Parallel I/O Controller B	512 bytes/128 words
0xFFFF F5FF **0xFFFF F400**	PIOA	Parallel I/O Controller A	512 bytes/128 words
0xFFFF F3FF **0xFFFF F200**	DBGU	Debug Unit	512 bytes/128 words
0xFFFF F1FF **0xFFFF F000**	AIC	Advanced Interrupt Controller	512 bytes/128 words

3．傅立叶公司为你提供什么开发资源？

作为开发商，傅立叶电子科技（FFT）为你提供从硬件、测试、操作系统以及用户程序的编写方法和移植在内的一体化解决方案。

3．1 硬件方面

FFT-RM9200 **评估板**

ATX **标准电源**

串行调试接口线

网线 1 **根（交叉网线）**

3．2 硬件可选配置

CAN **接口板**

S1D13806 **的接口板**

3．3 软件配置

Linux9.0 **的软件**

FFT-RM9200 **的** Linux **下的** Bootloader**、内核与文件系统影象**

BootLoader**、内核和文件系统的原代码**

面向 microwin **和** minigui **图形界面的文件系统**

经过商业包装后的针对 FFT-RM9200 **的** Linux **交叉编译开发环境，你可以轻松使用**

3．4 FFT-RM9200 正在开发的软件系统

WinCE **在** FFT-RM9200 **移植**

VxWorks 在 FFT-RM9200 中的移植

3．5 FFT-RM9200 开发资料

FFT-RM9200 的产品说明书

FFT-RM9200 的产品原理图

FFT-RM9200 的板卡布局图

FFT-RM9200 上的主要芯片设计资料

有关 ARM9 设计中的一些典型外围器件和设计

ARM9 开发的常见问题解答

3．6 针对 FFT-RM9200 的有关培训

2004 年 4 月开始，开始举办 ARM 方面的培训班，你可以随时了解傅立叶在各地办事处的培训班情况，让你能够轻松使用 ARM，了解和实验整个操作系统在 ARM 中的移植、如何进行 ARM 产品的系统设计，对于特殊用户的培训要求，请尽快和 FFT 的各地办事处联系

4．针对更多的需求，你还需要什么？

对于一个 ARM 系统，你的开发可能有两种选择，有操作系统和没有操作系统，对于你不用操作系统的开发，傅立叶可以为你提供 FFT-ICE+ADS1.2 的开发系统，你可以象单片机一样轻松进行 ARM 的开发；对于采用操作系统，目前傅立叶为你提供了 Linux 系统下的一体化解决方案，你可能并不需要非常熟悉 Linux，你只需要按照说明书上提供的方法进行开发，便可以完成你的项目。同时傅立叶也正在进行其他操作系统的移植，同时为你提供足够的支持。

可能对于 FFT-RM9200 板卡的开发，傅立叶都为你想到了，但是可能你还心存疑惑，对于这么多资源的板卡，你可能希望以最快的速度让它运转，当然也完成你的产品开发，你需要的可能还有如下方面：

4．1 培训和支持：

你可能是 ARM 的入门者，你需要 ARM 和操作系统的知识，可能这些都是从书本上不能得到的，因为能够开发 ARM 的人员，都集中在各个大公司的研发部门，从学校、研究所或者你周围的人们，研究 ARM 的人员很少，你可能会为一个小问题，小 Bug 而陷入开发的困境。

解决方法：傅立叶为你提供足够的支持，从你购买我们的产品到你的设计的结束，傅立叶永远是你的朋友和技术的后盾。对于你需要的扩展部分，傅立叶和你商讨方案的设计和方法，当然，如果想进行长期产品合作，傅立叶也很乐意。

4．2 产品硬软件的升级

对于我们的产品，软硬件也在不停的升级之中，每一次升级，傅立叶将为你免费提供新的资源，对于产品中的心得体会，傅立叶愿与你分享。

4．3 你可能最需要的还是技术支持

这部分是傅立叶目前最为关注的部分，客户的成功才是傅立叶的成功，因此对于技术支持，可能是你应该最为放心的，购买到我们的产品，首先我们让你把产品先运转起来，然后对于细节的问题，你可以找当地的技术支持，也可以向我们打电话或者发 Email，24 小时之内必有回复。

4．4 你还感到迷惑吗？

这说明你还需要了解更多的东西，比如操作系统的基础知识，如果你连任务调度都没有听说过、或者你从来不了解硬件系统，你可能首先阅读类似方面的书籍，你不需要了解很多，有个概念，或者看一看傅立叶在 2003 年 10 月份举行研讨会的会刊（可以向傅立叶申请），我想你就可以进行开发了。

4．5 你最应该仔细阅读的东西

对于几乎所有的工程师而言，阅读芯片的文档资料是非常重要的内容，对于大多数工程师，中文资料只是入门，大致了解之后，几乎所有人都把精力放在原版英文资料上，**鉴于水平有限，以免误导用户**，在阅读完这本说明书后，建议用户应该仔细阅读下面几个资料。

入门资料：

- 电路原理图
- 原理图上的器件资料
- AT91RM9200 的芯片资料
- Linux 相关软件开发的基本内容

高级用户

- 软件原代码阅读
- ARM920T 的 ARM 内核资料
- Microwin 和 minigui 的图形界面资料

对于仍然有疑惑的用户，请关注傅立叶的 ARM 技术培训班！

4．5 最后的建议

嵌入式开发是一个涉及到软件、硬件以及操作系统、驱动程序等于一体的开发，综合性非常强，首先，你应该了解方方面面的内容，打好基础，认真对待，这样产品开发才有可能成功；同时，又是一个不断迅速发展的领域，新的内容层出不穷，希望有志于此的工程师们获得成功。

祝你的产品早日成功上市！！！

傅立叶电子科技 ARM 技术研发部　tech@fftchina.com

2004-10-8

第 2 章　开发 FFT-RM9200 概述

***内容*：**

通过上一章，你可能已经了解了 FFT-RM9200 的概况，整体介绍如何开发 FFT-9200？通过这一章，你应该对 FFT-RM9200 的开发有个轮廓，确定你的开发方向。

当你拿到 FFT-RM9200 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，你大可不必要去看，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，熟悉 FFT-RM9200
- 对于不用操作系统的开发步骤
- 对于采用操作系统的开发步骤
- 如果是一个裸板，你应该怎么做？

1．搭建环境，为 FFT-RM9200 加电

下面几步是你拿到 FFT-RM9200 板子时，**从并行 FLASH 启动**，你能够看到的。。。。因为傅立叶已经在 Flash 中为你烧写好了 U-Boot 程序，对于你的空板子，请看本章第四节的内容。

STEP 1：

在熟悉 FFT-RM9200 的基础上，进行各种跳线设置（参照第一章的跳线设置表格），主要以下跳线：

（1）底版：

JP2　Closed；　　**JP3　Closed；**

JP1　3V3；　　**JP4　Closed；**

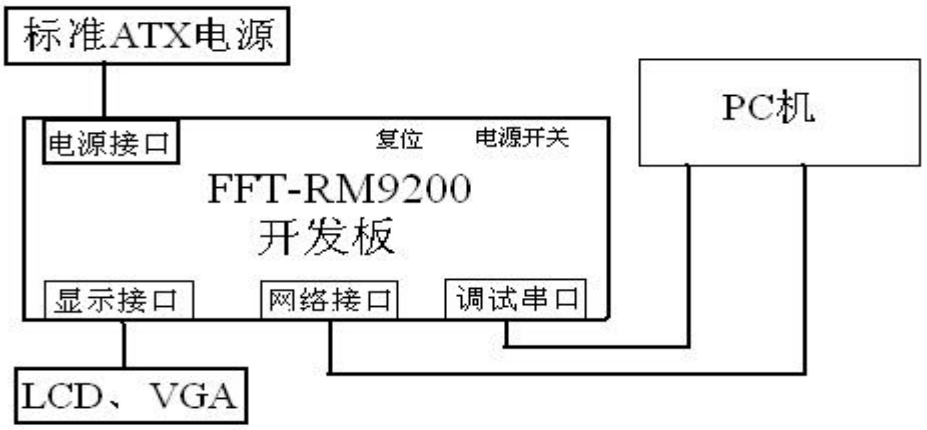
J9　JTAG　　**JP7　Closed**

（2）CPU 板

JP1，JP2：Closed

STEP 2：

按照下图，进行 FFT-RM9200 开发系统的连接。

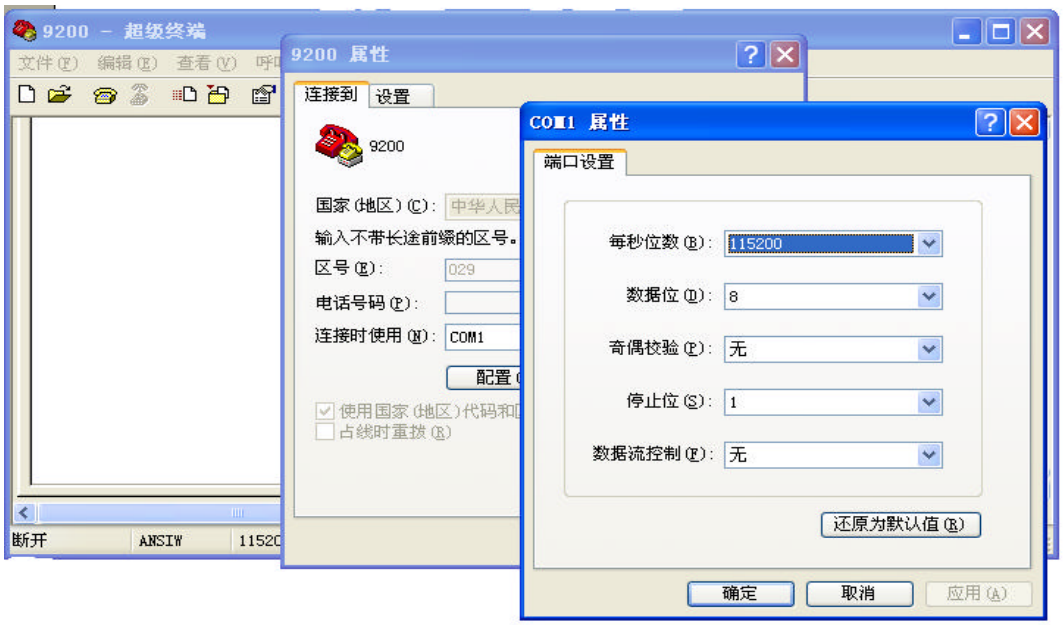


STEP 3：

进行 PC 宿主机的设置：

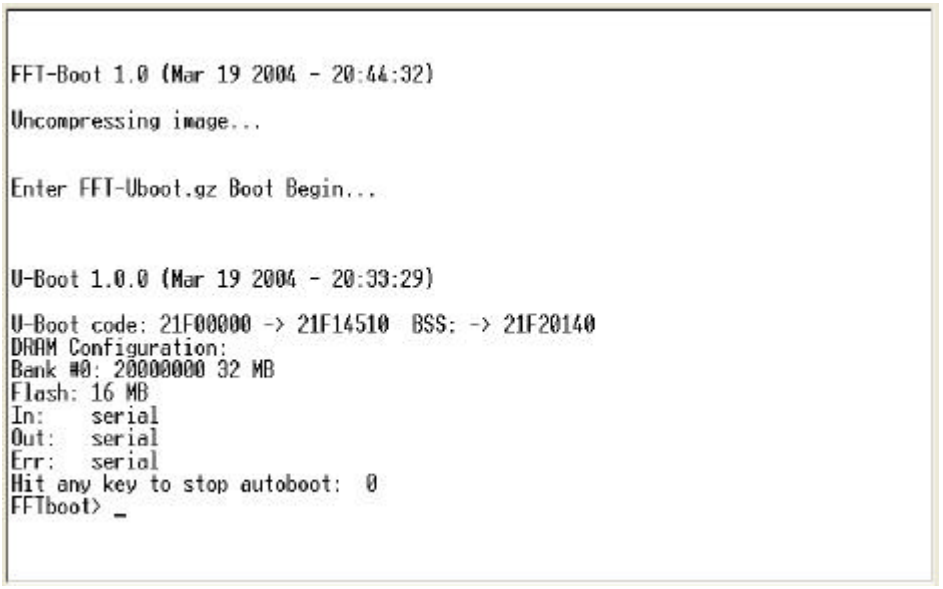
打开超级终端，设置串口为 115200，8，无，1，无，让超级终端处于接收状态；

连接 VGA 或 LCD、USB 键盘和 USB 鼠标



STEP 4：加电和复位

加电或者复位后，超级终端接收的信号如下：



按下任意健，超级终端会出现提示符 FFTBoot>，否则系统会自动从 FLASH 中运行。

2． 采用 ADS 和 FFT-ICE 开发 FFT-RM9200 的步骤

对于 ADS 和 FFT-ICE 的详细内容，本说明书不作详细介绍，在此，你应该在了解 FFT-RM9200 的基础上，知道傅立叶为你提供了什么？如何使用 ADS 和 FFT-ICE 的步骤。

STEP 1：

如果你采用傅立叶提供的 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 J9 跳线到右端，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 FFT-RM9200 的开发。

3． 采用嵌入式 Linux 开发 FFT-RM9200 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 FFT-RM9200 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 FFT-RM9200 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
- 文件系统的相关内容：比如 BusyBox、Samba 等等
- 检查其他的部分：图形环境、中文内核等等
- 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。

4． 如果你擦除了所有的东西，或者系统崩溃，你如何去做？

当你的系统崩溃时，或者不能启动、擦除了 FLASH 时，你将不能从 FLASH 中进行启动，也就是你看不到我们本章第一节的内容，这时，你需要从 AT91RM9200 的内部 BOOTROM 进行启动，**Open JP4（去掉 JP4 跳线）**，打开超级终端（115200、8、无、1、无），加电，在超级终端会出现“ CCCCCCCC。。。。”，你需要做下面几步。

STEP 1：

在超级终端下，找到 U-BOOT 目录下的 binary，首先在 Xmodem 下，发送 fft-loader.bin 文件，然后超级终端会出现下载 Uboot 的提示，然后继续出现“ CCCCCCCCC.…… ”

STEP 2：

在 Xmodem 下，发送 fft-Uboot-v2.0.bin 文件，发送完毕后显示>FFTBoot 的提示符

STEP 3：

擦除 FLASH

U-BOOT>protect off all ;

U-BOOT>erase all ;

STEP 4：

装入 fft-boot.bin

U-BOOT>Loadb 20000000 ;

在超级终端，用 Kermit 模式发送文件 fft-boot.bin

U-BOOT>cp.b 20000000 10000000 5fff

在超级终端，显示拷贝的情况

STEP 5：

装入 fft-Uboot.gz

U-BOOT>Loadb 20000000 ;

在超级终端，用 Kermit 模式发送文件 fft-Uboot-v2.0.gz

U-BOOT>cp.b 20000000 10010000 ffff

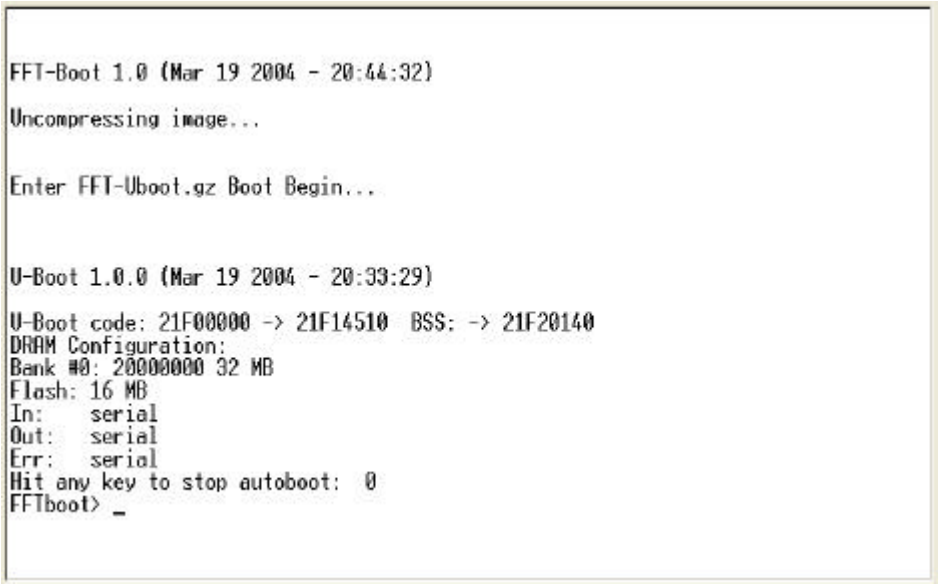
在超级终端，显示拷贝的情况

U-BOOT>protect on 10000000 1001ffff ;

FLASH 区域保护

STEP 6：

Closed JP4（合上 JP4 跳线），复位后，超级终端接收如下



你的 U-BOOT 完全烧写成功，你可以进行操作系统内核和文件系统的烧写和测试了。

第 3 章 FFT-RM9200 的硬件系统简介

主要内容：

简单介绍 FFT-RM9200 的一些的硬件模块，详细的内容，客户请参见原理图即可

1． 系统的硬件构成

（1）CPU 板

AT91RM9200

目前采用 BGA **封装，也可以采用** MQFP **封装（相对应底版上一组** USB **主口失效）**

SDRAM

2 **片** SDRAM**，兼容** ***32MB 和 64MB***

片选为 CS1**，**

缺省：使用 32MB 的 SDRAM

地址范围：0x20000000---0x21ffffff

FLASH

片选：CS0 **和** A24 **译码，得到高位和低位片选**

低位上接 2MB **和** 16MB **两种封装的** FLASH**，高位上接一片** 16MB **的** FLASH

缺省：采用一片 FLASH（16MB）：

地址范围：0x10000000---0x10ffffff

接口插座（2.0mm **间距）**

共三组，每组 72 **根线，把重要的信号全部引出。**

（2） 底版

- **插针**

  连接 9200 **的** CPU **板**

- **缓冲器（**16245**）**

  把 CPU **板的信号与底版的信号相隔离**

- **电源**

  保持原来的 ATX **电源不变，留下** 4*2 **的插针两组，留下接单个电源的接口**

- A/D、D/A、GPIO、IIC、**中断和数码管**
  - A/D（ADS7852）
  - D/A
  - GPIO
  - IIC
  - **中断**
  - **数码管（**2 **个）**

- EEPROM

- DATAFLASAH 接口
- NAND FLASH（K9F2808U0C）接口
- SMART MEDIA 接口
- RS232 PORT 2/PORT3
- RS485 PORT 0/PORT1
- 网络
- 音频/触摸屏
- SD/MMC 接口
- SMART CARD 接口
- CF 卡接口
- USB 主（2）从（1）口
- 调试串口
- 串口 0/串口 1
- 红外接口
- 显示接口
- CAN 接口

（3）CAN 版

两路 CAN 总线接口，和底版以插针形式连接，以两个 9 针串口形式输出

（4）显示版

和底版以插针连接，输出 VGA 和 AV、16 位 LCD、8 位 LCD 和 4 位 LCD

2．系统的地址分配表

（1）D/A

A 路：5040 0000

B 路：5040 0004

（2）数码管

数码管 0 打开，1 关闭：5080 0020

数码管 0 关闭，1 打开：5080 0010

数码管 0 打开，1 打开：5080 0000

数码管 0 关闭，1 关闭：5080 0030

（3）A/D 地址

写启动时地址：

通道 0： 5240 0000

通道 1： 5240 0004

通道 2： 5240 0008

通道 3： 5240 000C

通道 4： 5240 0010

通道 5： 5240 0014

通道 6： 5240 0018

通道 7： 5240 001C

读数据时的地址：

5240 0000

（4）CAN0 的地址：

5080 0000（5080 0008 **为写数据**）

（5）CAN1 的地址：

50C0 0000（50C0 0008 **为写数据**

（6）USER1 基地址

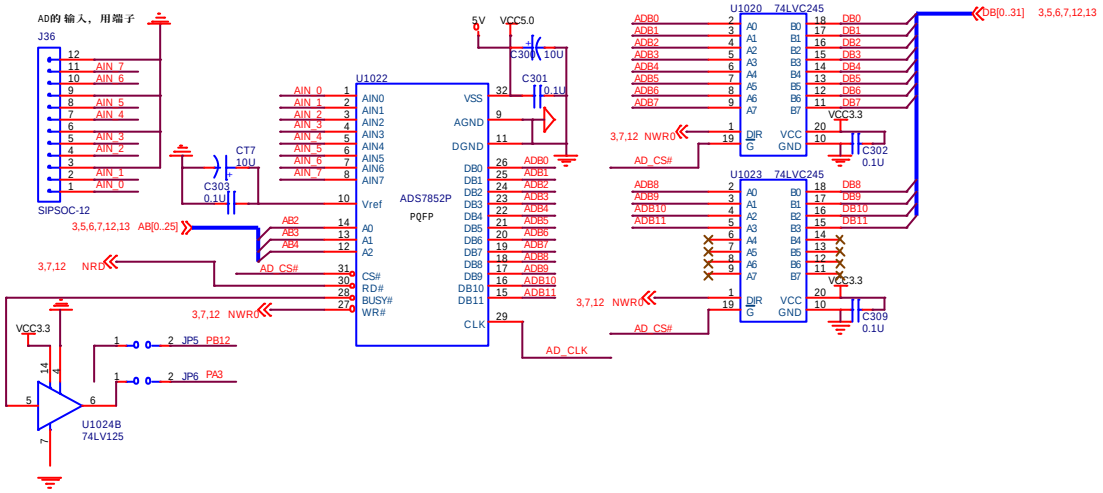
5280 0000

（7）USER2 基地址

52C0 0000

3．A/D 和 D/A

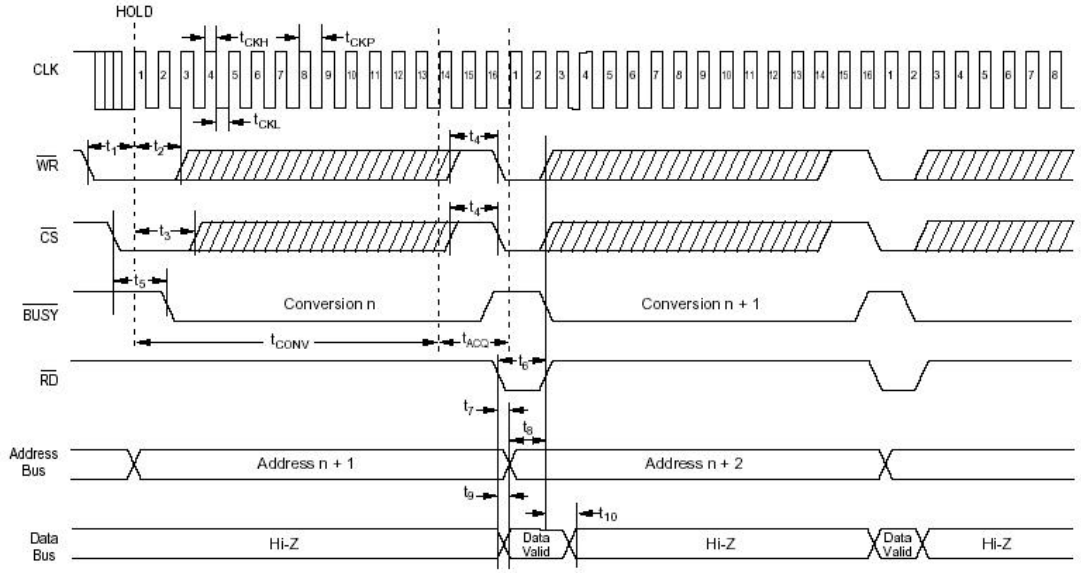
（1）A/D 部分



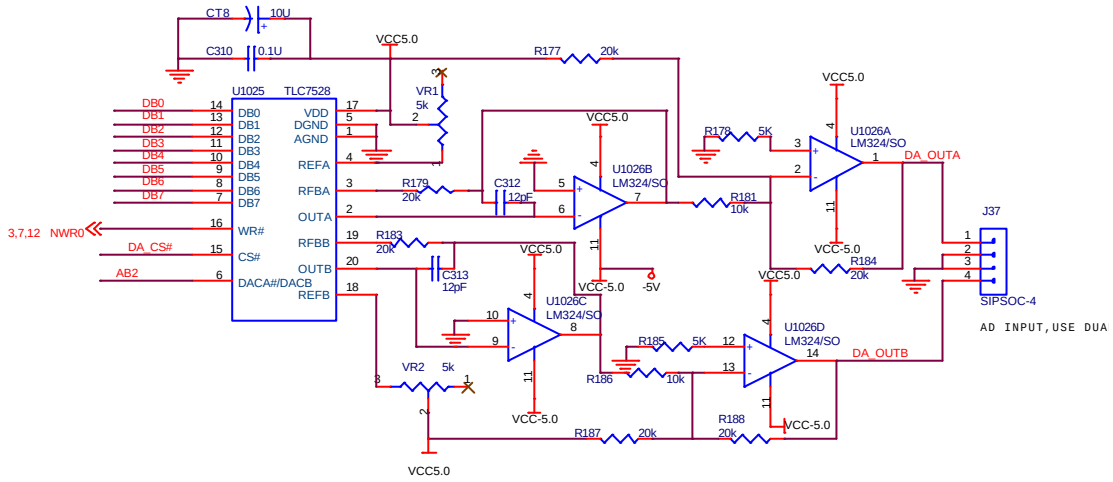
特点：

- **8 路模拟信号输入；**
- **采样率 500ksps，分辨率 8 位；**
- **和 AT91RM9200 的接口采用 3V/5V 转换芯片 74LVC245；**
- **A/D 转换时钟的提供有两种方式：直接晶振和 AT91RM9200 的定时/计数器模式，共 6 种接口可选，见电路原理图；**
- **A/D 转换完成信号通过 5/3V 转换，接到 PB12 和 PA3 可选，以查询或者中断方式皆可；**

时序图：



（2）D/A 部分



特点：

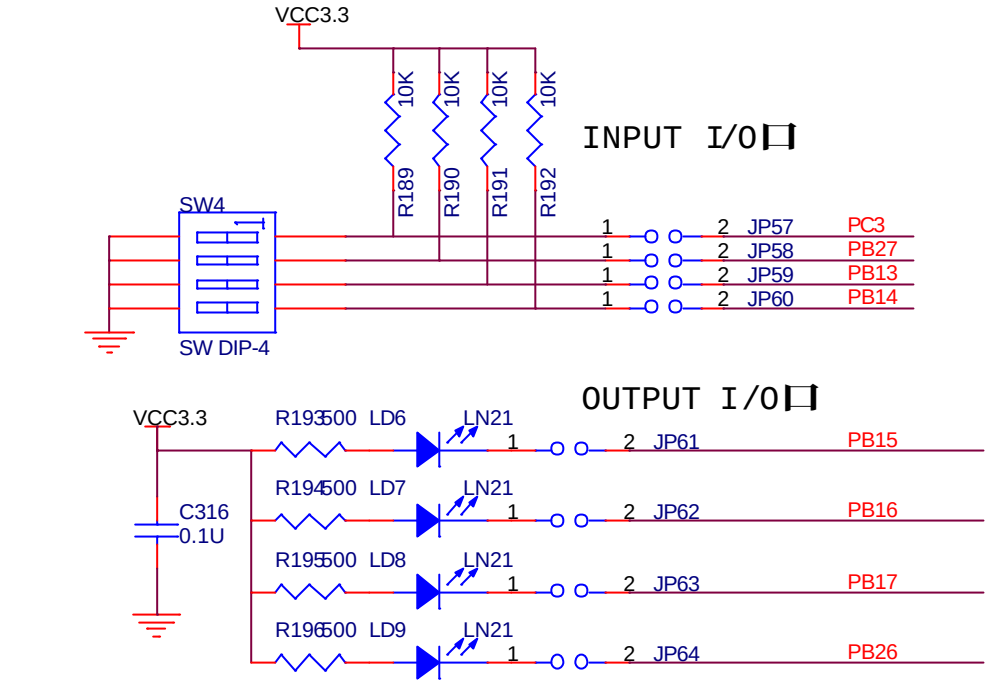
- 两路 D/A
- 输出的电压范围-5V 到+5V
- 8 位分辨率

4．GPIO 和 IIC

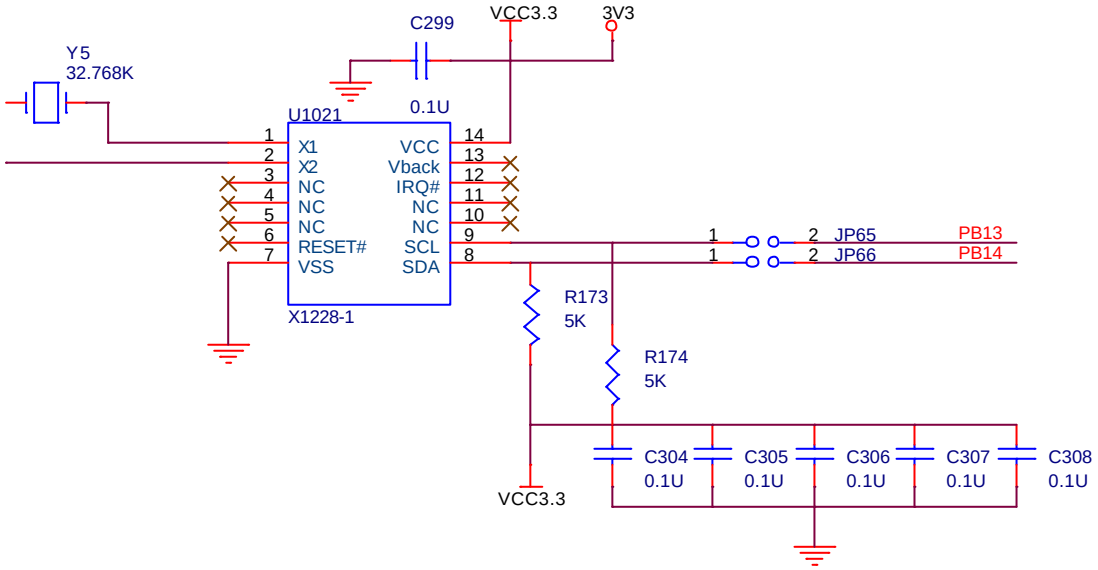
特点：

- 4 路输入 I/O，思路输出 I/O；
- 输入 I/O 通过拨档开关进行输入的 0/1 切换；
- IIC 芯片采用 x1228，兼容日历/时钟/复位/看门狗
- IIC 采用虚拟 IIC 的接口控制：PB13 和 PB14

GPIO 电路

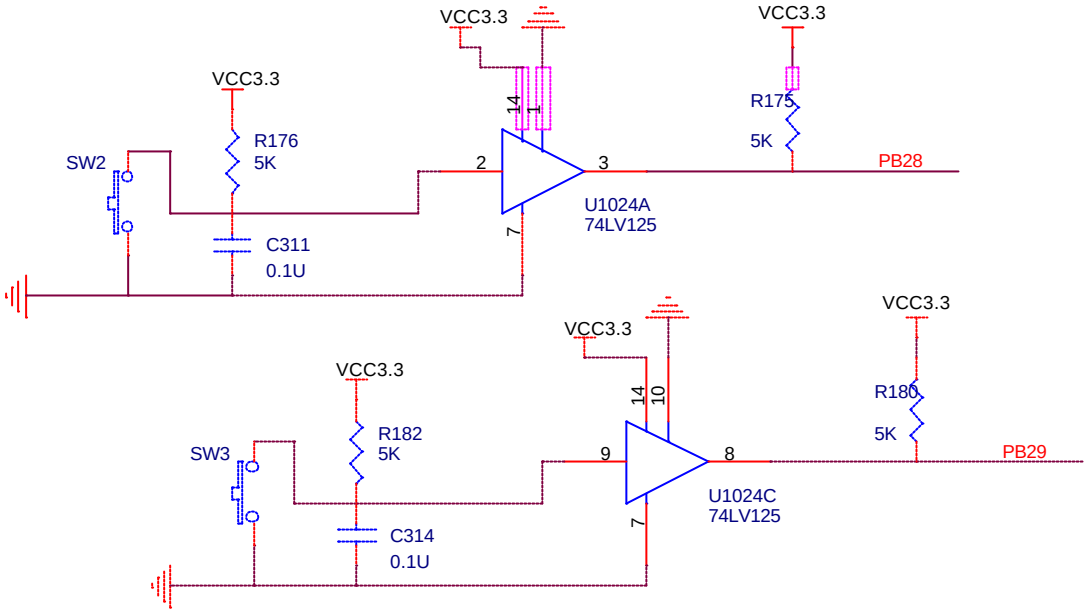


IIC 电路



5．系统中断介绍

9200 具有 31 级中断，其中还有一个 FIQ 中断，FFT-RM9200 上面设计了中断实验，通过按键可以进行中断级别的实验，电路采用 PB28 和 PB19，可以实现 I/O 中断、IRQ 中断、FIQ 中断、中断级别实验



6．系统的存储卡接口

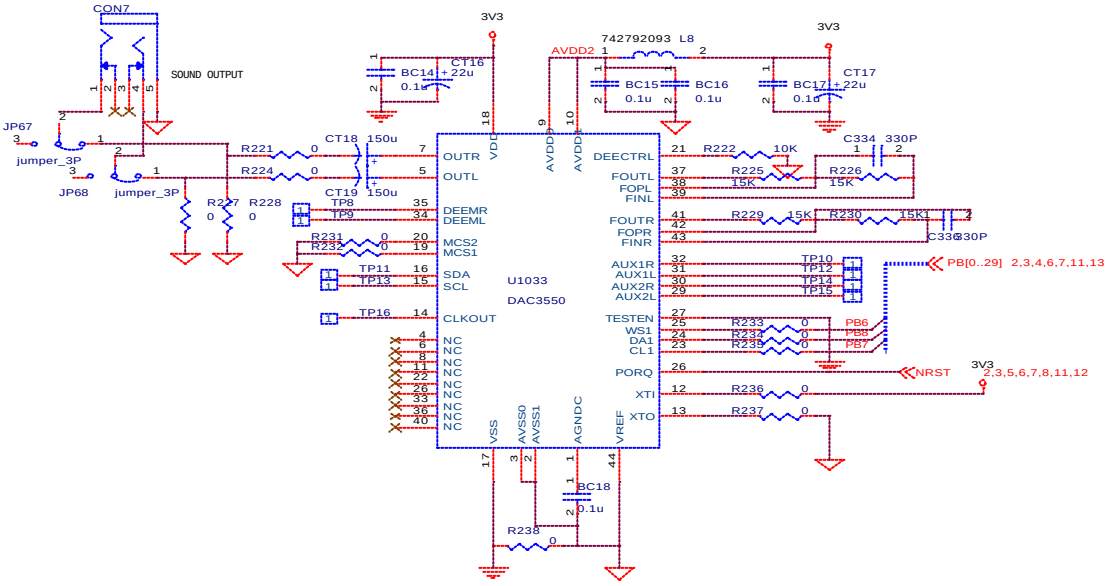
FFT-RM9200 有三种存储卡：SD 卡，SMC 卡和 CF 卡。还有一个为运行 WINCE 予留的 DATA FLASH 卡。

详细的情况先参见原理图。

7．音频/触摸屏接口

FFT-9200 集成了两个音频的芯片：DAC3550 和 USB1400，用户可以根据自己的要求进行选择。DAC3550 只有音频输出，USB1400 集成了音频输入、音频输出、还有一个触摸屏接口。

我们此处介绍 DAC3550 部分，对于 USB1400 部分，请注意 FFT 发布的补丁通知。



在使用时候，注意跳线的设置

8．CAN 总线和 485 总线

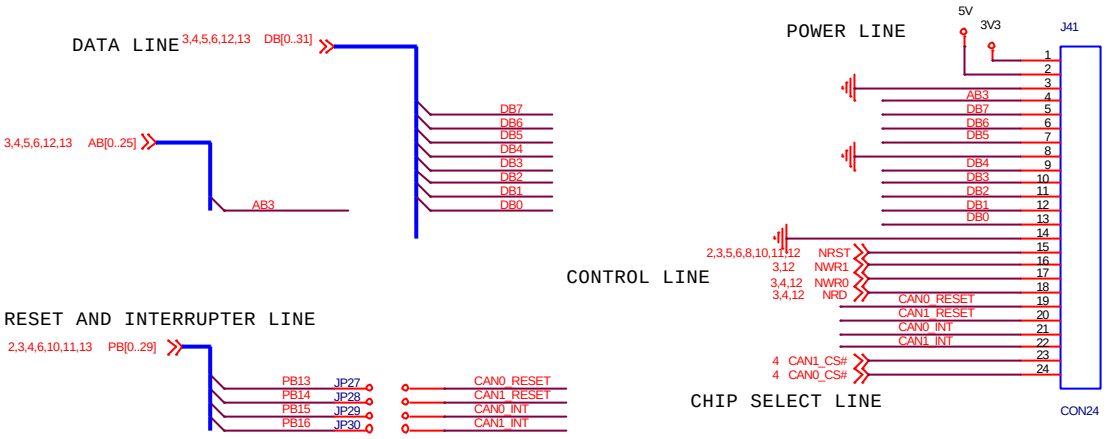
（1）485 总线：2 路

FFT-RM9200 的 485 总线和 RS232 接口的串口 2/串口 3 合用，通过跳线进行选择，当然也可以同时使用。详细参见原理图。

（2）CAN 总线：2 路

FFT-RM9200 具有 2 路独立的 CAN2.0 总线接口，全速运行，最高可以达到 1Mbit，在电路上采用分离的模式，底版上只留出插座的接口。

CAN 接口：



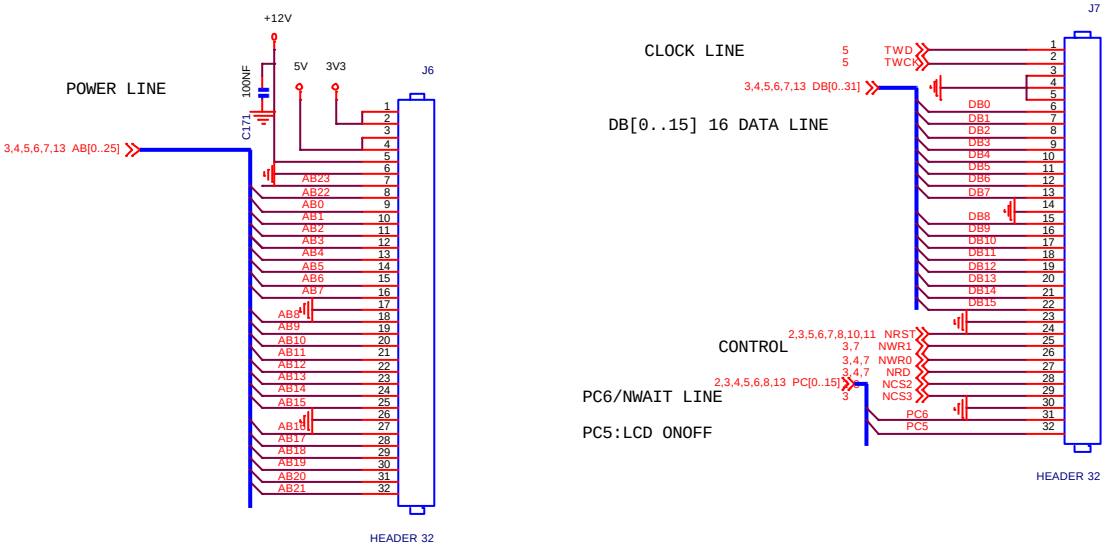
主要有：电源线，数据线，地址线，控制线、中断线和复位线

CAN 的详细电路：

参见详细的电路原理图

9．显示接口

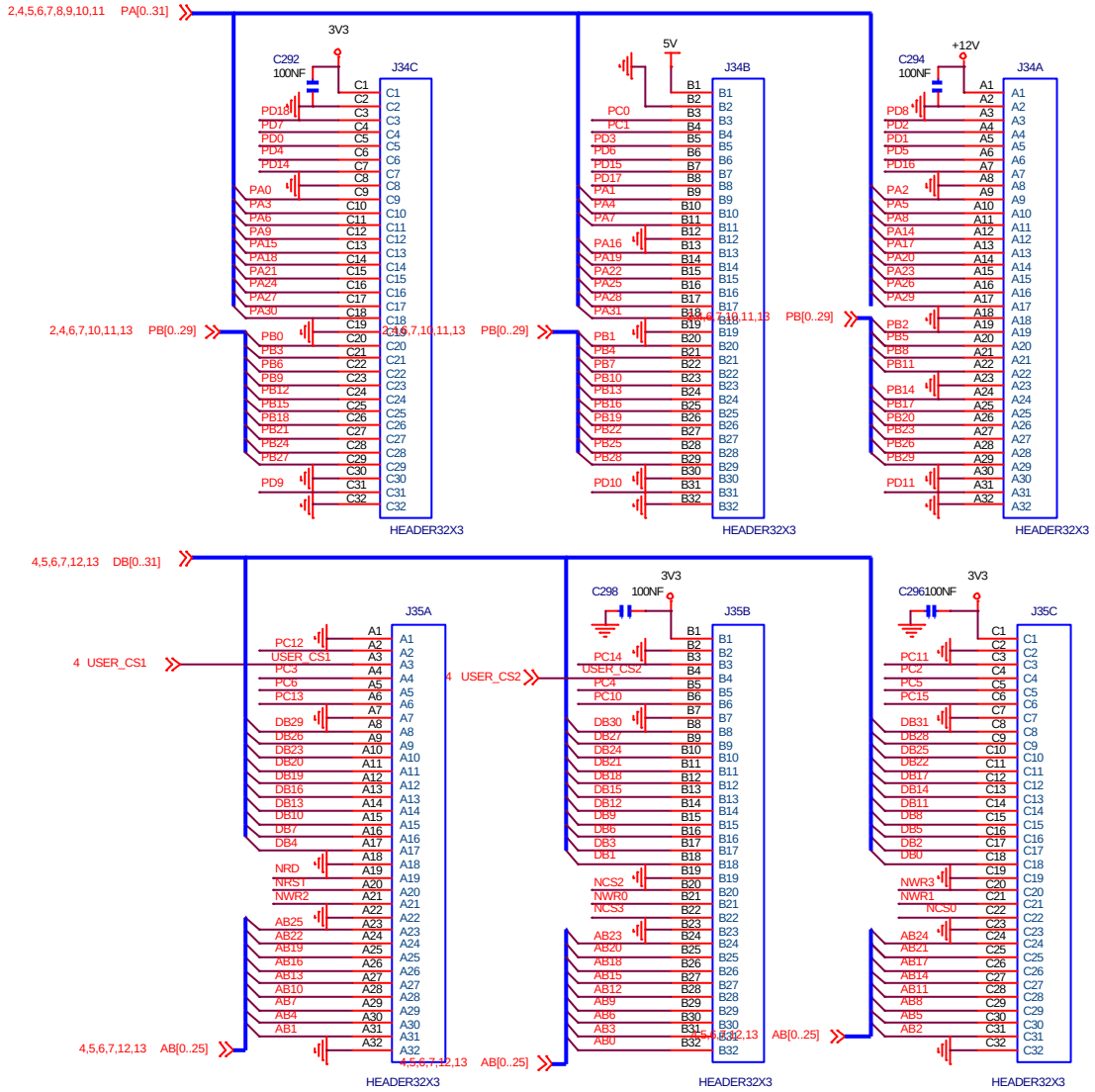
FFT-RM9200 的显示部分，在电路上采用分离的模式，底版上只留出插座的接口。



同样：有数据线，地址线、控制线、复位线等等
详细的显示板卡，参见电路原理图

10． 系统的用户扩展接口

FFT-RM9200 **给用户留出丰富的扩展接口，包括数据线、地址线、**I/O **线、电源线和片选线，用户可以方便进行各种总线扩展和** I/O **扩展。**



第 4 章　U-BOOT 介绍--开发 FFT-RM9200 的入门砖

内容概要

U-Boot 是一个非常复杂的东西，它也许体现了嵌入式系统的一个非常重要的特征：自己定制，他脱胎于 PC 机的 Linux，但是需要你关注更多，我们提供 U-BOOT1.0.0 原码，你也可以在网站上直接下载，或者找到更新的版本，本章主要介绍 U-boot 情况，版本、下载和应用，而对于 U-BOOT 的编译、修改请按照 MAKEFILE 文件在 Linux 下完成。

1．U-BOOT 能够干什么？

在 FFT-RM9200 的第一个版本上，我们提供 Thoundboot（ZooBoot），当然你现在还可以继续使用，我们建议你升级为 U-BOOT。

U-BOOT 和其他任何 BOOTLOADER 都是一样的，主要是对系统进行初始化、系统引导、FLASH 操作等功能。FFT-RM9200 的 U-BOOT 主要是对板子的硬件进行初始化，包括：时钟和 PLL、定时器、调试串口（Debug UART）等等，具体请看原程序。有了 U-BOOT，我们可以在主机的超级终端通过调试串口和目标机进行通信和设置

2．傅立叶提供的资源。。。

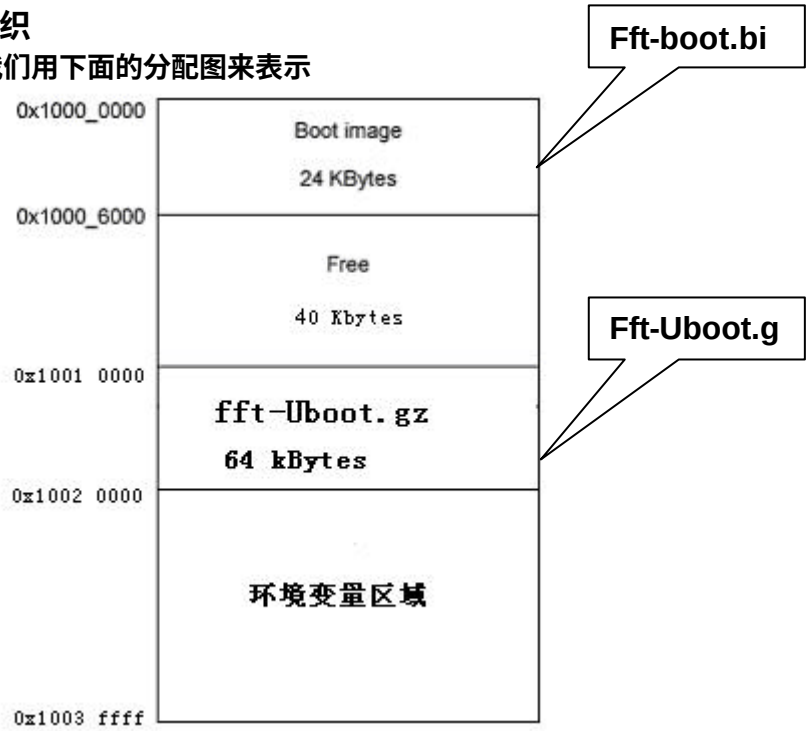
关于 U-BOOT，傅立叶提供如下

- 包含 U-BOOT、BOOT 和 LOADER 的原代码
- 包含编译好的 fft-loader.bin、fft-Uboot.bin、fft-boot.bin、fft-Uboot.gz 文件的目录
- 包含 U-BOOT 如何使用的文献

3．你应该更为详细的了解 U-BOOT

3．1 U-BOOT 的组织

对于 U-BOOT，我们用下面的分配图来表示



基本的 bootstrap 为 fft-boot.bin，必须驻留在 AT91RM9200 的 Flash 地址处 NCS0，0x10000000

3．2 编译 U-BOOT

建议你如果没有特殊需要，不需要编译，但是如果你建造你的特殊要求的板子，在阅读本说明的基础上，还需要详细阅读 U-BOOT 的 readme、makefile 和原代码

3．2．1 编译 U-BOOT

光盘中给你提供了一个编译的版本，这样你可以直接使用 mkimage，当然在 Linux 系统下，你也可以重新编译，用下面的命令

```
> cd （U-Boot目录）                    ；进入目录
> make at91rm9200dk_config       ；编译
> make all
> gzip -c u-boot.bin > u-boot.gz    ；压缩为gz文件
```

3．2．2 编译 boot.bin

```
> cd （Boot目录）
> make
```

3．2．3 编译 loader.bin

```
> cd （Loader目录）
> make
```

3．3 下载 U-BOOT

请详细阅读第二章的第四节，在超级终端的 FFTUboot>提示符下，直接下载 fft-boot.bin 和 fft-Uboot.gz

4．对于更多的需求，你需要做什么？

当然 U-BOOT 还是比较复杂的，如果你只是一个 ARM 的应用者，我觉得足够了，如果你是对产品的开发细节非常关注的工程师，我想你可能会更去读原码，请下载 U-BOOT 的原代码，你才会了解更多，你如果更有兴趣，可以下载 U-BOOT 的各种版本，进行阅读和开发。

第 5 章　采用 ADS 和 FFT-ICE 开发 FFT-RM9200

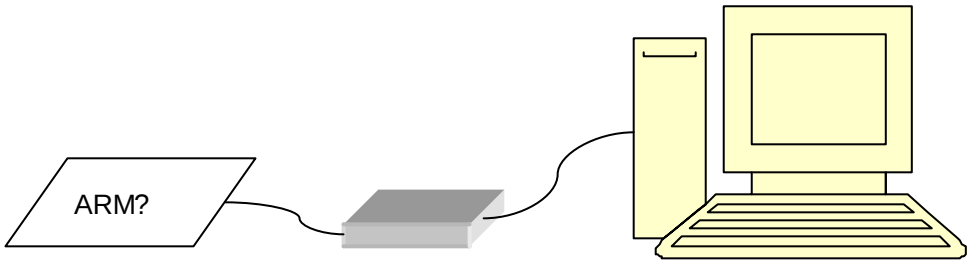
内容简介：

本章主要介绍在不采用操作系统的情况下，利用 ADS 和 FFT-ICE 开发 FFT-RM9200

1．了解 ADS 和 FFT-ICE

ADS 俗称 ARM Developer Support，是 ARM 公司推出的集成编辑、编译和调试工具，用来替代前一版的 SDT 开发工具；FFT-ICE 是傅立叶 ARM 技术研发部开发的一款 ARM 仿真机，支持 ARM 公司的 ARM7、ARM9、ARM10、StrongARM　Xscale 等 ARM 内核。

ADS 主要由三个部分组成：Multi-ice Server（连接工具，用于识别 ARM 内核），Codewarrior（集成编辑、编译和链接工具）和 AXD（调试工具）。FFT-ICE 是一个硬件仿真机，一端是并口，接计算机，另一端是 JTAG，接 AT91RM9200 的目标板，连接图如下



对于 ADS 和 FFT-ICE 的更多了解，请查看傅立叶公司的 FFT-ICE 产品说明书

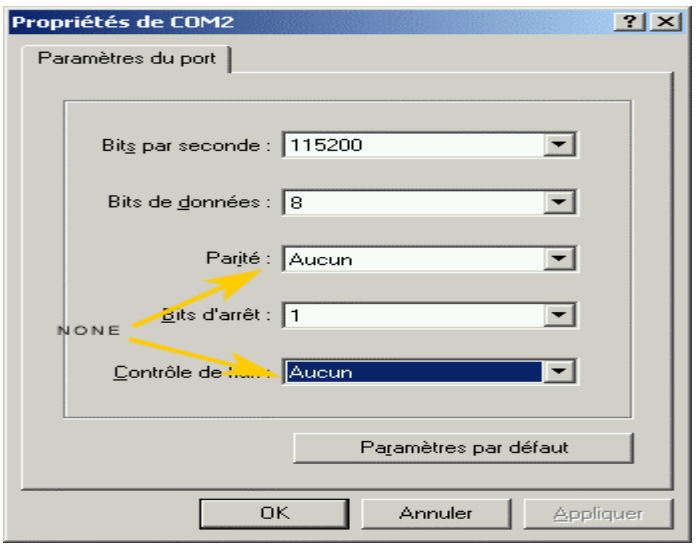
2．采用 ADS 和 FFT-ICE 开发 FFT-RM9200

和所有的 ARM 内核的芯片开发一样，采用 ADS 和 FFT-ICE 开发 FFT-RM9200 如下

2．1 开发环境设置

① 参考 2，确定和安装 FFT-ICE 以及 ADS，连接和设置 FFT-ICE；

② 连接 FFT-RM9200 上的串口 1（调试串口）到计算机的串口，然后打开计算机的超级终端 HyperTerminal，设置串口如下（115200，8，无，1，无）：

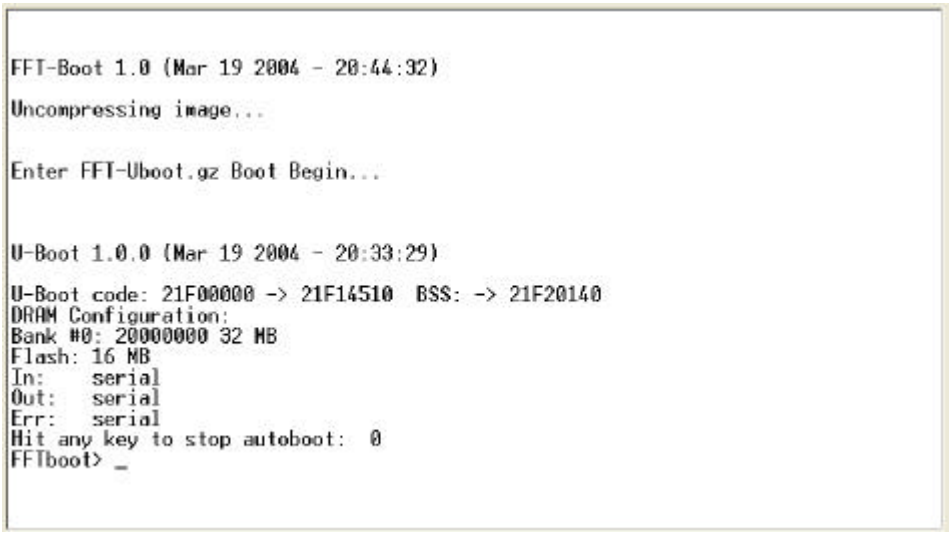


确定后显示超级终端的接收界面

2．2　系统初始化代码的编译、下载和监控

STEP 1：

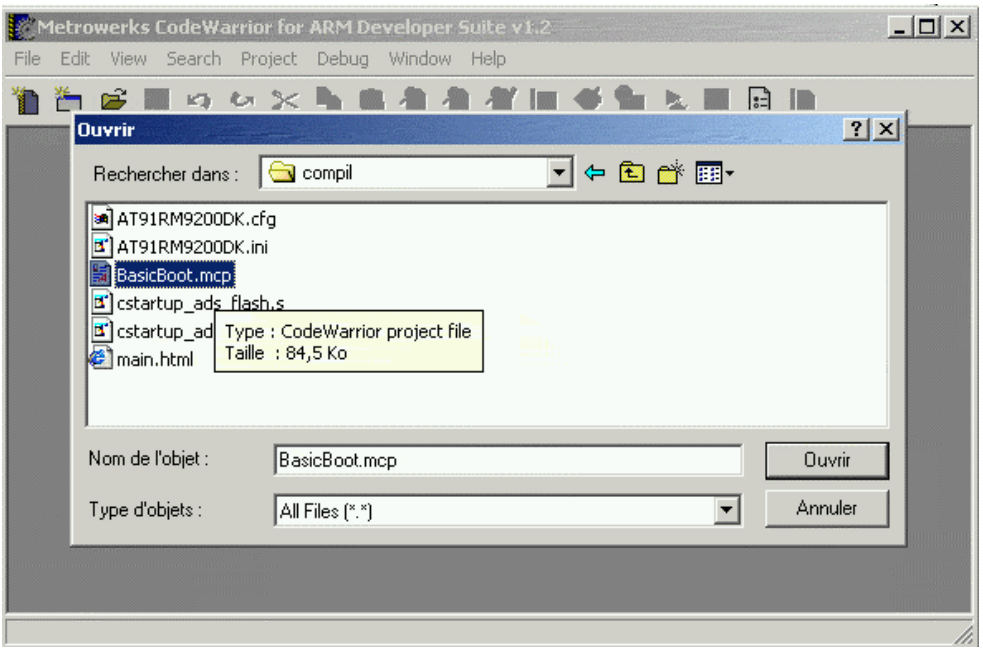
确认开发环境设置完毕，目标板加电或者，系统启动，超级终端显示如下的内容



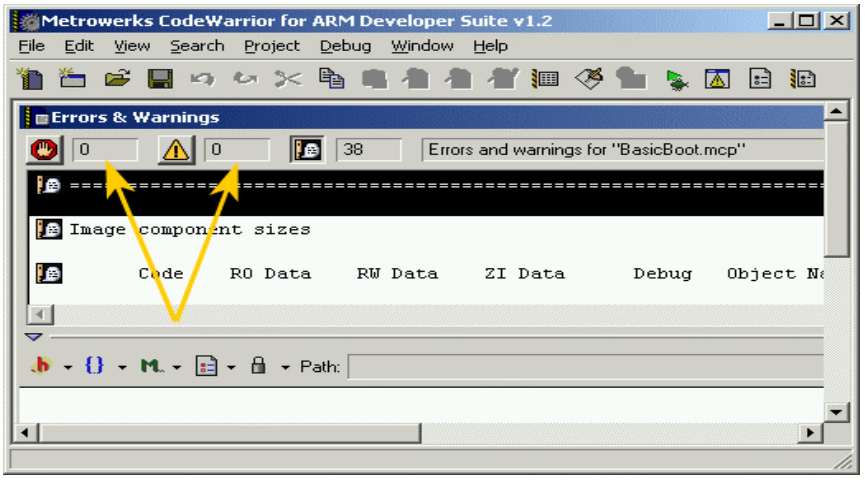
这是 FFT-RM9200 的启动程序显示

STEP 2：

打开 ADS 的 CodeWarrior 编译集成环境，打开 basicboot.mcp（具体位置，请详细参见光盘的说明），进行编译

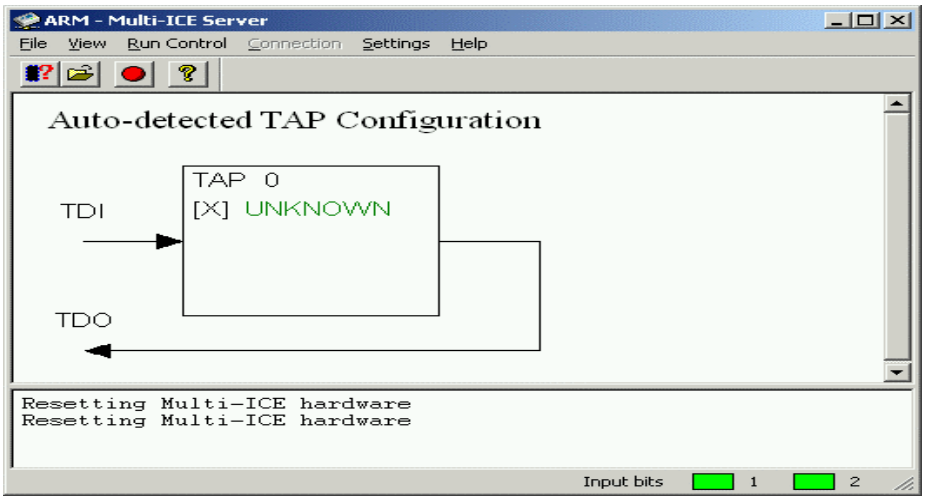


进行编译，得到下述结果

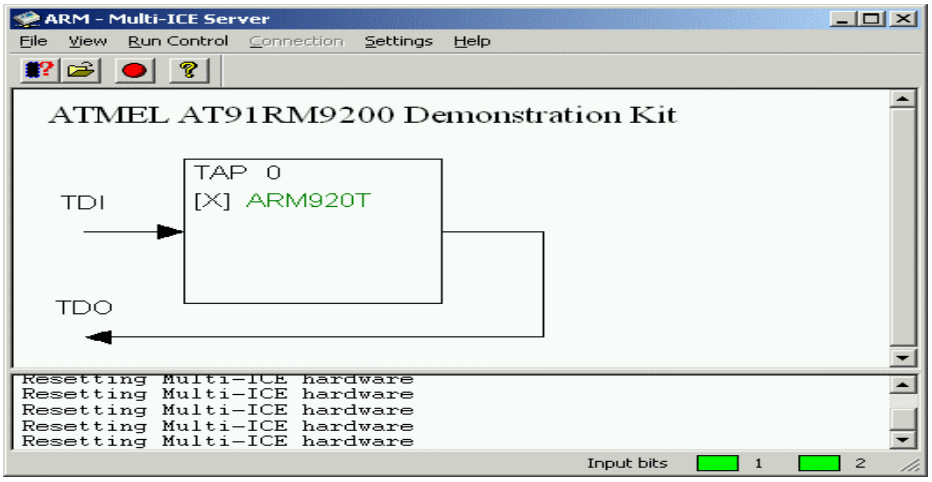


STEP 3：

打开 Multi-ICE **程序，进行连接，出现下面的结果**

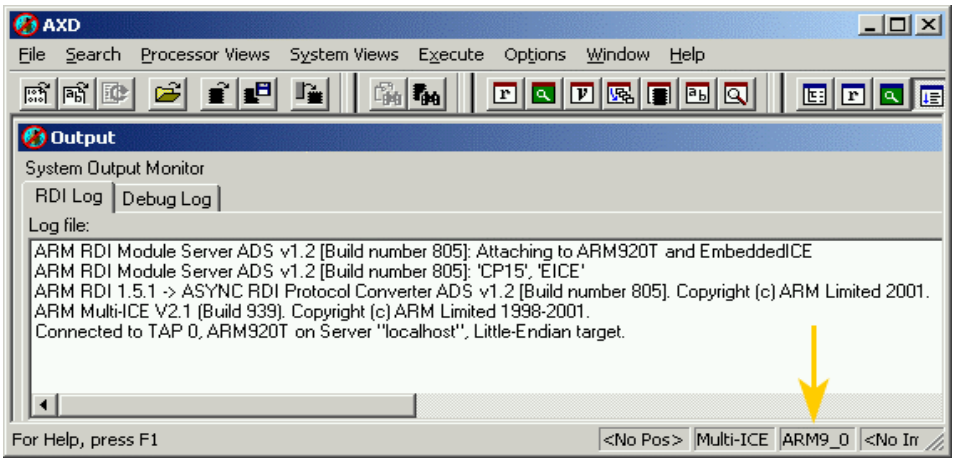


在 File **菜单下，点击** load configure**，装入** ARM **的配置文件** AT91RM9200DK.cfg**（配置文件的位置请参见光盘说明）**

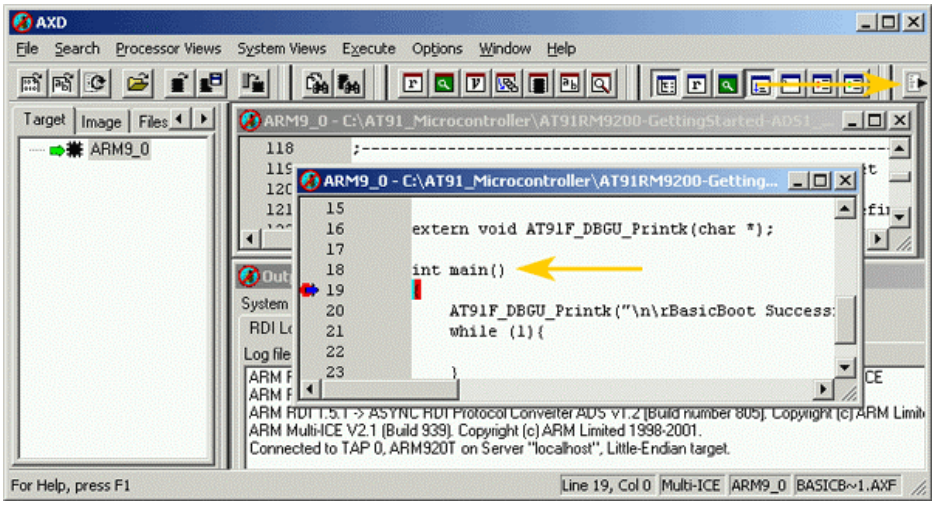


STEP 4：

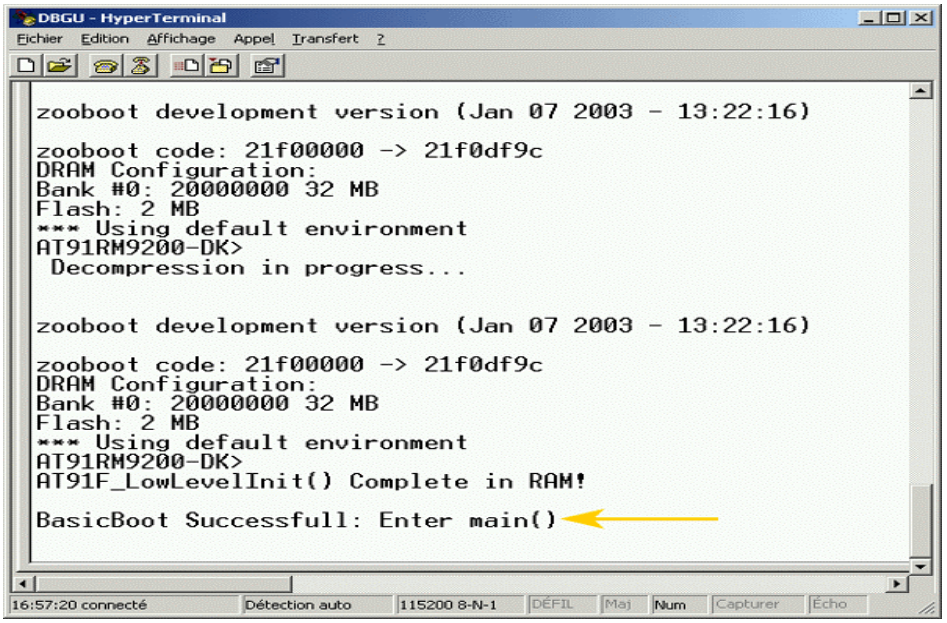
打开 AXD**，设置** target configure **为** Multi-ice**，**



在 File **菜单下** load image，**装入** Basic Boot.axf，**运行**



程序运行到 main()，**再继续运行主程序，在超级终端下，进行观察**



！程序已经正常运行

3．傅立叶公司为你提供的开发例程

3.1 经过改动 ATMEL 原例程之后的例子

“ Hello world” example :	AT91RM9200-Basic	GHS 3.6	ADS 1.2	Hello world 例子
Boot example :	AT91RM9200-BasicBoot	GHS 3.6	ADS 1.2	BOOT 例子，包括初始化
Parallel Flash example :	AT91RM9200-BasicFlash	GHS 3.6	ADS 1.2	对 FLASH 的读写、擦除
Graphic display example :	AT91RM9200-BasicGraphicDisplay	GHS 3.6	ADS 1.2	图形显示例子
Processor Idle example :	AT91RM9200-BasicIdle	GHS 3.6	ADS 1.2	处理器一种低功耗模式
IRDA example :	AT91RM9200-BasicIRDA	GHS 3.6	ADS 1.2	红外通信
ISO7816 example :	AT91RM9200-BasicISO7816	GHS 3.6	ADS 1.2	ISO7816 存储卡操作
MMU example :	AT91RM9200-BasicMmu	GHS 3.6	ADS 1.2	MMU 操作
SPI dataflash example :	AT91RM9200-BasicSPIDataFlash	GHS 3.6	ADS 1.2	SPI DATA FLASH 测试
TWI EEPROM example :	AT91RM9200-BasicTWIEeprom	GHS 3.6	ADS 1.2	EEPROM 测试
UDP example :	AT91RM9200-BasicUSB	GHS 3.6	ADS 1.2	USB 从口测试
USART and PDC :	AT91RM9200-BasicUSARTPDC	GHS 3.6		PDC 数据通道和 USART 测试
Basic I2S :	AT91RM9200-BasicI2S	GHS 3.6		声卡测试
I2S Test:	AT91RM9200-TestI2S	GHS 3.6		
EMAC example :	AT91RM9200-BasicEMAC		ADS 1.2	网卡测试
MCI device example :	AT91RM9200-BasicMCIDevice		ADS 1.2	MCI 存储卡测试
Basic ROM services :	AT91RM9200-BasicROM_Services		ADS 1.2	
Basic 1st level Bootloader :	AT91RM9200-BasicROM_Services_BootLoader		ADS 1.2	
UHP example :	AT91RM9200-BasicUHP		ADS 1.2	主 USB 测试
USB Pipe example :	AT91RM9200-BasicUSBPipe		ADS 1.2	

3．2 傅立叶提供的各个接口的 ADS 下的测试例程（傅立叶 ADS 和 FFT-ICE 下测试程序 2 目录下）

测试内容	测试程序	备　　注
串口 0 测试	Usart0	
串口 1 测试	Usart1	
串口 2 测试	Usart2	
串口 3 测试	Usart3	
RS485-0 测试	485-0	
RS485-1 测试	485-1	
CAN0 测试	Can0	
IRQ0 中断测试	Irq	通过按键 SW3（IRQ0）进行测试
DAC3550 声卡测试	Sound	DAC3550 的声音输出测试
数码管测试	数码管	可以对两个数码管同时测试
输入输出 I/O 口测试	IO	
定时器输出测试	TC	用 PA19，输出 1MHz 的方波
ADC 测试	ADC	AD 输入的 0 通道测试
IIC 测试	IIC	对 AT24C512 的测试
DAC 测试	DAC	程序测试了 OUTB 通道

对于各种测试程序，请参考相应的文献

4．利用 ADS 和 FFT-ICE，你还需要什么？

本章讲述的是不采用操作系统的情况下如何开发 FFT-RM9200 的问题？你可能还有很多的问题需要搞清楚，比如 ADS 的使用、各种文献和硬件的熟悉与阅读，还有各种测试程序下的启动程序等等，我觉得在傅立叶把你引入门后，剩下的就看你的产品开发本身了。对于这些问题，你需要慢慢的熟悉，仔细阅读说明书和资料。

但重要的是，你已经“跑”起来了。

第 6 章 嵌入式 Linux 在 FFT-RM9200 上的开发

内容点击

本章我们首先了解嵌入式 linux，介绍在 FFT-RM9200 上进行 Linux 开发，我们需要什么，如何去做？如何下载？

1．应该对嵌入式 Linux 的开发有个大致的了解

当然，采用 Linux 进行 FFT0RM9200 的开发，你应该首先对 Linux 有个大致的了解，如果你具备嵌入式 Linux 的开发经验，可以跳过本节，否则，你需要了解下面几个内容：

- 了解 Linux 操作系统的一些基本内容，比如：linux 的启动过程，Linux 的原码和开放性、Linux 的命令，Linux 的开发工具和环境；
- 你尝试能够在 Linux 下进行 C 语言程序的简单开发，比如“Hello World”程序，Linux 的编译工具 gcc，Makefile 文件等等，如果可以进行 gdb 调试更好；
- 你了解了 Linux 后，应该了解嵌入式 Linux 开发是宿主机—目标机（HOST-TARGET）交叉开发，这样你的系统编译工具要换成交叉编译工具，对于这个，你只需要指明它的路径即可，需要交叉开发环境，并且建立交叉编译环境；
- 嵌入式 Linux 系统下的 BootLoader；
- 目标板各种资源在 Linux 下的驱动程序；
- Linux 的内核和文件系统的裁剪和编译；
- 当然最后一点就是如何下载了。

我们在此不对嵌入式 Linux 做非常详细的介绍，主要介绍交叉编译环境，BootLoader、Linux 内核和文件系统、下载。

1．1 了解 BootLoader

在 FFT-RM9200 上，我们采用的是 U-BOOT，这部分内容我们在第 3 章已经进行了介绍。U-BOOT 在嵌入式系统中相当于 PC 机的 BIOS 加上操作系统引导头部的内容 ，并且引导操作系统进行装载和运行，U-BOOT 启动后有一系列的命令，使得我们能够方便地对 FLASH、RAM 进行操作，U-BOOT 已经对系统的频率、定时器进行了设置，初始化了一个调试串口，我们可以通过串口或者以太网进行数据的下载。

go	- 在地址 'addr' 处开始程序执行
run	- 运行命令
bootm	- 从内存中进行应用程序影象运行
bootp	- 通过网络用 BootP/TFTP 协议来启动影象
tftpboot	- 通过网络用 TFTP 协议、设置服务器和客户机的 IP 地址进行影象文件传送
loadb	- 通过串口线(kermit mode) 来装载二进制文件
printenv	- 打印环境变量
setenv	- 设置环境变量
saveenv	- 保存环境变量到内存

下面是 U-BOOT 中的简单环境变量

baudrate 波特率
bootdelay boot 延迟
bootcmd Boot 命令
bootargs Boot 参数
bootfile
ipaddr 客户机 IP 地址
serverip 服务器地址
loadaddr 装载地址
ethaddr 网卡 MAC 地址

如果想了解更为详细的情况，请参照第 3 章的内容和详细的英文文档

1．2 常用的命令

- go - start application at address 'addr'
- run - run commands in an environment variable
- bootm - boot application image from memory
- bootp - boot image via network using BootP/TFTP protocol
- tftpboot- boot image via network using TFTP protocol
- and env variables "ipaddr" and "serverip"
- (and eventually "gatewayip")
- rarpboot- boot image via network using RARP/TFTP protocol
- diskboot- boot from IDE devicebootd - boot default, i.e., run 'bootcmd'
- loads - load S-Record file over serial line
- loadb - load binary file over serial line (kermit mode)
- md - memory display
- mm - memory modify (auto-incrementing)
- nm - memory modify (constant address)
- mw - memory write (fill)
- cp - memory copy
- cmp - memory compare
- crc32 - checksum calculation
- imd - i2c memory display
- imm - i2c memory modify (auto-incrementing)
- inm - i2c memory modify (constant address)
- imw - i2c memory write (fill)
- icrc32 - i2c checksum calculation
- iprobe - probe to discover valid I2C chip addresses
- iloop - infinite loop on address range
- isdram - print SDRAM configuration information
- sspi - SPI utility commands
- base- print or set address offset
- printenv- print environment variables
- setenv - set environment variables
- saveenv - save environment variables to persistent storage

- protect - enable or disable FLASH write protection
- erase - erase FLASH memory
- flinfo - print FLASH memory information
- bdinfo - print Board Info structure
- iminfo - print header information for application image
- coninfo - print console devices and informations
- ide - IDE sub-system
- loop- infinite loop on address range
- mtest - simple RAM test
- icache - enable or disable instruction cache
- dcache - enable or disable data cache
- reset - Perform RESET of the CPU
- echo- echo args to console
- version - print monitor version
- help- print online help
- ? - alias for 'help'

1．3 交叉开发环境的建立

你可以有两种选择，一种是采用已经编译好的开发环境，你只需要解开到安装的位置，另一种就是自己下载原码、编译和建立开发环境，对于注重于应用层的工程师，建议采用第一种方法即可，因为一般开发商都帮你提供好了。

1．3．1 安装预先编译好的开发环境

你可以在 FFT-RM9200 的光盘中得到预先编译的交叉开发环境 cross-2.95.3.tar.bz2，当然你也可以从网站 ftp://ftp.arm.linux.org.uk/pub/armlinux/toolchain/上下载，当然需要编译了，网站上一般提供原码 ，主要包含下面内容

- Bin utilities
- ARM linux C compiler and linker
- glibc Library
- ARM linux C++ compiler

当拿到压缩文件后，你需要安装，傅立叶的安装位置在/usr/local/下，当然你可以安装到任何地方，但是要记住地址。

```
bash$ su                                    变为超级用户
Password:
bash# cd /usr/local
bash# mkdir arm                             建立 ARM 目录
bash# cd arm
bash# tar Ixvf cross-2.95.3.tar.bz2         解开
```

当然也可以简单用图形界面完成这一切，只要把原码压缩包解开即可。

这样你的开发环境已经建立在/usr/local/arm/2.95.3/bin 下面，当然你需要使用的时候，应该指出编译器的位置

1．3．2 自己制作交叉开发环境

当然，你需要自己下载原码，并且阅读其中的 readme 和 makefile，按照过程自己进行编译，最后安装到具体位置。这样的文章到处都是，此处不作具体介绍。

1．4 内核与文件系统

当你完成上面两步时，你所要做的就是对内核和文件系统进行配置了，当然这两部分是嵌入式 Linux 的核心内容，我们放到下面做详细介绍，在本章，请你继续向下，下载我们已经编译好的内核与文件系统影象

2．嵌入式 Linux 在 FFT-RM9200 上的移植

本章我们介绍两种移植和启动模式，网络服务器的启动模式和 FLASH 启动模式，前者应用于网络模式，内核与文件系统的影象文件存放在 PC 服务器上，系统加电或复位后，通过 TFTP 协议把操作系统的内核与文件系统直接传输到系统的 SDRAM 中执行；后者的内核与文件系统放在 FLASH 中，加电与复位后，从 FLASH 中执行，属于单机模式。

2．1 网络服务器的启动模式

首先你的 U-BOOT 已经正常启动，这时我们应该从光盘上找到 Linux 的内核与文件系统的影象(usb1-uimage 和 myramdisk2.gz)，然后再向下。

2.1.1 准备工作

设置硬件

- 连接串口，连接串口线在 PC 机和 FFT-RM9200 的 J23 上
- 连接网线，采用双机相连网线直接相连，或者采用 HUB 相连

设置 PC 机

- 设置串口（115200、8、无、1、无），打开超级终端进行监控
- 找到 TFTP 协议的目录，把 tftfboot 和 tftpserver 拷贝到 C 盘的根目录下
- 把内核（usb1-uimage）和文件系统(myramdisk2.gz)的影象拷贝到 tftpboot 的目录下
- 运行 tftpsrv.exe，启动 TFTP 协议

2.1.2 进行下载

有两种方式可以进行下载，串口模式和以太网模式

2.1.2.1 串口模式

```
FFTUboot> loadb 21100000
```

通过 kermit 模式下载文件系统（myramdisk2.gz）

```
FFTUboot> loadb 21000000
```

通过 kermit 模式下载内核（usb1-uimage）

```
FFTUboot> bootm  21000000                    ;从 21000000 处开始运行
```

2.1.2.2 以太网模式

STEP 1：进行网络参数设置

```
FFTUBoot > setenv ethaddr 12:34:56:78:99:aa          ；MAC 地址设置
FFTUBoot > setenv ipaddr IP 地址（缺省 192.168.0.11）      ；目标板 IP 地址
FFTUBoot > setenv serverip 服务器地址（缺省 192.168.0.55）  ；服务器 IP 地址
FFTUBoot > setenv bootdelay 5                        ；延时
FFTUBoot > saveenv                                   ;可以进行保存，如果不保存，
                                                     掉电后需要重新设置
```

STEP 2：下载

```
FFTUBoot >tftp 21000000 usb1-uimage          ；下载内核
FFTUBoot >tftp  21100000  myramdisk2.gz      ；下载文件系统
FFTUBoot >bootm  21000000                    ；开始运行
```

2.1.3 系统自动运行

FFTUBoot >setenv bootargs root=/dev/ram rw initrd=0x21100000,6000000 ramdisk_size=15360 console=ttyS0,115200 mem=32M

FFTUBoot > saveenv

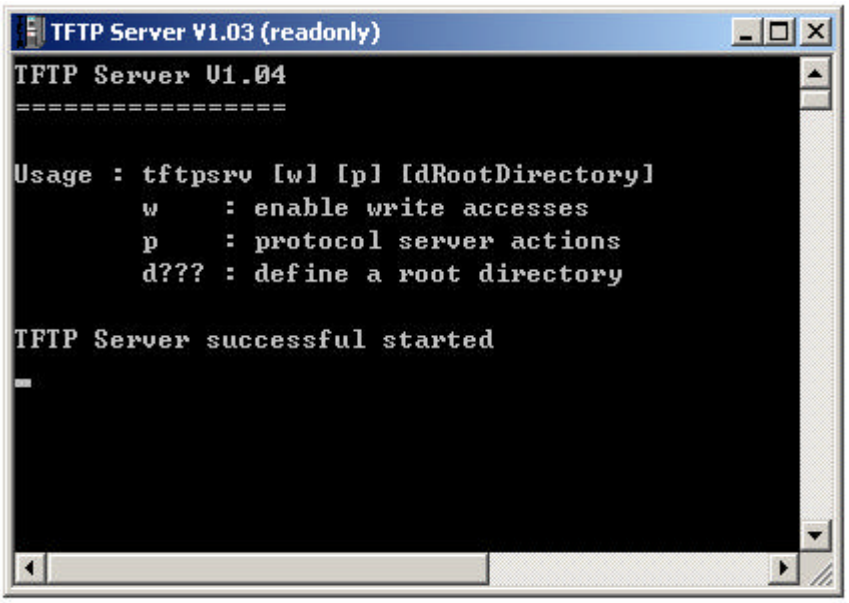
FFTUBoot >setenv bootcmd tftp 21000000 usb1-uimage\; tftp 21100000 myramdisk2.gz\;bootm 21000000

FFTUBoot > saveenv

2.1.4 系统下载和运行过程

STEP 1：

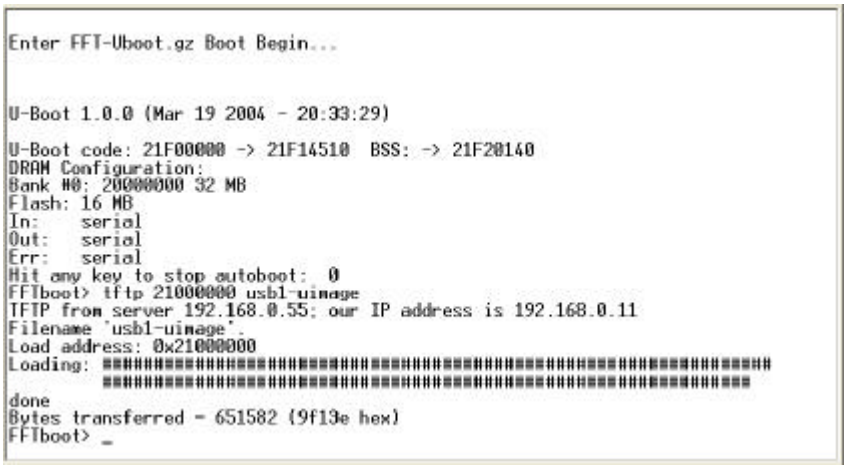
到 C:\TFTPSERVER **下，运行** TFTPSRV，**出现**



tftp **已经启动**

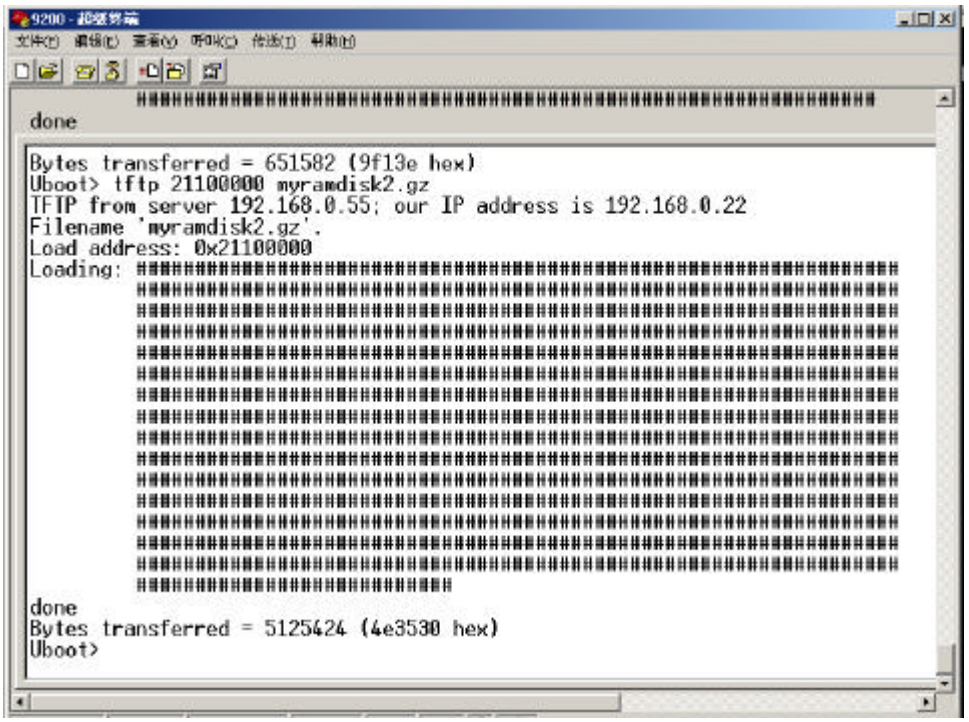
STEP 2：

下载内核，FFTUBoot > tftp 21000000 usb1-uimage



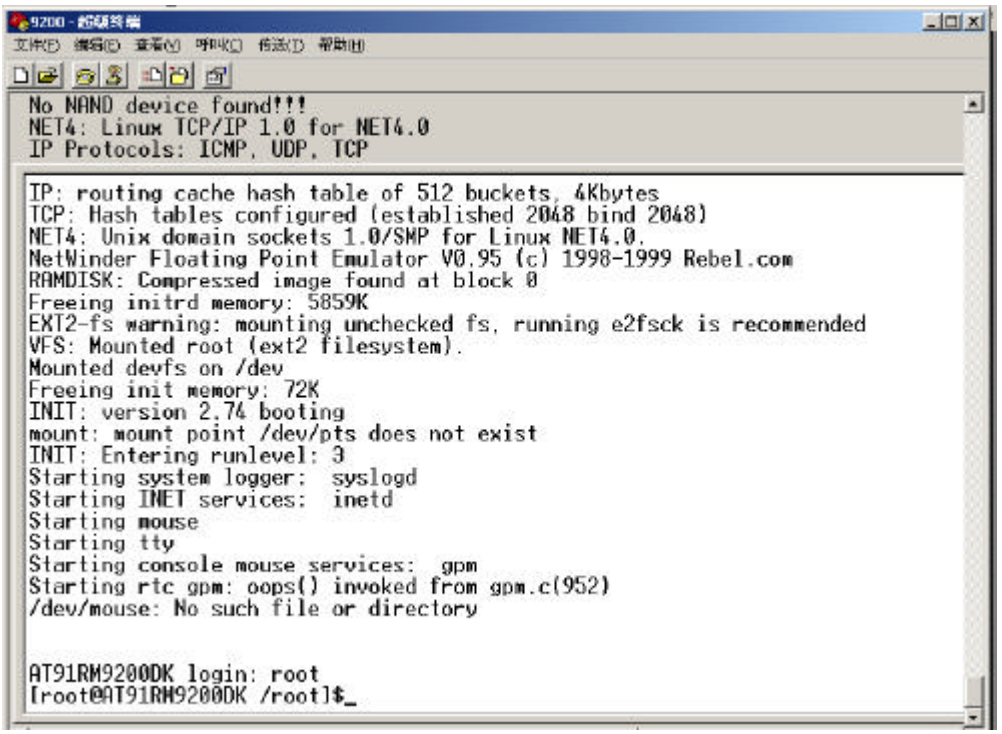
STEP 3：

下载文件系统 myramdisk2.gz，FFTUBoot > TFTP 21100000 myramdisk2.gz



STEP 4：

运行，FFTUBoot > bootm 21100000



Linux **系统已经正常启动，可以通过板子上的键盘和鼠标进行操作，**

2．2　FLASH 启动模式

当然对于一个 FLASH **启动的系统我们必然必须做两件事情，第一，我们如何下载操作**

系统到 FLASH 中；第二，操作系统如何能够正确执行。在此，我们根据系统已经烧写好的系统步骤来进行。

STEP 1：

移植烧写好 fft-boot.bin（0x10000000）和 fft-Uboot.gz(10010000)

STEP 2：设置环境变量（TFTP 协议和网络部分）

FFTUBoot > setenv ethaddr 12:34:56:78:99:aa ；MAC 地址设置

FFTUBoot > setenv ipaddr IP 地址（缺省 192.168.0.11） ；目标板 IP 地址

FFTUBoot > setenv serverip 服务器地址（缺省 192.168.0.55） ；服务器 IP 地址

FFTUBoot > setenv bootdelay 5 ；延时

FFTUBoot > saveenv ；保存网络设置变量

STEP 3：传输内核并烧写入 FLASH（0x10060000）

tftp 20000000 usb1-uimage

cp.b 20000000 ***10060000*** 内核影象大小（TFTP 传输时会看到）

STEP 4：传输文件系统并烧写入 FLASH（0x10200000）

tftp 20000000 myramdisk2.gz

cp.b 20000000 ***10200000*** 文件系统大小（TFTP 传输时会看到）

STEP 4: 设置文件系统的调用

FFTUBoot >setenv bootargs root=/dev/ram rw initrd=0x***21100000***,6000000 ramdisk_size=15360 console=ttyS0,115200 mem=32M

FFTUBoot >saveenv ；保存文件系统调用

STEP 5：设置自动启动命令

FFTUBoot >setenv bootcmd cp.b ***10200000 21100000*** 文件系统大小\;bootm ***10060000***

FFTUBoot >saveenv ；保存自动启动命令

STEP 6：系统复位和自动运行

去掉网线，重新复位，系统自动运行

3．到此时，你所想和需要的是什么？

你的 Linux 系统已经启动了，系统的硬件部分你也比较清楚了，同时你也可以随心所欲地运行其中的各种程序，但是此时你可能还有很多的疑惑之处，可能你会提出下面具体的问题，

我的内核如何来配置和编译？如何裁剪？

我的文件系统如何建立？

我的顶层程序如何编写和编译，如何加入到我的文件系统之中？

图形和网络浏览器如何？

……

其实，你不需要着急，你所想的也是大多数工程师想的，在后面的章节中，我们会一一解决这些问题

第 7 章　嵌入式 Linux 的内核开发

内容简介：

我们介绍如何开发 Linux 的内核？对于这章的一些例子，客户仅仅需要明白它的意思，不保证例子的正确性，客户也没有必要使用这个例子。通过这一章，大家应该了解 Linux 的配置、编译过程，并且能够形成最后的 Linux 影象文件，掌握自己如何加入驱动程序的方法，真正对嵌入式 Linux 的内核有深刻的了解。

1．傅立叶给你提供的内核开发资源

当然，作为产品开发商，傅立叶提供你可以对内核操作的所有资源，包括

- FFT-RM9200 **的内核原码**
- **交叉编译工具（**arm-linux-**）**
- **影象文件的形成工具（**mkimage**）**
- **必不可少的关键性说明**

这些内容的提供，使得你可以轻松面对内核的操作。

2．开发 Linux 内核的步骤

2．1 应该准备的

交叉开发环境的安装

在第 5 **章中，我们已经介绍了交叉开发环境，我们应该记住位置**/usr/local/arm/2.95.3/bin **或者其他位置。**

内核原码的安装

在光盘中找到内核的原码程序 linux-2.4.19-rmk7**，拷贝到**/usr/src/arm **下面，并且解压缩，得到内核原码，你也可以安装到别的地方。**

2．2 配置内核前的必要设置

主要在内核原码中设置 makefile **文件，主要设置两个地方** ARCH **和** CROSS_COMPILE**，**

ARCH :=arm　　**；表示目标板为** arm

CROSS_COMPILE=**交叉编译工具的地址　　；设置交叉编译工具的地址**

例如 CROSS_COMPILE= /usr/local/arm/2.95.3/bin/arm-linux-).

一般需要详细阅读 readme **文件**

2．3 内核配置

make menuconfig　　**；菜单界面**

make xconfig　　**；图形界面**

2．4 内核编译

```
make    clean
make    dep
make
```

2．5 建立内核的影象

在 linux 内核目录…./arm/linux-2.4.19/下

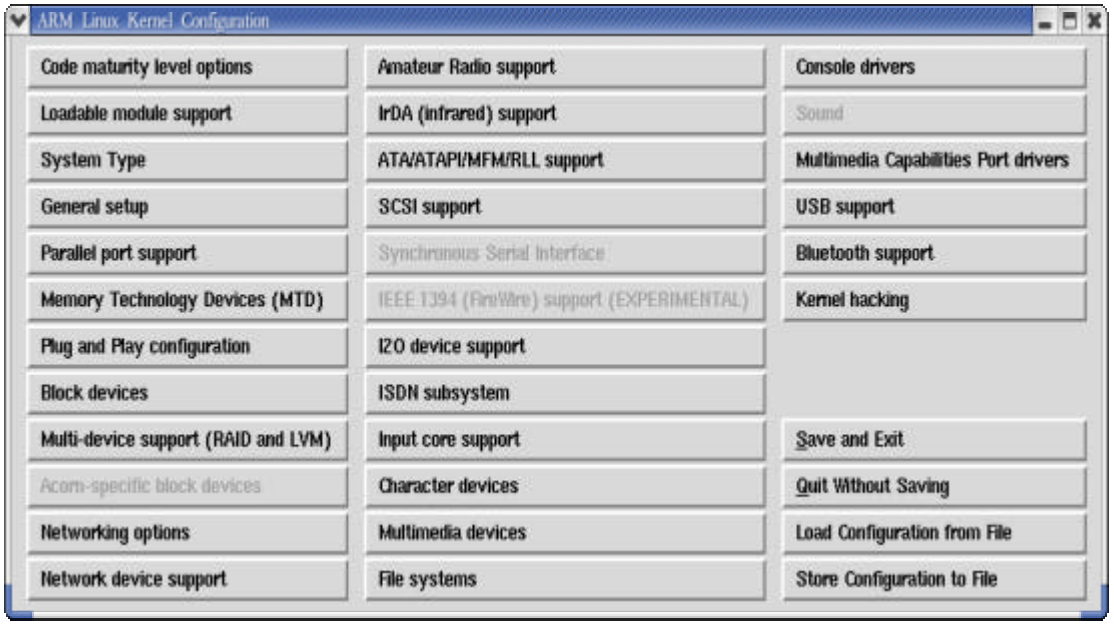
```
make Image                          ；形成 vmlinux
/usr/local/arm/2.95.3/bin/arm-linux-objcopy -O binary -S vmlinux linux.bin   ；形成
                                                                    linux.bin
gzip -v9 linux.bin                  ；形成 linux.bin.gz
```

在 u-boot 的目录…./u-boot-1.0.0/tools 下

```
./mkimage -A arm -O linux -C gzip -a 0x20008000 -e 0x20008000 -d linux.bin.gz uImage
                                    ；把 linux.bin.gz 文件变为最终的影象文件 uImage
```

3．如何配置和裁剪 Linux 的内核

这一节我们来配置 linux 的内核，在 Linux 下，用 make xconfig 或者 make menuconfig 进入配置的界面。在内核配置中，一般有四种选择：Y（选种）、N（不选）、M（模块）和数字，用户可以根据裁剪需要进行设置，最后配置完毕，选择是否对配置结果进行保存？在 FFT-RM9200 中的图形配置界面如下：



Block Devices:

- ➔ Network block device support : n
- ➔ -> Ramdisk support: y
- ➔ -> Default Ramdisk size: 15360
- ➔ -> Initial Ramdisk (initrd) support: y

File System -> Network File systems:

➔ NFS file system support : n
➔ -> NFS server support: n
Console drivers -> Frame Buffer Support:
➔ support for frame buffer devices (exp): y
➔ -> epson LCD/CR/TV controller support: y
➔ -> epson S1 D1 3806 support for AT91RM9200DK: y
➔ -> virtual frame buffer support (only for testing!): n
➔ -> advanced low level driver option: y
➔ -> 16 bpp packed pixeld support: y (others n)
USB Support:
➔ support for USB: y
➔ -> AT91RM9200 OHCI- compatible host interface: y
➔ -> USB mass storage support : y
➔ -> USB Human Interface device (full HID) support: y
➔ -> HID input layer support : y
Kernel configuration when using NFS ramdisk:
Below the details of the parameters when typing: make xconfig:
General setup:
-> Default kernel string : Erase its contents
Block Devices:
➔ Network block device support : y
➔ -> Ramdisk support: n
File System:
➔ Quota support: n
➔ -> Kernel automounter support: y
➔ -> DOS FAT fs support: y
➔ -> VFAT (Windows 95) fs support: y
➔ -> Journalling flash file system v2 (JFFS2) support: 0
➔ -> /proc file system support: y
➔ -> /dev file system support (EXP): y
➔ -> Automatically mount a boot: y
➔ -> Second extended fs support: y
File System -> Network File systems:
➔ NFS file system support : y
➔ -> provide NFSv3 client support: y
➔ -> root file system on NFS: y
➔ -> NFS server support: n

当然，更多的东西，请你恭身入局，实际在 linux **下看看，你才会真正了解**

4．如何添加新的内核配置和驱动

傅立叶公司为你提供了上述第 3 **节中的驱动和配置，对于很多客户，可能只需要进行裁剪，去掉自己不需要的内容，然后重新编译内核，形成自己的内核影象文件，而对于一些构**

造复杂系统的客户，可能还需要在原来的基础上添加新的设备和驱动程序，比如：你可能需要增加一个标准并口的支持，以挂接打印机，可能增加 IDE 接口，对系统挂接标准硬盘，当然，还有很多客户开发属于自己接口的各种驱动程序等等，这时，我们在裁剪内核的基础上，可能就需要添加内核和驱动程序。

实际上，不论是操作系统还是用户的上层程序，实际上都是程序代码，可以这么说：你可以编写从底层操作系统到上层实现的一系列流程。当然如果你的设备要求不高，你可以把你的驱动初始化就直接写到你的上层程序中，当然，你也可以把两者分开，但是这样你肯定会花更多的精力，我们在此介绍如何添加新的内核配置和驱动？？

驱动程序的使用可以按照两种方式编译，一种是静态编译进内核，另一种是编译成模块以供动态加载。由于嵌入式 Linux 不能够象桌面 Linux 那样灵活的使用 insmod/rmmod 加载卸载设备驱动程序，因而这里只介绍将设备驱动程序静态编译进 linux 内核的方法。下面以嵌入式 Linux 为例，介绍在一个以模块方式出现的驱动程序 myshebei.c 基础之上，将其编译进内核的一系列步骤：

STEP 1： 改动 myshebei.c 源带代码

第一步，将原来的：

```
#include <linux/module.h>
#include <linux/version.h>
char kernel_version[]=UTS_RELEASE;
```

改动为：

```
#ifdef MODULE
#include <linux/module.h>
#include <linux/version.h>
char kernel_version[]=UTS_RELEASE;
#else
#define MOD_INC_USE_COUNT
#define MOD_DEC_USE_COUNT
#endif
```

第二步，新建函数 int init_myshebei(void)

将设备注册写在此处：

```
result=register_chrdev(254," myshebei",&myshebei_fops);
```

STEP 2：

将 myshebei.c 复制到../drivers/char 目录下,并且在…./drivers/char 目录下 mem.c 中，int chr_dev_init( )函数中增加如下代码：

```
#ifdef CONFIG_TESTDRIVE
init_myshebei ();
#endif
```

STEP 3：

在…../drivers/char 目录下 Makefile 中增加如下代码：

```
ifeq($(CONFIG_TESTDRIVE),y)
L_OBJS+=myshebei.o
Endif
```

STEP 4：

在……../arch/m68knommu 目录下 config.in 中字符设备段里增加如下代码：

```
bool 'support for testdrive' CONFIG_TESTDRIVE y
```

STEP 5：

运行 make menuconfig（**在** menuconfig **的字符设备选项里你可以看见我们刚刚添加的** 'support for testdrive'**选项，并且已经被选中）；**

STEP 6：文件系统的相应改变

在….. /dev/**目录下创建设备：**

```
mknod myshebei c 254 0
```

对文件系统的目录进行添加即可。

至此，你已经在 Linux **中增加了一个新的设备驱动程序**

5．嵌入式 Linux 内核的编译

对于 Linux **内核的编译，上面已经进行了说明，我们在此解释一下** make dep**：**

dependence **从字面上是依赖的意思，**make dep **的意思就是说：如果你使用程序** A**（比如支持特殊设备），而** A **需用到** B**(比如** B **是** A **的一 个模块/子程序)。而你在做** make config **的时候将一个设备的驱动 由内核支持改为** module,**或取消支持,这将可能影响到** B **的一个参数的设置，需重新编译** B**，重新编译或连接** A....**如果程序数量非常多， 你是很难手工完全做好此工作的。 所以，你要** make dep**。如果你** make menu **或** make config **或** make xconfig **后，直接** reboot**，会更快。只是你的内核根本没有任何改变。一般** linux **内核全部编译需要下述命令**

```
make xconfig（menuconfig）;
make dep;
make clean;
make bzImage;
make modules;
make modules_install
```

6．对于嵌入式 Linux 的内核开发，你还需要什么？

我想你应该对于嵌入式Linux **的内核开发有个了解,并且可以对** FFT-RM9200 **进行**Linux **内核的编译和裁剪了，同时，我们还介绍了你的新的驱动程序以及设备如何添加？如果你的驱动程序已经具备，你完全可以进行** Linux **的内核开发了。**

但是，你现在可能还在疑惑两个问题。一是，我怎么去编写一个新的设备驱动程序，这个问题就比较复杂了，对于不同的设备，你的代码都千差万别，你千万别试图从这本说明书中找到答案，应该去了解关于 Linux **驱动程序开发的相关内容。另外一个问题就是文件系统的问题了，因为内核启动起来后，关键是你什么都看不到，这时，你需要继续阅读下一章的内容。**

第 8 章　嵌入式 Linux 的文件系统开发

内容简介：

本章介绍嵌入式 Linux 文件系统的开发步骤，如何配置各种文件系统的内容等等

1．傅立叶给你提供的文件系统开发资源

当然，作为产品开发商，傅立叶提供你可以对文件系统开发的所有资源，包括

- FFT-RM9200 的 ext2 文件系统：myramdisk2.gz
- Microwin 的原代码（你也可以从网上下载）
- Busybox
- Samba
- Apache
- Mingetty
- 根文件系统和 jffs2 文件系统的创建
- 必不可少的关键性说明

这些内容的提供，使得你可以轻松面对文件系统的操作。

2．开发 Linux 文件系统的一般开发步骤

对于 Linux 文件系统的开发，有两种方法，一种就是自己从头开始建立根文件系统，另外一种是下载或者获取一个已经生成的文件系统，然后在此基础上添加和修改，最后形成自己的文件系统

2．1 以一个建好的文件系统为基础来创建

对于 FFT-RM9200 的开发板上的文件系统，我们建议在已经具备的文件系统上进行开发，这样你将会省去很多的开发时间，对于在傅立叶提供的文件系统上开发，有下面的步骤（ramdisk 统指文件系统影象，傅立叶提供 fftramdisk1-rmk7 和 fftramdisk1-http 两种）。

- 解开压缩

  bash$ gunzip ramdisk.gz
- 影象文件挂装

  bash$ mount -o loop ramdisk /mnt/your_ramdisk_directory
- 对/mnt/your_ramdisk_directory 目录进行操作，随意增减文件

  bash$ cd /mnt/your_ramdisk_directory

  bash$ do_whatever_you_want (create directories, files ...)
- 到你的影象文件目录下

  bash$ cd where_your_ramdisk_file_is
- 卸装文件系统

  bash$ umount /mnt/your_ramdisk_directory
- 压缩文件系统，生成最终的文件系统影象

  bash$ gzip -c -v9 ramdisk > /tftpboot/ramdisk

2．2 自己建立根文件系统

当然，你也可以自己去制作根文件系统，一般创造根文件系统可以有下面步骤：

- 创建一定大小的根文件系统

  ```
  mke2fs -vm0 /dev/ram 4096
  ```

- 根文件系统挂装

  ```
  mount -t ext2 /dev/ram /mnt
  ```

- 文件系统的操作

  ```
  cd /mnt
  cp /bin, /sbin, /etc, /dev ... files in mnt
  cd ../
  ```

- 去除挂装

  ```
  umount /mnt
  ```

- 文件系统生成

  ```
  dd if=/dev/ram bs=1k count=4096 of=ext2ramdisk
  ```

- 得到文件系统影象

  gzip –v9 ext2ramdisk

mke2fs 是用于在任何设备上创建 ext2 文件系统的实用程序 — 它创建超级块、索引节点以及索引节点表等等。

在上面的用法中，/dev/ram 是上面构建有 4096 个块的 ext2 文件系统的设备。然后，将这个设备（/dev/ram）挂装在名为 /mnt 的临时目录上并且复制所有必需的文件。一旦复制完这些文件，就卸装这个文件系统并且设备 （/dev/ram）的内容被转储到一个文件（ext2ramdisk）中，它就是所需的 Ramdisk（Ext2 文件系统）。

上面的顺序创建了一个 4 MB 的 Ramdisk，并用必需的文件实用程序来填充它。

一些要包含在 Ramdisk 中的重要目录是：

/bin — 保存大多数象 init、busybox、shell、文件管理实用程序等二进制文件。

/dev — 包含用在设备中的所有设备节点

/etc — 包含系统的所有配置文件

/lib — 包含所有必需的库，如 libc、libdl 等

2．3 改变 ramdisks 的大小

要想使用 ramdisk 你必须或是得到内核的支持或是以模块的形式将他加载到系统中。其中内核的配置选项是 CONFIG_BLK_DEV_RAM . 把 ramdisk 编译成一个可加载的模块的好处是你可以在加载是重新确定 ramdisk 的大小。

第一个办法。在 lilo.conf 文件中加入这一行：

ramdisk_size=15000 (or ramdisk=15000 for old kernels)

这样在你使用 lilo 命令和重新启动计算机之后，ramdisk 的默认大小将会是 15M。这是一个/etc/lilo.conf 文件的例子：

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
image=/boot/vmlinuz
    label=linux
```

root=/dev/hda2

read-only

ramdisk_size=15000

确切的说，我只是使用了 14M 多的空间，文件系统也将占用一定空间。

当你以模块的形式编译 ramdisk 时，你可以在加载的时候决定 ramdisk 的大小。这也可以通过修改/etc/conf.modules 的选项设置来做到。

options rd rd_size=15000

或是在命令行中指定参数给 ismod：

insmod rd rd_size=15000

以下是介绍如何使用这样的模块的例子：

卸载 ramdisk，`umount /tmp/ramdisk0` .

卸载模块(再上一节所提到的过程中自动加载), `rmmod rd`

加载 ramdisk 模块并且把它的大小设为 20M，`insmod rd rd_size=20000`

创建一个文件系统, `mke2fs /dev/ram0`

加载 ramdisk, `mount /dev/ram0 /tmp/ramdisk0`

3．如何开发 FFT-RM9200 的文件系统

傅立叶提供给你 FFT-RM9200 的评估板时，也为你提供了已经开发好的一个文件系统（展开时大小大约为 15M 多），同时在光盘资料汇总为你提供了 busybox，samba，mingetty，Apache、Microwin 等工具，供你在建立自己的文件系统时使用。

3．1 创建大容量的文件系统

上节我们谈了创建文件系统的一般步骤，但是上面所创造的文件系统只有 4M，傅立叶为你提供了创建大容量文件系统以及在成熟文件系统上建立自己文件系统的方法。

需要准备的工作

比如 FFT 为你已经提供的文件系统 fftramdisk1-rmk7，我们把它挂装到/mnt/test1 上，而我们自己创建的文件系统，我们挂装到/mnt/test2 上，文件设备名假使为/tmp/loop_tmp

操作步骤

STEP 1：把 fftramdisk1-rmk7 进行挂装

Mount –o loop …/fftramdisk1-rmk7 /mnt/test1

STEP 2：建立 loop 设备的临时挂接点和大小为 size 的文件，并清零

Mkdir /mnt/loop 2>/dev/null

Dd if=/dev/zero of=/tmp/loop_tmp bs=1k count=size >/dev/null

STEP 3：将 loop 设备和临时文件联系

Losetup /dev/loop0 /tmp/loop_tmp

STEP 4：建立大容量文件系统并配置自身

Mke2fs –m 0 /dev/loop0 2>/dev/null

STEP 5：虚拟盘挂接

Mount /dev/loop0 /mnt/test2 –t ext2

STEP 6：拷贝文件，改变初始化等文件，卸载挂接点

从/mnt/test1 和其他文件中选择拷贝到/mnt/test2 中

拷贝自己的应用程序到相应的目录下

对 inittab 等脚本文件进行修改

umount /mnt/test2

umount /mnt/test1

STEP 7：得到最后的文件系统

Dd if=/dev/loop0 bs=1k count=size of =…/ext2ramdisk1

Gzip –v9 ../ext2ramdisk1

3．2 交叉编译 busybox

从 http://www.busybox.net/downloads 下载 busybox 的最新版本

bash$ cd busybox-xx

bash$ make CROSS=arm-linux- LIBCDIR=/usr/local/arm/2.95.3 LIBRARIES = /usr/local/arm/2.95.3/arm-linux/lib/libc.a –lgcc GCCINCDIR =/usr/local/arm/2.95.3/arm-linux/include CROSS_CFLAGS+=-I$(LIBCDIR)/lib/gcc-lib/arm-linux/2.95.3/include -I$(GCCINCDIR) PREFIX=/home/user/usr/local/busybox **编译** busybox

bash$ make install **安装** busybox

所有文件被安装到/home/user/usr/local/busybox，用下列命令将其拷贝到 RamDisk 中

bash$ cp -a /home/user/usr/local/busybox/bin/* /mnt/ramdisk-as/bin/*

bash$ cp -a /home/user/usr/local/busybox/sbin/* /mnt/ramdisk-as/sbin/*

bash$ cp -a /home/user/usr/local/busybox/usr/sbin/* /mnt/ramdisk-as/usr/sbin/*

bash$ cp -a /home/user/usr/local/busybox/usr/bin/* /mnt/ramdisk-as/usr/bin/*

3．3 交叉编译 samba

从 http://www.samba.org 下载 samba 的最新版本

bash$ cd samba-xxx

bash$ CC="arm-linux-gcc" INCLUDES="/usr/local/arm/2.95.3/arm-linux/include/ /usr/local/arm/2.95.3/lib/gcc-lib/arm-linux/2.95.3/include" LIBS="/usr/local/arm/2.95.3/arm-linux/lib/libg.a"
CFLAGS="$INCLUDES $LIBS" ./configure --prefix=/home/user/usr/local/samba --sbindir=/usr/home/amine/usr/local/samba/sbin/ **配置编译选项**

bash$ make **编译** samba

bash$ make install **安装** samba

所有文件被安装到/home/user/usr/local/samba，用下列命令将其拷贝到 RamDisk 中

bash$ cp -a /home/user/usr/local/samba/sbin/* /mnt/ramdisk-as/usr/local/samba/sbin/*

bash$ cp -a /home/user/usr/local/samba/lib/* /mnt/ramdisk-as/usr/local/samba/lib/*

bash$ touch /mnt/ramdisk-as/usr/local/samba/lib/smb.conf

3．4 安装 RPM 包

bash$ mount -o loop your_ramdisk /mnt/your_ramdisk_directory **映射** RamDisk

bash$ cd /mnt/your_ramdisk_directory

bash$ rpm2cpio **目录**/package.rpm | cpio -i -d --no-absolute-filename **安装** RPM **包到** RamDisk

3．5 设置网络接口

在/etc/rc.d/init.d 目录中添加一个文件，文件内容为

ifconfig eth0 ip 地址

在/etc/rc.d/rc3.d 中创建一个链接

bash$ cd /etc/rc.d/rc3.d

bash$ ln -s ../init.d/interfaces S20interfaces

这样每次 Linux 启动式就会自动设置好网络接口了

3．6 安装 mingetty

mingetty 是一个小型的虚拟控制台程序，

可以从附属光盘或者

http://handhelds.org/download/linux/arm/netwinder-rpms/RPMS/base/3.1-15/获得.

先将 mingetty 安装到 RamDisk 中

bash$ mount -o loop ramdisk /mnt/ramdisk

bash$ arm-linux-strip -s -g whereever_mingetty_is_installed/sbin/mingetty

bash$ cp whereever_mingetty_is_installed/sbin/mingetty /mnt/ramdisk/sbin

在/etc/inittab 中加入一行

1:235:respawn:/sbin/mingetty tty1

3．7 创建 jffs2 文件系统

启动 FFT-RM9200 登陆后创建 jffs2 文件系统

[root@AT91RM9200DK /]$cd /mnt

[root@AT91RM9200DK /mnt]$mkdir AT45DB642　　创建目录用来映射 jffs2 文件系统

[root@AT91RM9200DK /mnt]$mount -t jffs2 /dev/mtdblock/0 AT45DB642　　映射文件系统

这样就可以通过/mnt/AT45DB642 来访问 AT45DB642 存储芯片了

3．7 系统运行

系统下载完毕，linux 启动后，出现

AT91RM9200DK login:

键入 root，系统登录，进入 linux 系统

4．图形界面和上网

4．1 图形界面

系统已经移植了 microwin，系统启动后，只需要到/root/bin 目录下面，进行运行即可

4．2 上网和浏览器

我们只介绍目前移植的部分，更多的当然需要客户进行 linux 下的网络编程了，上网和浏览器步骤如下：

- 系统加电启动，分别从目标板和主机进行登陆
- 进行网络设置（假设 DNS 服务器已经设置好，否则在/etc/resolv.conf 中设置）

```
ifconfig  eth0  192.168.0.11                ；Ip 地址
route  add  default  gw  192.168.0.1        ；网关
route                                         ；查看网络设置
ping  192.168.0.1                           ；是否 ping 通
```

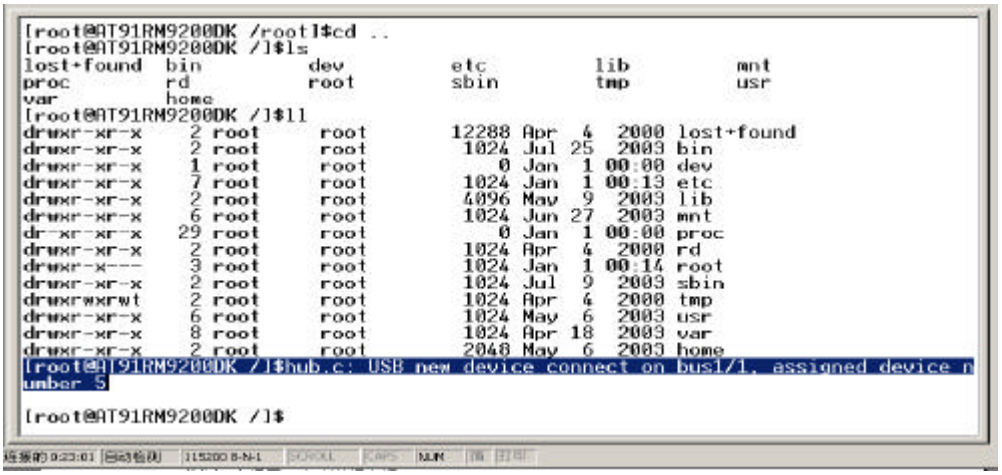
- 在目标板上，进入/root/bin 目录，运行 nano-X，出现图形界面
- 在主机的超级终端，进入/usr/local/viewml/bin 目录
- 运行./viewml，在目标板的 LCD 或者 VGA 上出现浏览器
- 通过目标板的键盘和鼠标即可上网(如果 ping 正常而不能上网,请查看和修改 DNS 服务器/etc/resolv.conf 文件)

5．U 盘的访问

FFT-9200 系统可以通过 USB--HUB 扩展出多个 USB 接口，支持 U 盘、USB 鼠标、USB 键盘、USB 硬盘、USB 摄像头等等，对于 USB 鼠标和键盘，运行时自然支持，我们在此讨论 USB 其他设备的使用。

5．1 U 盘和 USB 硬盘

当插入 U 盘时，出现下面显示



U 盘的设备位于/dev/scsi/host0/bus0/target0/lun0 下，类似于 PC 机上的 USB 设备，我们需要 mount –t vfat /dev/scsi/host0/bus0/target0/lun0/part2 /dev/usb，打开 cd /dev/usb;ls 我们可以看到 U 盘的内容，并且可以进行相应的操作。

6．对于嵌入式 Linux 的文件系统开发，你还需要什么？

至此，我们已经开发完毕文件系统，你可以进行内核和文件系统的下载，系统已经能够正常启动了，可能你这时已经信心十足，完全可以进行你的产品开发而专注于自己的顶层应用程序，这时，傅立叶恭喜你，我们的任务已经基本完成，但是傅立叶还会给你更多的技术支持，帮助你进入真正的开发领域。

这时候，你可能会着重考虑 linux 下的网络编程或者你自己的产品软件设计了，如果你对顶层程序的编写还有什么疑惑的话，请你继续阅读下面的章节，当然还有优美的图形界面，我们还专门用一章来进行介绍。

第 9 章 如何开发和运行你的顶层程序

[NOTES]

本章给客户举个例子，介绍应用程序的编写方法，客户只需要掌握它的方法，对于直接使用这个例子的情况，鉴于很多客户环境创建的问题，傅立叶不保证例子的正确性，仅仅讲述方法。

1.“Hello World”程序的编译与调试

其实，用户的顶层程序属于文件系统的一部分，我们在文件系统中已经介绍了，你只需要在 Linux 主机下，编译好应用程序，加入到文件系统的对应目录下即可，最多你可以改变脚本文件，使得其能够自动执行或者其他

标准 Linux 下的本地应用程序的编译及安装一般分三步,运行'./configure'自动配置，'make'开始编译，'make install'完成安装。这只是一种习惯用法，如果某个程序员对此置之不理的话,编译他写的程序就得看看 INSTALL 文件了,如果他又恰巧忘记写 INSTALL 文件，那你可能需要多花一些时间。

以上只是标准 Linux 下的本地应用程序的编译及安装方法,很多应用程序不支持交叉编译，这样就需要自己对 Makefile 文件做一些修改，主要是改动其编译器的名称，比如把 CC=gcc 改成 CC=arm-linux-gcc，另外一些编译选项也要做相应的修改。

如果是自己写的程序，那就动手写个 Makefile，以下是在为 FFT-RM9200 的 linux 上运行的 hello 程序的示例。

```
/* * hello.c */
#include <stdio.h>
int main(void)
{
  printf("hello, world!\n");
  return 0;
}
/* * Makefile for hello */
hello: hello.c
  arm-linux-gcc -o hello.o hello.
```

如果你已经熟悉了以上个部分，就可以编写自己的嵌入式 Linux 程序了。

2．如何使得你的应用程序在目标板上运行

上节已经讲过了，任何程序都属于文件系统的一部分，加入到文件系统中，就可以运行

3．对于顶层程序的开发，你还需要什么？

当然，一个产品的开发，关键就是顶层程序，这就是你自己的事了，如果你开发路由器等产品，你自己只需要写好初始化文件和脚本文件，如果你开发 PDA 等类似产品，需要较好的人机界面的话，还需要考虑在 Microwin 下进行编程，当然你还可以选择 mimi GUI 等环境，也许你可能少不了中文的支持，否则也许你的市场会大受影响。下一章将会介绍 microwin 下的编程和中文内核，请你继续阅读。。。。

第 10 章　图形界面的开发

[NOTES]

本章讲述图形界面开发的方法，对于其中列举的例子，由于考虑到很多客户实际中的各种问题，比如编译器的建立路径不匹配等等，傅立叶仅仅说明思路，不保证例子本身在客户机器上面的正确性。

1．FFT-RM9200 的图形界面介绍

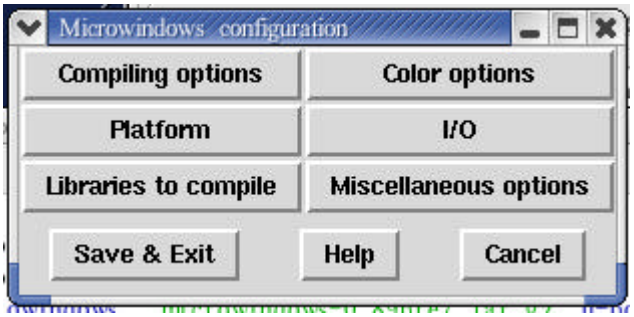
FFT-RM9200 采用的图形界面是 microwin0.89，所有的编译完成的程序放在/root/bin 下面，相应的光盘中提供了 microwin 的原代码，用户也可以从相应的网站上下载

1．1 安装 microwin

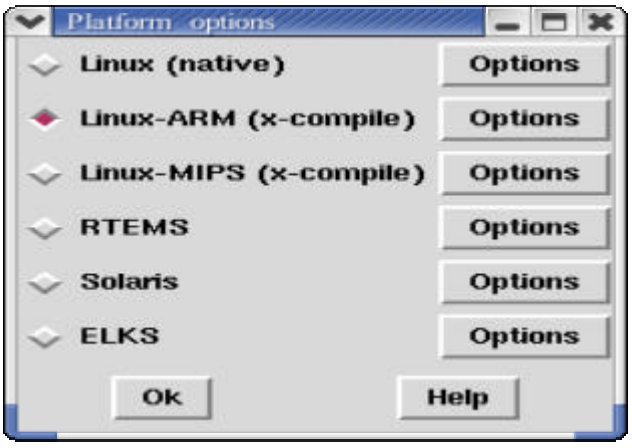
把 microwin 的原代码拷贝到你设定的目录下，然后解开

1．2 面向 FFT-RM9200 的 microwin 配置

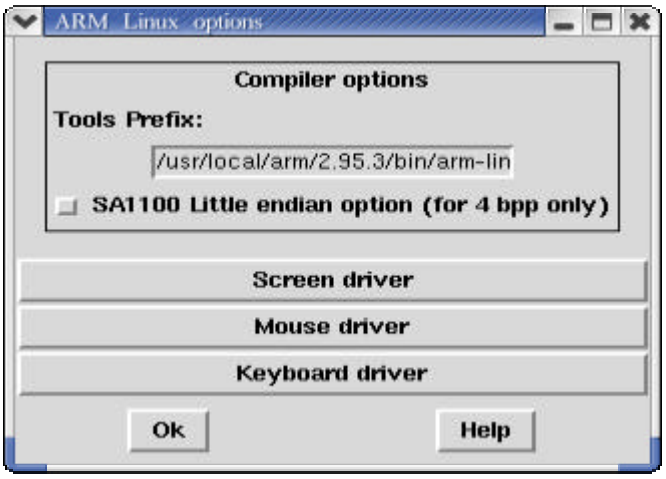
配置命令为 make xconfig



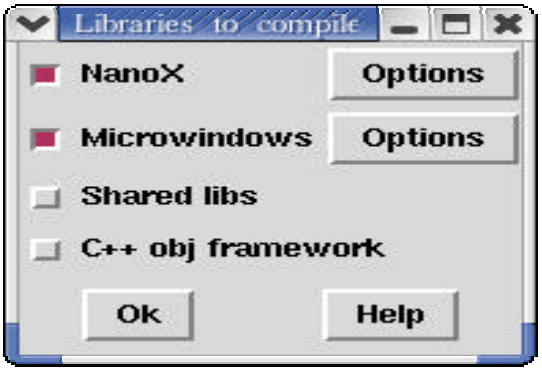
你可以打开各种选项进行设置，下面是平台选项



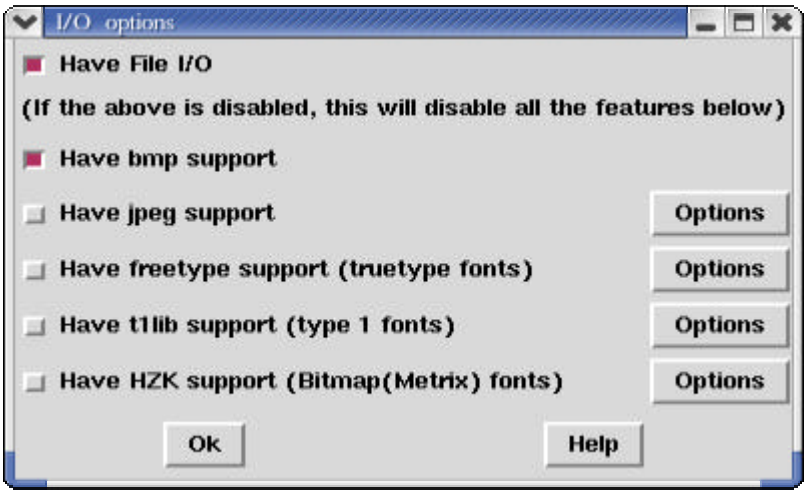
linux-arm 的 options



库设置



I/O **设置**



你可以进行你响应的选项

1．3 microwin 的编译

配置结束后，运行编译命令

make clean

make

生成的文件在…/microwin/src/bin **和**…/microwin/src/lib **下**

1．4 microwin 的最后移植

把…/microwin/src/bin 和…/microwin/src/lib 下的文件拷贝到相应的目录下即可运行

2．如何开发图形界面的顶层程序？

2．1 图形界面下的“ Hello World ”

```
#include <stdio.h>
#define MWINCLUDECOLORS
#include "microwin/nano-X.h"

GR_WINDOW_ID    wid;
GR_GC_ID        gc;

void event_handler (GR_EVENT *event);

int main (void)
{
    if (GrOpen() < 0)
     {
        fprintf (stderr, "GrOpen failed");
         exit (1);
     }

    gc = GrNewGC();
    GrSetGCUseBackground (gc, GR_FALSE);
    GrSetGCForeground (gc, RED);

    wid = GrNewWindowEx (GR_WM_PROPS_APPFRAME |
                         GR_WM_PROPS_CAPTION |
                         GR_WM_PROPS_CLOSEBOX,
                        "Hello Window",
                        GR_ROOT_WINDOW_ID,
                       50, 50, 200, 100, WHITE);

    GrSelectEvents (wid, GR_EVENT_MASK_EXPOSURE |
                         GR_EVENT_MASK_CLOSE_REQ);

     GrMapWindow (wid);
    GrMainLoop (event_handler);
}

void event_handler (GR_EVENT *event)
{
```

```
        switch  (event->type)
        {
        case  GR_EVENT_TYPE_EXPOSURE:
            GrText  (wid,  gc,  50,  50,
                        "Hello  World",    -1, GR_TFASCII);
             break;

        case GR_EVENT_TYPE_CLOSE_REQ:
             GrClose();
            exit  (0);
        }
    }
```

对于上述程序的编译，当然需要采用 ARM-Linux **的编译器，然后放到相应的目录中，系统运行后便可以执行，如果为了快速开发，最好在** microwin **的下面，统一编译生成最后的可执行文件。**

3．你还需要什么？

本说明书仅仅是简单的说明，对于 microwin **的内容很多，需要用户自己去完成，一般可能很多用户会关心下面两个问题**

- **如何能够出现象桌面** PC **的图形界面？**
- **对于图形化编程，有那些函数？如何调用？**

第一个问题需要编写初始化程序和脚本程序，这部分由用户自己根据需要完成，网上资料也很多，后面的问题，我们列举一些，以进行参考，请参考相应的 microwin **目录下的文献** Nano-X –docs.pdf

傅立叶还为你提供了更多的 microwin **下的文献，请你阅读光盘资料。**

附　录

附录 1　FAQ

1．AT91RM9200 Boot Rom problem using 8-bit parallel memories on NCS0

Question: Problem when trying to use the internal Boot Rom to access to an 8-bit parallel memories

Answer: There is an error in the internal Boot Rom program, the wait state number on CS0 is set to 0 during the Boot Rom initialisation. This gives an access time of 20ns at 48MHz Master Clock frequency. So the AT91RM9200 will not be able to access memories with standard access time such as 70 or 90ns. To conclude, the AT91RM9200 cannot boot from an 8-bit parallel memories. However, the AT91RM9200 internal Boot Rom can be used to boot from a SPI DataFlash or a Two-wire EEPROM.

Applies To: AT91RM9200

Post Date: 09/15/03

2．AT91RM9200DK Internal Boot ROM Program

Question: How to boot from the internal ROM Boot Program on the AT91RM9200DK Development Kit?

Answer: The Boot Mode Select (BMS) pin state during reset allows you to select the boot memory. By default, the BMS pin state on the AT91RM9200DK is set to low level and the AT91RM9200DK boot out from the external Flash memory which embeds the U-Boot boot program.

If the user want to boot out from the internal ROM Boot Program, the BMS pin state can be set to high level by removing the resistor named R159 from the PCB. This resistor can be found on the back side of the PCB just behind the 20-pin JTAG connector.

Applies To: AT91RM9200DK

Post Date: 11/26/03

3．Core and IO voltage power-up sequence and constraint

Question: Core and IO voltage power-up sequence and constraint?

Answer: There is no specific sequence to power up an AT91 microcontroller. Either the core voltage or I/O voltage can be powered first without any risk of destruction.

However, Output values and power consumption are not guaranteed during power up sequence. We advise to perform a product specific reset after stabilization of supplies to have a known state. Although only one voltage rail can be supplied without any risk of destruction, correct

functionality of the device cannot be guaranteed.
Applies To: All AT91 products
Post Date: 10/30/03

4．Different definitions of the Boot Mode Select pin from the AT91RM9200's Full Datasheet (ref. 1768A-ATARM-22-APR-03).

Question: There is two different definition of the Boot Mode Select pin (BMS) from the AT91RM9200's Full Datasheet (ref. 1768A-ATARM-22-APR-03) from page 87 and page 127. Which one is correct?

Answer: Actually, there is two definitions of the BMS pin from two different pages. In Page 87, section line, within parenthesis, you must read "BMS high during the reset". This will be corrected in the next version of the datasheet.
Applies To: AT91RM9200's Full Datasheet (ref. 1768A-ATARM-22-APR-03)
Post Date: 06/12/03

5．Wrong bootcmd command on AT91 CD-ROM May 2003

Question: When I try to load ramdisk and zImage files by tftp, it fails why?

Answer: The bootcmd is used when using AT91RM9200-DK running under Linux. You need first to set the destination_address and after the file_name. On the cd-rom May 2003 the order is inverse. The correct bootcmd command is: > setenv bootcmd tftp 0x20008000 zImage \; tftp 0x21000000 ramdisk \; go 0x20008000 Do not forget to save your parameters. > saveenv. Reset the AT91RM9200-DK.
Applies To: AT91RM9200-DK
Post Date: 11/17/03

6．Flash/ROM Startup Sequence Debugging

Question: I want to debug my application stored in Flash/ROM memory from the reset to the REMAP command. How can I do?

Answer: By using the Watchdog Timer. Actually, the Watchdog can be programmed to generate an internal Reset. This fully resets the AT91 device without resetting the external hardware. This can be done by programming the watchdog by entering the following commands through the Command Line Interface (CLI) or by loading a script file from your Debugger:
Script file example for ARM ADS 1.2 with the AT91x40 Series:

```
Com Enable the vector catch for reset only
spp vector_catch 0x1
Com Set a breakpoint at address 0
br 0x0
Com This script follows the WD Enabling Sequence from the x40 Datasheet
```

```
Com Disable the Watchdog by clearing the bit WDEN:
Com This step is unnecessary if the WD is already disabled (reset state).
setmem 0xFFFF8000 0x2340 32
Com Initialize the WD Clock Mode Register:
Com (HPCV = 15 and WDCLKS = MCK/1024)
setmem 0xFFFF8004 0x0000373F 32
Com Restart the timer
setmem 0xFFFF8008 0x0000C071 32
Com Enable the watchdog: (Internal Reset enabled)
Com setmem 0xFFFF8000 0x00002341 32
Com Enable Internal Reset generation
setmem 0xFFFF8000 0x00002343 32
go
```

After the timeout period of the Watchdog, the debugger will stop at address 0 due to the "br 0x0" command.

If the reset has been performed correctly, the EBI_CSR0 must contain the reset value as described in the datasheet. From now, you can run step by step until the remap command. Note that the remap command cannot be stepping-in due to the pipeline. You can set a breakpoint after the remap command.

Applies To: All AT91 Products.

Post Date: 12/02/02

7．HOLD and HOLDA pin names showed on the AT91RM9200-DK Electrical Schematics

Question: The IO pins PC14 and PC15 are called HOLD/PC14 and HOLDA/PC15 on the AT91RM9200-DK Electrical schematics but I looked up any information on the datasheet and there is nothing about the HOLD and HOLDA features?

Answer: On AT91RM9200-DK User Guide, you can find the AT91RM9200-DK Electrical Schematics. The pins M17 and L13 are called, respectively, HOLD/PC14 and HOLDA/PC15 but these pins are only IO and the HOLD and HOLDA functions do not exist. The pin description on the datasheet is correct. These pin must be called PC14 and PC15 only.

Applies To:AT91RM9200

Post Date: 09/02/03

8．How to add SDRAM memory on an AT91RM9200 Development Kit?

Question: How to add SDRAM memory on an AT91RM9200 Development Kit?

Answer: At the factory, the AT91RM9200-DK is fitted with 2 SDRAM devices organized as 2M x 16-bit x 4 banks (device reference: Micron MT48LC8M16A2TG-8E), for a total amount of 32Mbytes of memory. In order to upgrade this amount of memory, a compatible footprint has been designed on the board to fit larger memory devices. To do so, the user has to replace the SDRAM fitted by default with 2 devices organized as 4M x 16-bit x 4 banks (device reference: Micron MT48LC16M16A2TG-7E), for a total amount of 64Mbytes of memory.

Applies To:AT91RM9200 Development Kit
Post Date: 10/02/03

9．U-Boot Binary Files

Question: I have erased the flash content on the AT91RM9200DK where can I find the U-Boot binary files?

Answer: U-Boot is made up of three parts, a primary bootstrap, a de-compression executable and a gzip-compressed binary executable. The primary bootstrap is concatenated to the de-compression executable into one single file called the boot image and named boot.bin. The gzip-compressed binary executable is called the U-Boot image and named u-boot.gz. These two files are available in the AT91CDROM under the following directory path: CDROM\Pages\Software\Software_ARM9\Tools\U-Boot
Applies To: AT91RM9200DK
Post Date: 10/30/03

10．USART RS485 Mode, Faulty Description of RTS Pin Behavior in AT91RM9200 Datasheet, 1768B

Question: USART RS485 Mode, Faulty Description of RTS Pin Behavior in AT91RM9200 Datasheet, 1768B

Answer: USART RS485 Mode, Faulty Description of RTS Pin Behavior in AT91RM9200 Datasheet, Atmel Literature Number 1768B On page 419 of the datasheet, In the RS485 Mode section: 1. "The RTS pin is driven low when the trasnmitter is operating." Should read: The RTS pin is driven high when the transmitter is operating. 2. "Significantly, the RTS pin remains low when timeguard is programmed..." Should read: "Significantly, the RTS pin remains high when timeguard is programmed..." 3. Figure 196 and Figure 197 give incorrect descriptions of RTS behavior.
Applies To:
Post Date: 08/29/03

11．Wrong bootcmd command on AT91 CD-ROM May 2003

Question: When I try to load ramdisk and zImage files by tftp, it fails why?

Answer: The bootcmd is used when using AT91RM9200-DK running under Linux. You need first to set the destination_address and after the file_name. On the cd-rom May 2003 the order is inverse. The correct bootcmd command is: > setenv bootcmd tftp 0x20008000 zImage \; tftp 0x21000000 ramdisk \; go 0x20008000 Do not forget to save your parameters. > saveenv. Reset the AT91RM9200-DK.
Applies To: AT91RM9200-DK
Post Date: 11/17/03

附录 2　JTAG 接口电路