目录

XSBase快速使用向导

一．XSBase简介

XSBase是深圳亿道电子最新推出的一款基于英特尔XScale PXA255的高端嵌入式解决方案，此方案提供丰富的软硬件资源和参考设计方案。XSBase是一款理想的PDA、手机等消费电子、信息家电、通讯和工业控制等应用的开发系统。成功开发的案例有语音系统、车载系统、工业控制、电力信息网关、嵌入式监控系统、通讯终端、控制终端、消费电子、多媒体、视频、音频等方面的应用。XSBase为客户评估芯片、完成自己软硬件设计提供了方便。因此此方案对于客户自己底层系统的设计有重要的意义，硬件原理图、设备驱动原代码、集成开发环境等可以大大地加速软硬件工程师的开发设计。同时，也极大地解决了项目研发中软硬件开发不同步的问题，让软件人员在项目初期就可以展开实际的工作。同时此方案也非常适合高校的计算机专业、电子信息工程、自动化、仪器仪表、机电一体化等专业创建嵌入式实验室，为师生提供嵌入式领域的最新的技术发展方向，为师生开展学术研究、课题研究提供良好的实验平台。

XSBase(LINUX版)是在XSBase开发平台基础上基于LINUX 2.4.18内核的一套完整的嵌入式开发方案。我们提供完整的开发工具、Bootloader、内核、文件系统、外设驱动等源代码。我们还提供及时有效的技术支持，为客户的项目评估、项目开发提供帮助。

选择XSBase255，选择成功！

二．XSBase和主机连接

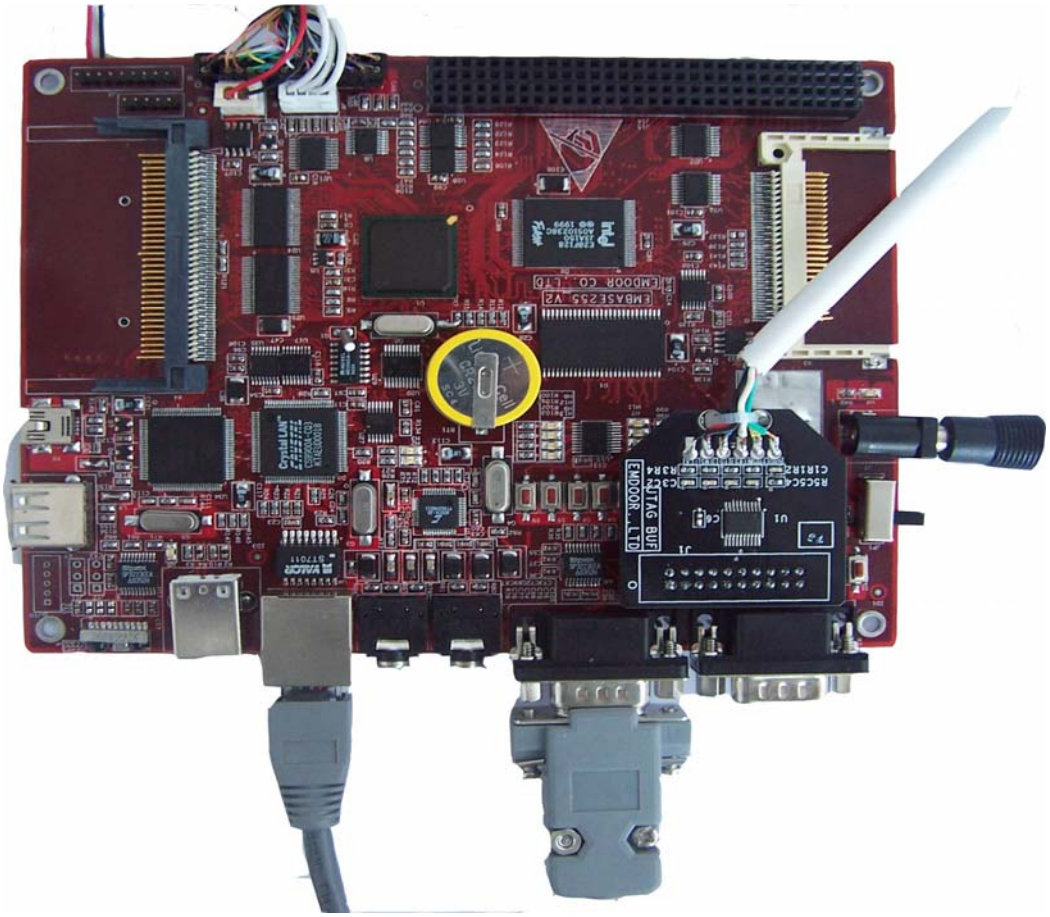
XSBase提供基于Linux 2.4.18内核的或基于WinCE 4.2嵌入式开发环境，为了实现在XSBase平台上开发嵌入式Linux的应用程序，创建或拥有一个完善的嵌入式Linux开发环境是非常必要的。

基于嵌入式Linux开发环境一般由目标系统硬件开发板XSBase和装有Linux桌面版的主机平台PC组成。我们这里用的是Redhat9.0的版本。目标系统硬件开发板XSBase用来运行

嵌入式操作系统Linux、用户系统应用程序等，而主机平台用来嵌入式操作系统内核编译，文件系统的制作和系统应用程序开发和调试等等。双方一般通过串口、并口和以太网等建立连接关系。

用户第一次打开实验箱时，请检查箱子里的配件是否齐全。

1. 第一次使用XSBase时请仔细阅读《XSBase使用手册Linux版本》。按照下图完成接线。



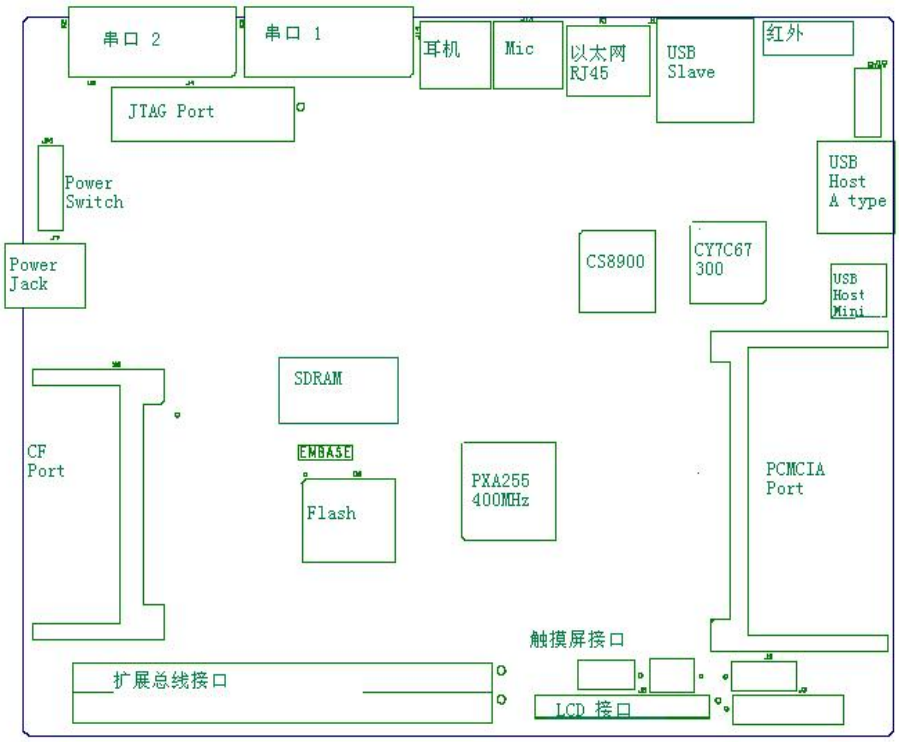
2. 首先使用我们配套的JTAG下载线连接到XSBase的20pin的JTAG接口上，另一端连接到主机平台的25pin的并口上。

3. 使用配套的串口连接线连接到XSBase的串口1上，另一端接到主机平台的串口上。

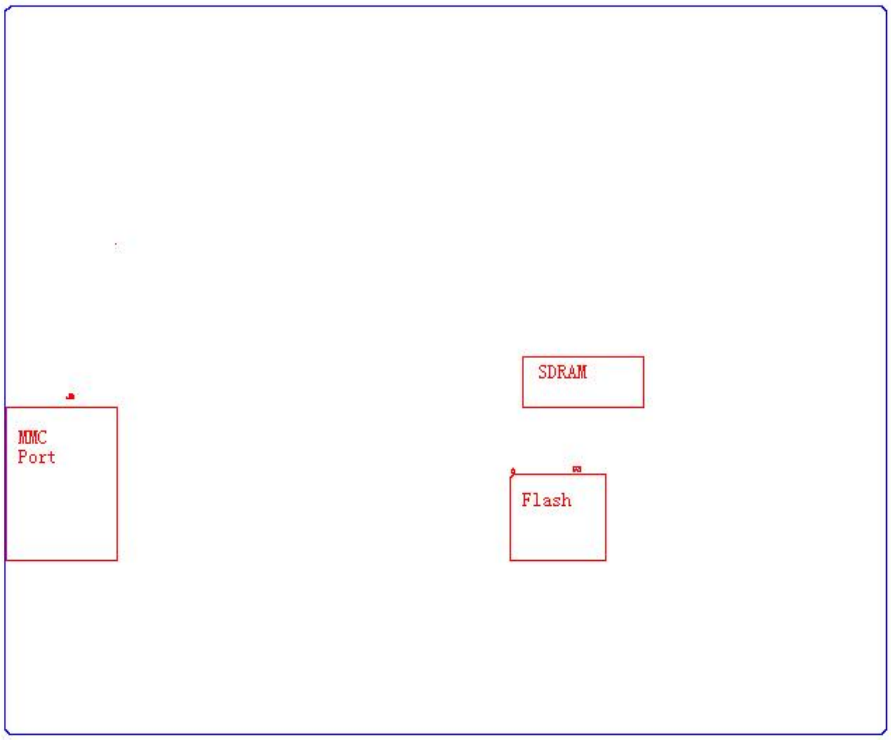
4. 使用配套的交叉网线将XSBase的网口和主机平台网口连接。

5. 使用配套的电源变压器将220V交流转换成5V直流电源。

6. 在打开电源开关前请先确保JTAG、串口、网口和电源接线的正确。



XSBase正视框图



XSBase背视框图

简介 1

本章将对XSBase系统做概括介绍。

1.1 XSBase的简介

XSBase采用高性能的Intel® PXA255处理器和运行Linux/Wince操作系统。通过多种多样的接口可以极大的扩展系统并且很容易的为 PXA255外设做全方位的测试。

XSBase板的特色

- 这个系统采用高性能（400MHz主频）和低功耗的Intel® PXA255处理器，所以它非常适用于手持移动产品。

- 这个系统安装了最新的2.4.18版本的Linux内核,它是以稳定性和可靠性闻名的操作系统。用户可以在一个非常稳定的环境下运行用户程序，从而实现高质量的产品。

- 用户应用程序可以在不需要外接存储设备如硬盘的环境下运行。同时MTD的FLASH文件系统的使用极大的提高了系统的容量和可靠性。

- PXA255提供众多的外设接口为开发提供了极大的扩展性和适应性。

- XSBase需要与一个带有网口和串行口的主机平台进行通讯。

- XSBase的LINUX已在Redhat 9.0（X86系列的LINUX）下进行过测试。也可以使用其他版本的LINUX。

1.2 系统组成

本节将介绍系统提供的硬件和软件资源及相关配件。

1.2.1 硬件说明书

Item	Description
处理器	Intel® XScale PXA255 400MHz
SDRAM	Samsung 64Mbyte
Flash	Intel® strata flash 32MByte
以太网	CS8900A 10BaseT
声卡	AC'97 Stereo audio
显示	LG TFT LCD 6.4"(640 * 480)
触摸屏	ADS7843 touch screen
USB	USB Slave，USB HOST
PCMCIA	1 Slot
实时时钟	Real time clock RTC4513
红外	HDSL3600
CF	1 Slot
MMC	1 Slot

表1-1 硬件资源表

1.2.2 软件说明书

Item	Description
操作系统	Linux 2.4.18 kernel
设备驱动	CS8900 Ethernet
	AC'97 streo audio
	Frame buffer
	ADS7843 touch screen
	USB
	PCMCIA(CF driver included)
	RTC4513(Real Time Clock)
	IrDA
	MMC
文件系统	JFFS2
图形用户界面	Tiny-X server

表1-2 软件资源表

1.2.3 配件

Item	Description
主板	PXA255 Board
光盘	相关资源和映像文件
电源	5V直流电源
电缆	Serial cable JTAG cable Ethernet cross cable USB cable
文档	XSBase中文使用手册
Case	Aluminum case

表1-3 产品配件表

1.2.4 光盘内容

```
XSBase CD
    |
    +---- Bootloader        PXA255 Bootloader for XSBase
    |
    +---- Datasheet         Datasheet for XSBase
    |
    +---- Documents         Docoments for XSBase
    |
    +---- Filesystem        Root filesystem for XSBase
    |
    +---- Image             Binary image(bootloader, kernel, rootfs) for
XSBase
    |
    +---- Jflash-XSBASE     Jflash program for PXA255
    |
    +---- Kernel            Linux Kernel source for XSBase
       |
       +---- Patch          Linux Kernel source patch for XSBase
    |
    +---- RPM               Tools used Host computer
    |
    +---- Source            PCMCIA Host and Client driver, Card Services
source
    |
    +---- Toolchain         PXA255 Toolchain for XSBase
    |
    +---- Tiny-X            Tiny-X server source
```

操作 2

本章将介绍系统的安装过程和最初的操作测试。

2.1 XSBase的外观图

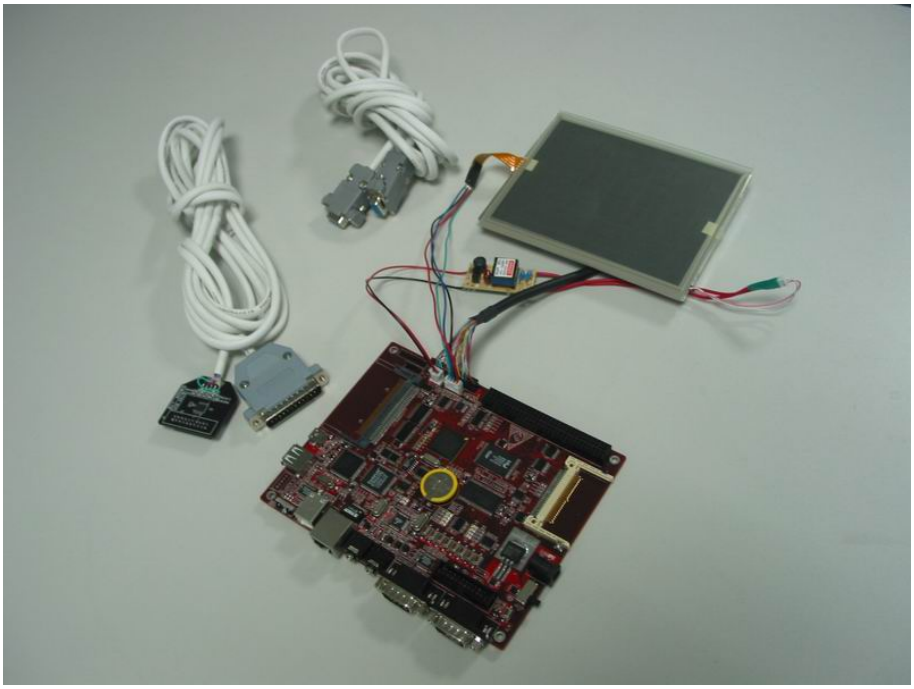


图2-1 正面外观图

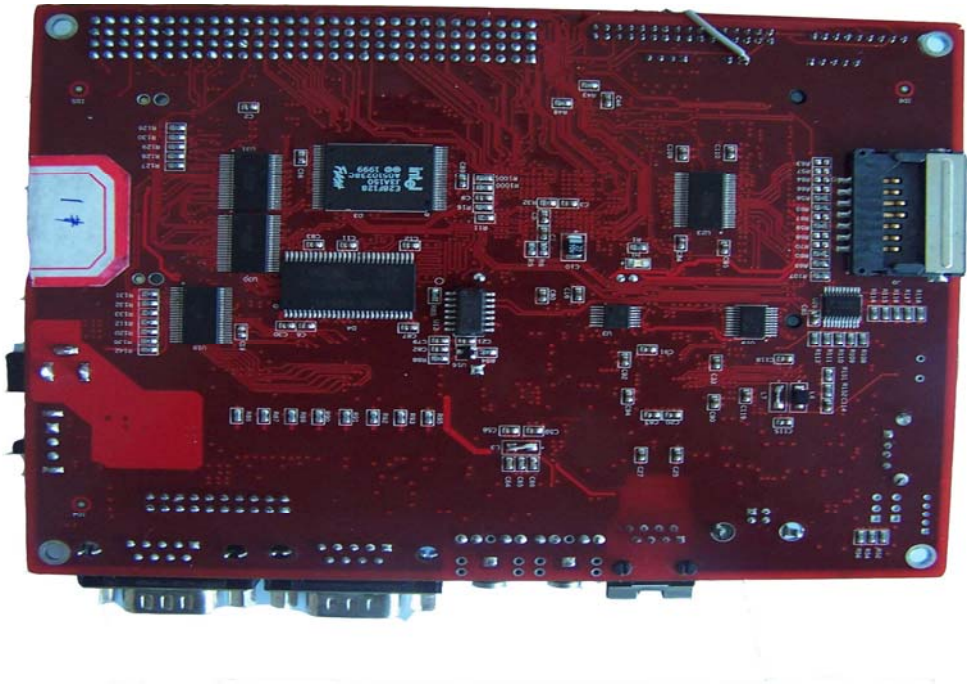


图2-2 背面外观图

2.2 连接到XSBase255板子

警告：移动板子时请手拿板子的两端以防止静电对板子的伤害。在仔细检查接线后再通电。

1、第一次打开箱子时请核对箱子的内容。

2、连接JTAG线。用JTAG线把板子的20-pin的接口和主机的并口连接起来。

3、连接串口线。通过板子上的串口1与主机平台的串口连接。

当连接串口1时请仔细的检查Linux的minicom或Windows的超级终端的配置

4、连接以太网线。用以太网线将主机平台的网卡接口和板子上的RJ45网口接上。

5、连接电源线－－当连接电源线时请确保使用我们提供的AC/DC的电源转换器，它是将220V的交流电压转换成5V的直流电压。

2.3 设置Linux的Minicom

Minicom是LINUX下的串口调试工具。

输入命令 "minicom -s"

```
[root$ super ]# minicom –s
```

“-s”选项将产生下面的配置菜单。

```
+------[configuration]----------+
| Filenames and paths           |
| File transfer protocols       |
| Serial port setup             |
| Modem and dialing             |
| Screen and keyboard           |
| Save setup as dfl             |
| Save setup as                 |
| Exit                          |
| Exit from Minicom             |
+-------------------------------+
```

选择"serial port setup"项显示下面的菜单。

```
+-----------------------------------------------+
| A - Serial Device      : /dev/ttyS0           |
| B - Lockfile Location  : /var/lock            |
| C - Callin Program     :                      |
| D - Callout Program    :                      |
| E - Bps/Par/Bits       : 115200 8N1           |
| F - Hardware Flow Control : No                |
| G - Software Flow Control : No                |
|                                               |
| Change which setting?                         |
+-----------------------------------------------+
```

配置如上面所述。

″ttyS0″表示使用串口 1，″ttyS1″表示使用串口 2。

保存设置，选择″Save setup as dfl.″

按 ESC 键结束配置菜单。

然后，minicom 窗口打开。

如果重启板子，你将会看到如图2－3看到的启动信息。

如果没有，请检查 bootloader loading，minicom 设置和连接线。

2.4 Windows超级终端的设置

超级终端是Windows的串口调试工具。

- 运行超级终端

  选择 开始>程序>附件>超级终端。
- 创建一个新的连接：文件>新建连接。输入一个连接名字按OK。
- 选择一个串口设备通常用串口1或串口2。
- 请输入如下的连接配置。

  Bits per second ：115200

  Data bits ：8

  Parity ：None

  Stop bits ：1

  Flow control ：None

  按OK。
- 重启 XSBase板子，将看到图2-3所示内核启动信息。

2.5 XSBase 启动

启动过程将以minicom为例描述，因为启动过程在minicom和超级终端中是一样的。

1、连接到主机并打开minicom。请参考第2.2节。

2、打开板子电源开关。

3、请检查minicom出现的启动信息。系统在发行的时候在flash存储器中预装了Bootloader、 kernel、文件系统。

```
XSBASE
 Copyright (C) 2002 EMDOOR,. ltd.
 Support: http://www.emdoor.com

Autoboot in progress, press any key to stop ...Autoboot started.

Starting kernel ...

Uncompressing Linux..... done, booting the kernel.
Linux version 2.4.18-rmk7-pxa1-XSBASE (root@bedguy.pe.kr) (gcc version 2.95.3 20010315 (release))  12 15:24:40 KST 2003
CPU: Intel XScale-PXA255 revision 6
Machine: SUPER XSBase
Warning: bad configuration page, trying to continue
MCS0 = 0x7ff87ff0
MCS1 = 0x5aa85aa8
MCS2 = 0x24482448
On node 0 totalpages: 16384
zone(0): 16384 pages.
zone(1): 0 pages.
zone(2): 0 pages.
Kernel command line: root=1f03 rw console=ttyS0,115200 init=/linuxrc
Console: colour dummy device 80x30
Calibrating delay loop... 397.31 BogoMIPS
Memory: 64MB = 64MB total
Memory: 62252KB available (1548K code, 301K data, 368K init)
```

```
Dentry-cache hash table entries: 8192 (order: 4, 65536 bytes)
Inode-cache hash table entries: 4096 (order: 3, 32768 bytes)
Mount-cache hash table entries: 1024 (order: 1, 8192 bytes)
Buffer-cache hash table entries: 4096 (order: 2, 16384 bytes)
Page-cache hash table entries: 16384 (order: 4, 65536 bytes)
POSIX conformance testing by UNIFIX
Linux NET4.0 for Linux 2.4
Based upon Swansea University Computer Society NET3.039
Initializing RT netlink socket
PXA USB Controller Core Initialized
get_random_bytes called before random driver initialization
USB Function Ethernet Driver Interface
Starting kswapd
JFFS2 version 2.1. (C) 2001 Red Hat, Inc., designed by Axis Communications AB.
Console: switching to colour frame buffer device 80x30
pty: 256 Unix98 ptys configured
Serial driver version 5.05c (2001-07-08) with no serial options enabled
ttyS00 at 0x0000 (irq = 14) is a PXA UART
ttyS01 at 0x0000 (irq = 13) is a PXA UART
ttyS02 at 0x0000 (irq = 12) is a PXA UART
ads7843 touch screen driver initialized
block: 128 slots per queue, batch=32
Cirrus Logic CS8900A driver for Linux (V0.01)
eth0: CS8900A rev D at 0xf0000300 irq=0, no eeprom , addr: 00:12:34:56:78:9A
ac97_codec: AC97 Audio codec, id: 0x4352:0x5914 (Cirrus Logic CS4297A rev B)
Probing XSBASE Flash at physical address 0x00000000
Using buffer write method
No RedBoot partition table detected in XSBase Flash
Using static partition definition
Creating 4 MTD partitions on "XSBase Flash":
0x00000000-0x00040000 : "Bootloader"
0x00040000-0x000c0000 : "Partition Tables"
0x000c0000-0x001c0000 : "Kernel"
0x001c0000-0x02000000 : "Filesystem"
Linux Kernel Card Services 3.1.22
  options:  none
```

```
Intel PXA250/210 PCMCIA (CS release 3.1.22)
XSBASE PCMCIA INIT
MMC subsystem, $Revision: 0.3.1.14 $
MMC block device driver, $Revision: 0.3.1.16 $
PXA250 MMC controller driver, $Revision: 0.3.1.12 $
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP
IP: routing cache hash table of 512 buckets, 4Kbytes
TCP: Hash tables configured (established 4096 bind 4096)
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
NetWinder Floating Point Emulator V0.95 (c) 1998-1999 Rebel.com
VFS: Mounted root (jffs2 filesystem).
Freeing init memory: 368K
Setting up RAMFS, please wait ...
done and exiting

INIT: version 2.78 booting

INIT: Entering runlevel: 3
Starting syslog Starting system logger:
Starting pcmcia Starting PCMCIA
Starting etwork Starting network
Starting X...

XSBASE login: root
[root@XSBASE /root]$
```

图2-3 XSBase Booting message

4、当看到XSBase登陆注册时输入“root”不用输入密码，回车即可。

当启动完成时，X-SERVER将运行在LCD(TFT)上。

如果在minicom没有出现启动信息或者X-SERVER没有在LCD运行，请重新下载Bootloader, Kernel, and Filesystem。参考第3.2节。

如果重新下载还没有看到任何启动信息，请联系我们的技术支持。

XSBase Building 3

本章将介绍XSBase板子的 JTAG, Bootloader, Kenel, Filesystem，和Tiny-X 的制作。

关于命令的解释： [root$super]#在主机平台上执行，[root@XSBase /]$在XSBase板子上执行。

3.1 准备

首先，请先确保我们提供的CD是完好的。CD的目录在第1.2.4节找到。

复制CD的内容到你的硬盘。

```
[root$super ]# mount /dev/cdrom /mnt/cdrom
[root$super ]# mkdir /XSBASE
[root$super ]# cp –a /mnt/cdrom/* /XSBASE
```

挂载光驱，创建/XSBASE目录，然后复制CD内容到这个目录中。

在下面章节的操作中/XSBASE是默认的目录。

3.2 Loading Bootloader, Kernel, 和 Filesystem Image

下面介绍怎样下载Bootloader, Kernel, 和 Filesystem。在这节中用已经做好的光盘中自带的映像文件。

3.2.1 下载 Bootloader

下载Bootloader有两种不同的方法。

如果Bootloader没有下载就使用20pin JTAG接口。

```
[root$super root]# cd Image
[root$super Image]# ./Jflash-XSBASE x-boot255
```

当你使用仿真器下载bootloader 时使用20 pin的JTAG接口连接仿真器，请参考仿真器使用手册。

当bootloader已经下载了，使用TFTP。

1. 配置TFTP请参考第5章 Network。

2. 复制bootloader 映像文件到 /tftpboot 目录。

```
ot$super root]# cd Image
ot$super Image]# cp x-boot255 /tftpboot
```

3. 运行minicom，打开板子电源开关。

4. 出现启动信息。

```
XSBASE-R1
 Copyright (C) 2002 EMDOOR Co,. ltd.
 Support: http://www.emdoor.com

Autoboot in progress, press any key to stop .
Autoboot aborted
Type "help" to get a list of commands
XSBASE>
```

当你看到 “Autoboot in progress, press any key to stop” 时3秒内按下任意键，你将进入bootloader 命令模式。

5. 用TFTP命令下载bootloader 。

```
BASE> tftp x-boot255 loader
```

如果TFTP没有正常工作，请检查网络配置。参考第5章Network。

6. 复制到flash中。

```
XSBASE> flash loader
```

3.2.2 下载Kernel

1. 配置TFTP和BOOTP。参考第5章Network。

2. 复制kernel image 到/tftpboot 中。

```
[root$super root]# cd Image
[root$super Image]# cp zImage /tftpboot
```

3. 打开minicom，打开板子电源开关。

4. 显示启动信息。

```
XSBASE-R1
 Copyright (C) 2002 EMDOOR Co,. ltd.
 Support: http://www.emdoor.com

Autoboot in progress, press any key to stop .
Autoboot aborted
Type "help" to get a list of commands
XSBASE>
```

当你看到 “Autoboot in progress, press any key to stop” 时3秒内按下任意键，你将进入bootloader 命令模式。

5. 使用TFTP命令下载kernel 。

```
XSBASE> tftp zImage kernel
```

如果TFTP没有正常工作，请检查网络配置。参考第5章Network。

6. 复制到flash中。

```
XSBASE> flash kernel
```

3.2.3 下载Filesystem

1. 配置TFTP和BOOTP。参考第5章Network。

2. 复制filesystem image 到 /tftpboot 中。

```
[root$super root]# cd Image
[root$super Image]# cp rootfs.img /tftpboot
```

3. 打开minicom，打开板子电源开关。

4. 显示启动信息。

```
XSBASE-R1
 Copyright (C) 2002 Emdoor Co,. ltd.
 Support: http://www.emdoor.com

Autoboot in progress, press any key to stop .
Autoboot aborted
Type "help" to get a list of commands
XSBASE>
```

当你看到 “Autoboot in progress, press any key to stop” 时3秒内按下任意键，你将进入bootloader 命令模式。

5. 使用TFTP命令下载 filesystem。

```
XSBASE> tftp rootfs.img root
```

如果TFTP没有正常工作，请检查网络配置。参考第5章Network。

6. 复制到flash中。

```
XSBASE> flash root
```

3.2.4 启动linux

如果上述3个映像文件都成功地下载到FLASH中，那么Linux 启动条件将满足。开始启动。

```
XSBASE> boot
Starting kernel ...

Uncompressing Linux.....................
```

当电源打开时将出现同样的信息。

3.3 安装Toolchain

3.3.1 Toolchain

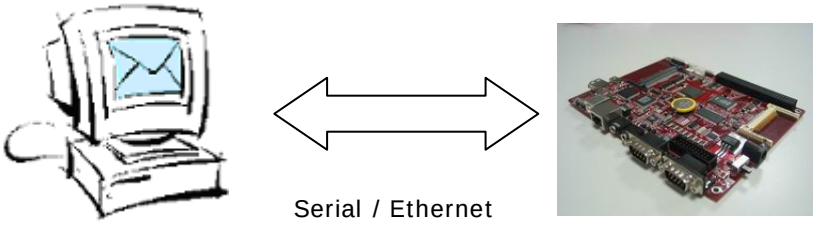
开发嵌入式系统，因为嵌入式系统的特有限制不可能装备很大的存储设备和友好的人机交互界面，所以一般开发环境(ToolChian)都需要安装在PC上。而通过Toolchain生成的最终目标文件将可以运行在相应的目标平台上。

ARM交叉编译环境不同于X86系列桌面的编译环境。因为XSBase采用的PXA255芯片同样是基于ARM体系结构的，所以在基于XSBase的开发过程中必须使用ARM的交叉编译环境。

这个编译器环境将使用下面的GNU工具。

- GNU gcc compilers for C, C++
- GNU binutil
- GNU C Library
- GNU C header

这个编译器使用上述的GNU交叉编译工具，编译后的二进制代码能在ARM中执行。



Host system　　XSBase

在主机平台上用交叉编译环境生成二进制代码 (bootloader, kernel, and applications)通过以太网或串口下载到XSBase。

3.3.2 安装Toolchain

注意：XSBase在本手册中在Redhat 9.0进行测试。如果在其它版本的Linux中出现问题，请联系我们的技术支持或者访问我们的网站。

```
[root$super root ]# cd /XSBASE/Toolchain
[root$super Toochain ]# ls
super-arm-linux-R1.1.tar.gz
[root$super Toochain ]# cp super-arm-linux-R1.1.tar.gz /usr/local
[root$super Toochain ]# cd /usr/local
[root$super local ]# tar xvzf super-arm-linux-R1.1.tar.gz
```

移动到/XSBASE的Toolchain目录下。super-arm-linux-R1.1.tar.gz是Toolchain的压缩文件。复制到/usr/local目录下解压。解压完后生成super-arm-linux-R1.1目录。

```
[root$super root ]# vi ~/.bash_profile
PATH=$PATH:$HOME/bin
PATH=$PATH:/usr/local/super-arm-linux-R1.1/bin        ← ADD
[root$super root ]# source ~/.bash_profile
```

然后设置路径，用VI编辑器打开/root/.bash_profile文件并添加上述的路径。现在在任何的目录下都能打开/usr/local/-arm-linux-R1.1/bin。

3.3.3 测试

这个编译器生成ARM的二进制代码，不同于在X86系列的GCC生成的二进制代码。所以请参考专门的编译过程和选项的文档。

现在用一个简单的程序测试这个编译器。

• 请分别使用gcc and arm-linux-gcc 编译下面一个短小的例子。

```
#include <stdio.h>
int main()
{
        printf("Hello World");
}
[root$super ]# gcc –o hello hello.c
[root$super ]# arm-linux-gcc –o hello-arm hello.c
```

• 用"file"命令检查生成的二进制代码。

```
[root$super ]# file hello
hello: ELF 32-bit LSB executable, Intel 80386, version 1, dynamically linked (uses shared libs), not stripped
[root$super ]# file hello-arm
hello-arm: ELF 32-bit LSB executable, Advanced RISC Machines ARM, version 1, dynamically linked (uses shared libs), not stripped
```

• 下载这个二进制文件到板子，执行。 "Hello World" 将出现。

3.4 JTAG Building

JTAG的功能：一是板子的调试，二是bootloader的下载。下面将介绍JTAG常用的功能和要点，如果想了解更多的JTAG逻辑和内部的结构请参考其它文档。

3.4.1 什么是JTAG?

70年代中期测试板子是使用“bed-of-nails"技巧直接接触PCBs(print circuit boards)的方式（现在也在使用叫zig的测试设备）。但是随着板子各端子之间的距离缩小， 测试越发困难， 最终随着多层板子的出现上述测试方法彻底失败， 经常在测试过程中损坏硬件或引起高额成本费用等问题。为解决这些问题， 在80年代中期组成了Joint European Test Access Group组织， 发展了 “a serial shift register around the boundary of the device”的概念。

通常，Boundary-Scan比JTAG更常用，JTAG芯片内部有Boundary Cell， 可与外部的pin脚一对一地连接起来，通过中间的Cell来人为地执行处理器的所有操作， 因而可进行各种硬件的测试和检查连接状态等。

3.4.2 JTAG 功能

JTAG提供的功能可以激活设备所有的外围管脚并读取它们的值，与CPU状态无关。

- 截取所有外部连接信息 (connecting each pin one-on-one to outside)
- 每一个cell通过串行移位寄存器连接(边界扫描寄存器)。
- 全部的接口由5个管脚控制(TDI, TMS, TCK, nTRST, TDO)。
- 测试器件的线路和电气连接。
- 测试设备间的连接。
- Flash烧写

3.4.3 创建JTAG

移动到JTAG目录下。如果"Makefile"可用，使用"make"命令编译JTAG。当编译成功将生成Jflash-XSBASE二进制文件。

为了编译XSBase的Jflash，必须配置Jflash-XSBASE目录下的Compile_switches.h文件。

Jflash-XSBASE Compile_switches.h

```
//#define DEBUG

//#define INSIGHT_JTAG
#define PARALLEL_JTAG

/*
 * Super pxa255 board platform
 */
//#define XSBASEA     // 16bit flash memory
#define XSBASEB     // 32bit flash memory ← For using XSbase
```

```
[root$super ]# cd /XSBASE/Jflash-XSBASE
[root$super Jflash-XSBASE]# make
[root$super Jflash-XSBASE]# ./Jflash-XSBASE ../Image/x-boot255
```

假设bootloader映像文件存在，输入上述命令可以通过JTAG把bootloader映像文件写入板子上FLASH存储器的起始地址为0的存储空间中。

3.5 Bootloader Building

通常PC机上常见的BIOS不会用在嵌入式系统中。Bootloader实现了这样的功能。Bootloader初始化硬件如CPU，SDRAM，FLASH，UART，& GPIO等，并为用户提供系统引导等功能。

3.5.1 Bootloader的功能

Bootloader的主要功能：

- 初始化硬件。
  初始化CPU clock, Memory timing, interrupt, UART和GPIO。

- 启动Linux
  这是bootloader最重要的功能.
  它将内核映像复制到SDRAM中并跳转到内核入口地址处。

- 下载Image
  下载内核和文件镜像到SDRAM中。下载只能通过以太网。
  Ethernet – tftp and bootp

- Flash 存储器管理

- 用write, erase, lock, and unlock 等命令管理Flash 存储器。

3.5.2 Bootloader 命令

• **help**

Usage	Help
Description	display simple desciption of each command.
Arguments	None
Example	XSBASEB> Help

表3-1 Bootloader command (help)

• **reload**

Usage	reload [kernel]
Description	Copy Kernel image saved into flash memory into SDRAM. Run automatically during Autoboot to copy Kernel image into SDRAM
Arguments	Kernel ? Copy Kernel image from flash memory into SDRAM 0xa0008000
Example	XSBASEB> reload kernel

表3-2 Bootloader command(reload)

• **bootp**

Usage	Bootp
Description	Run bootp to bring ip assigned to the target board and necessary information of the host PC from the host PC installed with bootpd.
Arguments	None
Example	XSBASEB> bootp

表3-3 Bootloader command(bootp)

• **tftp**

Usage	tftp [file] [loader/kernel/root] tftp [file] [addr]
Description	Download the host data to SDRAM via ethernet.
Arguments	file – name of the file in the host PC to transfer loader – save the transferred file into loader sector of SDRAM 0xa0000000 kernel – save the transferred file into kernel sector of SDRAM 0xa0008000. root – save the transferred file into 0xa0000000. Addr – data saving address in SDRAM
Example	XSBASEB> tftp zImage kernel XSBASEB> tftp zImage 0xa0000000

表3-4 Bootloader command(tftp)

• **flash**

Usage	flash [loader/kernel/root] flash [dest] [src] [len]
Description	write the data from SDRAM into specified address of flash memory
Arguments	loader – save loader sector of SDRAM 0xa0000000 into flash memory 0x0 Kernel – save kernel sector of SDRAM 0xa0008000 into flash memory. root – save root sector of SDRAM 0xa0000000 into flash memory 0x1c0000. dest – data saving address of flash memory src – address of source data len – length to save
Example	XSBASE> flash kernel XSBASE> flash 0xc0000 0xa0000000 0x100000

表3-5 Bootloader command(flash)

• **erase**

Usage	erase [loader/kernel/root] erase [addr1] erase [addr2] [len]
Description	erase specified flash block.
Arguments	loader – erase loader sector of flash memory. kernel – erase kernel sector of flash memory. root – erase root sector of flash memory. addr1 – base address of block of flash memory to erase addr2 – base address of beginning block of flash memory to erase len – length to erase
Example	XSBASE> erase kernel XSBASE> erase 0xc0000 XSBASE> erase 0xc0000 0x100000

表3-6 Bootloader command(erase)

• **lock**

Usage	lock [kernel/root] lock [addr1] lock [addr2] [len]
Description	lock the specified flash block.
Arguments	kernel – lock kernel sector of flash memory. root – lock root sector of flash memory. Addr1 – lock the address of flash memory. addr2 – lock the address of flash memory by len. len – ? length to lock
Example	XSBASE> lock kernel XSBASE> lock 0xc0000 XSBASE> lock 0xc0000 0x100000

表3-7 Bootloader command(lock)

• **unlock**

Usage	Unlock
Description	unlock flash memory.
Arguments	None
Example	XSBASE> unlock

表3-8 Bootloader command(unlock)

• **boot**

Usage	Boot boot [opt1] [opt2] boot [addr] [opt1] [opt2]
Description	execute kernel in SDRAM. execute kernel in specified address or by specified arguments.
Arguments	opt1 – kernel option(Only 0) opt2 – machine type(X-Hyer250 : 200) addr – kernel image address
Example	XSBASE> boot XSBASE> boot 0 200 XSBASE> boot 0xa0008000 0 200

表3-9 Bootloader command(boot)

• **memcpy**

Usage	Memcpy [dest] [src] [len]
Description	copy to specific address inside of SDRAM
Arguments	dest – destination address src – source address len – length to copy
Example	XSBASE> memcpy 0xa0000000 0xa1000000 0x10000

表3-10 Bootloader command(memcpy)

• **memdump**

Usage	memcpy [addr] [len]
Description	output memory contents.
Arguments	addr – beginning output address len – output length
Example	memdump 0xa0000000 0x100

表3-11 Bootloader command(memdump)

• **hexdump**

Usage	hexdump [addr] [len]
Description	output memory contents in hex value.
Arguments	addr – beginning output address len – output length
Example	hexdump 0xa0000000 0x100

表3-12 Bootloader command(hexdump)

• **memcmp**

Usage	memcmp [addr1] [addr2] [len]
Description	compare values in specified sectors of SDRAM.
Arguments	addr1 – beginning address addr2 – end address len – length
Example	XSBASE> memcmp 0xa0000000 0xa1000000 0x10000

表3-13 Bootloader command(memcmp)

• **memset**

Usage	memset [addr] [value] [len]
Description	set desired value to specified sector of SDRAM
Arguments	addr – beginning address value – hex value in char(8bit) type to set len – length to set
Example	XSBASE> memset 0xa0000000 0xf 0x10000

表3-14 Bootloader command(memset)

• **write**

Usage	write {c/s/l} [addr] [value]
Description	write desired value into specified sector.
Arguments	c – 8bit s – 16bit l – 32bit addr – address value – value to write
Example	XSBASE> write s 0xa0000000 0x1234 XSBASE> write l 0xa0000000 0x12345678

表3-15 Bootloader command(write)

• **read**

Usage	read {c/s/l} [addr]
Description	read data from the specified sector
Arguments	c – 8bit s – 16bit l – 32bit addr – address
Example	XSBASE> read s 0xa0000000 XSBASE> read l 0xa0000000

表3-16 Bootloader command(read)

• **status**

Usage	Status
Description	report the status
Arguments	None
Example	Status

表3-17 Bootloader command(status)

• **reboot**

Usage	Reboot
Description	Soft reset
Arguments	None
Example	Reboot

表3-18 Bootloader command(reboot)

3.5.3 生成Bootloader

1. Toolchain必需安装来编译bootloader。参考第3.3节。

2. 解压

```
[root$super root]# cd /XSBASE/Bootloader
[root$super Bootloader]# tar xzvf Boot-XSBASE.tar.gz
[root$super Bootloader]# cd Boot-XSBASE
```

3. 编译

为了编译XSBase板子的Bootloader，你必须配置Bootloader/Boot-XSBASE/src/include目录下的config.h的文件。

Bootloader/Boot-XSBASE/src/include/config.h

```
/*-----------------Must select XSBASEA or XSBASE--------------------*/

//#define XSBASEA
#define XSBASE                  // for using XSbase

/*------------------------------------------------------------------------*/
```

```
[root$super Boot-XSBASE]# make
```

如果编译成功生成src/x-boot255文件。X-boot255是bootloader镜像文件。

使用Jflash-XSBASE下载它或者使用已经下载的bootloader将它写入FLASH存储器中。

3.6 Kernel Building

在这节中，光盘中提供XSBase内核编译的资源。制作XSBase内核需要给标准的内核安装补丁。

Linux Kernel : linux-2.4.18.tar.gz
XSBase Kernel : 2.4.18-rmk7-pxa1-XSBASE.tar.gz

3.6.1 XSbase 内核制作

编译光盘提供的XSBase内核文件2.4.18-rmk7-pxa1-XSBASE.tar.gz将会生成zImage映像文件。

1. 移动到内核资源目录。

```
[root$super ]# cd /XSBASE/Kernel/2.4.18-rmk7-pxa1-XSBASE
[root$super 2.4.18-rmk7-pxa1-XSBASE]# ls
COPYING         MAINTAINERS          REPORTING-BUGS       drivers
        init     lib                 scripts                    CREDITS
        Makefile        net Rules.make          kernel                  ipc
                        mm              fs
Documentation README                    arch                    include
```

2. “make menuconfig.”命令打开内核配置。

基本的配置已经做好了，请保存退出。

```
[root$super 2.4.18-rmk7-pxa1-XSBASE]# make menuconfig
```

3. “make dep.”检查依赖性

```
[root$super 2.4.18-rmk7-pxa1-XSBASE]# make dep
```

4. “make zImage.”生成内核映像

```
[root$ 2.4.18-rmk7-pxa1-XSBASE]# make zImage
```

如果编译没有出错，在arch/arm/boot 路径下生成映像文件zImage。使用TFTP下载到板子上。（参考第3.2.2节）

3.6.2 XSBase Kernel configuration

CONFIGURATION	TYPE	CONFIGURATION	TYPE
CONFIG_ARM	Y	CONFIG_UID16	Y
CONFIG_RWSEM_GENERIC_SPINLOCK	Y	CONFIG_EXPERIMENTAL	Y
CONFIG_MODULES	Y	CONFIG_KMOD	Y
CONFIG_ARCH_PXA	Y	CONFIG_ARCH_XSBASE	Y
CONFIG_ARCH_XSBASE	Y	CONFIG_PXA_USB	Y
CONFIG_PXA_USB_NETLINK	Y	CONFIG_PXA_USB_CHAR	M
CONFIG_CPU_32	Y	CONFIG_CPU_32v5	Y
CONFIG_CPU_XSCALE	Y	CONFIG_ZBOOT_ROM_TEXT	0
CONFIG_ZBOOT_ROM_BSS	0	CONFIG_HOTPLUG	Y
CONFIG_PCMCIA	Y	CONFIG_PCMCIA_PXA	Y
CONFIG_MMC	Y	CONFIG_MMC_PXA	Y
CONFIG_MMC_BLOCK	Y	CONFIG_MMC_PARTITIONS	Y
CONFIG_NET	Y	CONFIG_SYSVIPC	Y
CONFIG_BSD_PROCESS_ACCT	Y	CONFIG_SYSCTL	Y
CONFIG_FPE_NWFPE	Y	CONFIG_KCORE_ELF	Y
CONFIG_BINFMT_ELF	Y		
CONFIG_CMDLINE：root=1f03 rw console=ttyS0,115200 init=/linuxrc			
CONFIG_LEDS	Y	CONFIG_LEDS_TIMER	Y
CONFIG_LEDS_CPU	Y	CONFIG_ALIGNMENT_TRAP	Y
CONFIG_MTD	Y	CONFIG_MTD_PARTITIONS	Y
CONFIG_MTD_CHAR	Y	CONFIG_MTD_BLOCK	Y
CONFIG_MTD_CFI	Y	CONFIG_MTD_GEN_PROBE	Y
CONFIG_MTD_CFI_ADV_OPTIONS	Y	CONFIG_MTD_CFI_NOSWAP	Y
CONFIG_MTD_CFI_GEOMETRY	Y	CONFIG_MTD_CFI_B4	Y
CONFIG_MTD_CFI_I2	Y	CONFIG_MTD_CFI_INTELEXT	Y
CONFIG_MTD_PXA_XSBASE	Y	CONFIG_BLK_DEV_LOOP	M
CONFIG_PACKET	Y	CONFIG_PACKET_MMAP	
CONFIG_FILTER	Y	CONFIG_UNIX	Y
CONFIG_INET	Y	CONFIG_NETDEVICES	Y
CONFIG_NET_ETHERNET	Y	CONFIG_CS8900	Y
CONFIG_PPP	M	CONFIG_PPP_ASYNC	M
CONFIG_PPP_DEFLATE	M	CONFIG_PPP_BSDCOMP	M
CONFIG_NET_PCMCIA	Y	CONFIG_PCMCIA_3C589	M
CONFIG_PCMCIA_3C574	M	CONFIG_PCMCIA_FMVJ18X	M
CONFIG_PCMCIA_PCNET	M	CONFIG_PCMCIA_AXNET	M

CONFIG_PCMCIA_NMCLAN	M	CONFIG_PCMCIA_SMC91C92	M
CONFIG_PCMCIA_XIRC2PS	M	CONFIG_IRDA	Y
CONFIG_PXA_FIR	Y	CONFIG_IDE	M
CONFIG_BLK_DEV_IDE	M	CONFIG_BLK_DEV_IDEDISK	M
CONFIG_VT	Y	CONFIG_VT_CONSOLE	Y
CONFIG_SERIAL	Y	CONFIG_SERIAL_CONSOLE	Y
CONFIG_UNIX98_PTYS	Y	CONFIG_UNIX98_PTY_COUNT	256
CONFIG_ADS7843	Y	CONFIG_AUTOFS_FS	Y
CONFIG_AUTOFS4_FS	Y	CONFIG_FAT_FS	M
CONFIG_MSDOS_FS	M	CONFIG_UMSDOS_FS	M
CONFIG_VFAT_FS	M	CONFIG_JFFS2_FS	Y
CONFIG_JFFS2_FS_DEBUG	0	CONFIG_TMPFS	Y
CONFIG_RAMFS	Y	CONFIG_PROC_FS	Y
CONFIG_DEVPTS_FS	Y	CONFIG_EXT2_FS	Y
CONFIG_NFS_FS	Y	CONFIG_NFS_V3	Y
CONFIG_SUNRPC	Y	CONFIG_LOCKD	Y
CONFIG_LOCKD_V4	Y	CONFIG_MSDOS_PARTITION	Y
CONFIG_NLS	Y	CONFIG_FB	
CONFIG_DUMMY_CONSOLE	Y	CONFIG_FB_PXA	Y
CONFIG_FBCON_CFB2	Y	CONFIG_FBCON_CFB4	Y
CONFIG_FBCON_CFB8	Y	CONFIG_FBCON_CFB16	Y
CONFIG_FONT_8x8	Y	CONFIG_FONT_8x16	Y
CONFIG_SOUND	Y	CONFIG_SOUND_PXA_AC97	Y
CONFIG_FRAME_POINTER	Y	CONFIG_DEBUG_USER	Y
CONFIG_DEBUG_INFO	Y	CONFIG_DEBUG_KERNEL	Y
CONFIG_MAGIC_SYSRQ	Y	CONFIG_DEBUG_BUGVERBOSE	Y
CONFIG_DEBUG_ERRORS	Y	CONFIG_DEBUG_LL	Y

表3-19 Kernel Configuration

3.7 制作Filesystem

3.7.1 制作Filesystem映像文件

1. 移动到文件系统目录

```
[root$super root]# cd /XSBASE/Filesystem
[root$super Filesystem]# ls
mkfs.jffs2      r.sh     rootfs.img      root_XSBASE   root_XSBASE.tar.gz
```

root_XSBASE 是filesystem的根目录。

举个简单的例子把第3.3.3节编译好的hello-arm加进去。

```
[root$super root_XSBASE]#
bin   dev  lib      lost+found  proc  root  tmp  var
conf  etc  linuxrc  mnt         rd    sbin  usr
[root$super root_XSBASE]# cp /root/hello-arm ./
bin   dev  lib      lost+found  proc  root  tmp  var
conf  etc  linuxrc  mnt         rd    sbin  usr    hello-arm
```

hello-arm按上面一样加到文件系统的根目录中。

2. 因为文件系统是JFFS2，所以把修改后的文件系统做成JFFS2映像文件。

```
[root$super root_XSBASE]# cd ..
[root$super Filesystem]# ls
mkfs.jffs2      r.sh     rootfs.img      root_XSBASE   root_XSBASE.tar.gz
[root$super Filesystem]# ./mkfs.jffs2 -o root -e 0x40000 -r
                         root_XSBASE -p –l
```

使用mkfs.jffs2 是为了把 root_XSBASE目录制作成JFFS2 映像文件。

- mkfs.jffs2

用法: ./mkfs.jffs2 –o outfile –e erase size -r dirname –p -l

Dirname 是文件系统根目录， outfile 是生成的文件名。

3. 下载文件系统映像到板子。

3.7.2 Filesystem Flow

下面介绍文件系统的加载和执行过程。

1.	Root filesystem mount	Mount by Kernel. Mount the third partition of MTD since root=1f03 is entered in command line.
2.	/linuxrc exec	Set option init=/linuxrc in command line to run /linuxrc first after mounting. If the option is not set, init process will run. /linuxrc mounts /etc/tmp, /etc/var, and /root to ramfs and copies /mnt/var to /etc/var. When power goes off, saved data in ramfs will be disappered. Therefore, it will prevent log or other data from being saved in flash memory. Run init process at the end.
3.	/sbin/init exec	init process is the parent process. Init will run configuration file, /etc/inittab.
4.	/etc/inittab	Configuration file of init process. Set default runlevel and run system configuration, /etc/rc.d/rc.sysinit.
5.	/etc/rc.d/rc.sysinit	Run once during Booting and it will perform various system configuration. Put in desired code here, and it will run during booting.
6.	/etc/inittab	Run /etc/rc.d/rc 3 since default runlevel is set to 3. # Runlevel 0 is halt. # Runlevel 3 is multi-user. # Runlevel 6 is reboot.
7.	/etc/rc.d/rc 3	Run various demon or util according to runlevel configured in inittab. Run etc/rc.d/rc3.d/ when runlevel is 3.

8.	/etc/inittab	Run login process, /sbin/getty -L ttyS0 115200 vt100. Login is ttyS0.

9.	/sbin/getty	Handle Login. If user id and password are correct, bash shell will run.

10.	/etc/inittab	Enter /usr/X11R6/bin/x start to run X-server and Window manager.

3.8 制作Tiny-X

3.8.1 什么是Tiny-X?

Tiny-X 是 Kdriver Tiny X Server的缩写，由Keith Packard设计。它是在Xfree86 server的基础上改写的，为了在小容量内存的环境下运行的。

Tiny-X的库大小远远比XFree86 server的小很多而且不需要环境配置文件。当Tiny-X编译后，环境参数已经配置好。

默认的Kdriver Server是Xfbdev，它设计时用到/dev/fb(Frame buffer)。Xfbdev支持如下：

- 支持 Linux OS
- 普通Linux 键盘
- 普通鼠标
- Linux fbdev display (unaccelerated).
- 支持触摸屏

Reference: The KDrive Tiny X Server
(http://www.pps.jussieu.fr/~jch/software/kdrive.html)

3.8.2 Getting XFree86

XSBase 使用 Xfree86 4.2.0。
警告：当你使用其他版本时，可能有一些错误发生。

光盘提供XFree86 source 。
你可以通过ftp://ftp.xfree86.org/pub/XFree86 得到XFree86的其它版本。

3.8.3 编译和安装

- 解压

```
[root$super root ]# cd /XSBASE/Tiny-X
[root$super Tiny-X]# ls
X420src-1.tgz            XSBASE_xc-patch-0.1.gz
[root$super Tiny-X]# tar xvzf X420src-1.tgz
[root$super Tiny-X]# ls
X420src-1.tgz            XSBASE_xc-patch-0.1   xc
```

- 补丁

XFree86需要修改使之与XSBase兼容。

请打我们提供的补丁XSBASE_xc-patch-0.1.gz。

```
[root$super Tiny-X]# ls
X420src-1.tgz          XSBASE_xc-patch-0.1.gz          xc
[root$super Tiny-X]# cp XSBASE_xc-patch-0.1.gz xc
[root$super Tiny-X]# cd xc
[root$super xc]# gzip –cd XSBASE_xc-patch-0.1.gz | patch –p1
patching file config/cf/cross.def
patching file config/cf/host.def
patching file config/cf/kdrive.cf
patching file programs/Xserver/Imakefile
patching file programs/Xserver/hw/kdrive/Hyper255/Imakefile
patching file programs/Xserver/hw/kdrive/Hyper255/Hyper255.c
patching file programs/Xserver/hw/kdrive/Hyper255/ts.c
patching file programs/Xserver/hw/kdrive/linux/Imakefile
patching file programs/Xserver/hw/kdrive/linux/ts.c
[root$super xc]#
```

- 编译

```
[root$super xc]# make World   // Compile
[root$super xc]# make install   // Install
```

按上面的编译和安装如果没有看到错误，“Make install”安装所用的程序和库到ProjectRoot中（在/usr/X11R6中），配置config/cf/host.def文件。

- 复制到XSBase文件系统根目录

复制主机平台的/usr/X11R6-arm到XSBase文件系统的/usr/X11R6-arm/ 目录下。

警告： 主机平台的/usr/X11R6-arm是很大的，我们只复制必要的程序和库。

如果你仅仅想修改X server，你可以将XSBASE 复制到文件系统的根目录。

按上述的复制，制作JFFS2映像文件并再次下载文件系统。

设备驱动 4

本章将介绍XSBase用到的设备信息。

4.1 Frame Buffer

Frame buffer是显示屏用来临时存储显示信息的存储设备。XSBase 从SDRAM中分配一块空间给Frame buffer。

XSBase的显示屏是16位色TFT LCD (640 x 480象素)。

在frame buffer 驱动代码（drivers/video/pxafb.c）中能找到关于PXA255信息。

▪ 驱动代码

drivers/video/fbcon.c	Low level operations for display
drivers/video/fbmem.c	Initialize frame buffer
drivers/video/pxafb.c	PXA255 LCD controller Freame buffer driver
include/linux/linux_logo.h	linux booting logo for XSbase

▪ 设备结点

/dev/fb0	frame buffer Node

▪ 配置内核

```
Character devices  --->
  [*] Virtual terminal
Console drivers  --->
  Frame-buffer support  --->
    [*] Support for frame buffer devices (EXPERIMENTAL)
    [*]   PXA LCD support
    (16-Bpp) LCD Bit Dept
```

▪ LCD 控制寄存器值

- LCD_PIXCLOCK	0
- LCD_BPP	16
- LCD_XRES	640
- LCD_YRES	480
- LCD_HORIZONTAL_SYNC_PULSE_WIDTH	1
- LCD_VERTICAL_SYNC_PULSE_WIDTH	1
- LCD_BEGIN_OF_LINE_WAIT_COUNT	200
- LCD_BEGIN_FRAME_WAIT_COUNT	0

```
- LCD_END_OF_LINE_WAIT_COUNT            4
- LCD_END_OF_FRAME_WAIT_COUNT           0
- LCD_SYNC     (FB_SYNC_HOR_HIGH_ACT | FB_SYNC_VERT_HIGH_ACT)
- LCD_LCCR0    (LCCR0_LDM | LCCR0_SFM | LCCR0_IUM |
                LCCR0_EFM | LCCR0_QDM | LCCR0_BM | LCCR0_OUM |
                LCCR0_PAS)
- LCD_LCCR3    (LCCR3_PCP | LCCR3_PixClkDiv(0x1) | LCCR3_Bpp(0x04) |
                LCCR3_Acb(0xff))
```

4.2 以太网

配置Cirrus Logic®公司的10M以太网芯片CS8900A驱动。

XSBase 使用静态片选1(0x04000000)。

在arch/arm/mach-pxa/XSBASER1.1.c.文件建立内存图。

- 虚拟内存设置

virtual	physical	length	domain	r	w	c	b
0xf0000000	0x08000000	0x00100000	2	0	1	0	0

- IRQ EDGE 设置

arch/arm/mach-pxa/XSBASER1.1.c

XSBASER1_init_irq()

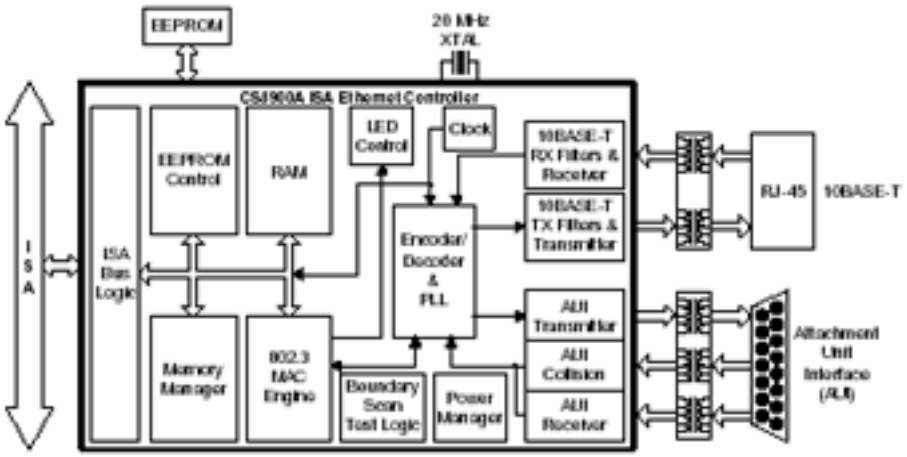
```
set_GPIO_IRQ_edge( 0, GPIO_RISING_EDGE);   /* Ethernet Interrupt */
```

- 驱动资源

drivers/net/cs8900.c	cs8900 ethernet driver

- 配置内核

```
Network device support  --->
  Ethernet (10 or 100Mbit)  --->
    [*] Ethernet (10 or 100Mbit)
    <*>   CS8900 support
```



Picture 4-1 cs8900 block diagram

Reference：CIRRUS LOGIC CS8900A Product Data Sheet

4.3 声卡

PXA255包含AC'97控制单元 (ACUNIT)。

ACUNIT支持音频控制器(AC-link)，它能通过串口传输数字音频、调制调解器、音频输入、控制寄存器和状态信息。

CIRRUS LOGIC CS4297 用作音频解码器。

- 外部接口连接到音频解码器

Name	Direction	Description summary
nACRESET	O	Active-low Codec reset. The Codec's registers reset when nACRESET is asserted.
GP28/BITCLK	I	12.288 MHz bit-rate clock.
GP31/SYNC	O	48 kHz frame indicator and synchronizer.
GP30/SDATA_OUT	O	Serial audio output data to Codec for digital-to-analog conversion.
GP29/SDATA_IN_0	I	Serial audio input data from Primary Codec.
GP32/SDATA_IN_1	I	Serial audio input data from Secondary Codec.

图 4-2 External interface to codecs

- 驱动资源

drivers/sound/ac97_codec.c	AC97 mixer/modem module
drivers/sound/ pxa-ac97.c	AC97 interface for PXA255
drivers/sound/ pxa-audio.c	audio interface for PXA255

- 设备结点

/dev/dsp	audio input, audio output
/dev/mixer	control of volume, sampling rates, bass, treble, etc

- 配置内核

```
Sound  --->
  <*> Sound support
  <*>    Intel PXA255/210 AC97 audio
```

- 设置Mixer

Mixer通过smixer程序生成。

设置在 /etc/rc.d/rc.sysinit。

```
[root@XSBASE /]$ /sbin/smixer –s /mnt/root/smixer.conf
```

/mnt/root/smixer.conf

```
vol     Vol             100
vol     Pcm             88
vol     Spkr            0
vol     Line            0
vol     Mic             100
vol     CD              0
vol     IGain           100
vol     Line1           0
vol     PhoneIn         0

# set recording source
recsrc  Mic
recsrc  Mic
```

- 测试

```
Record：
[root@XSBASE /]$ cat /dev/dsp > test.wav
Play：
[root@XSBASE /]$ cp sample.wav /dev/dsp

MP3 TEST
[root@XSBASE /]$ splay /mnt/root/NO1.mp3
```

Test用gqmpeg在X-server上完成。

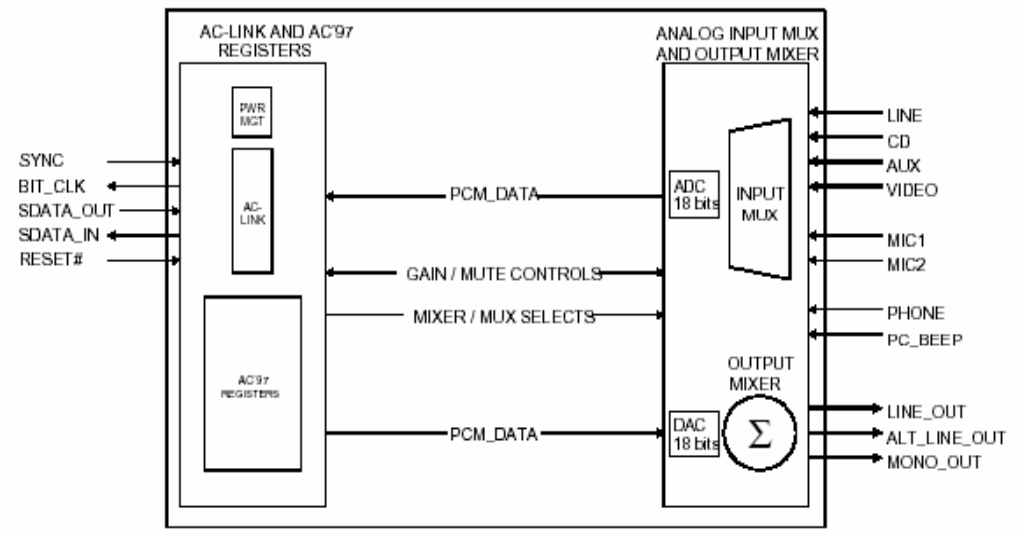


图4-3 cs4297 block diagram

4.4 RTC(RTC4513)

当关闭XSBase电源后，RTC4513通过一个电池来保持当前时间。

RTC4513通过3个串口接口信号来控制。

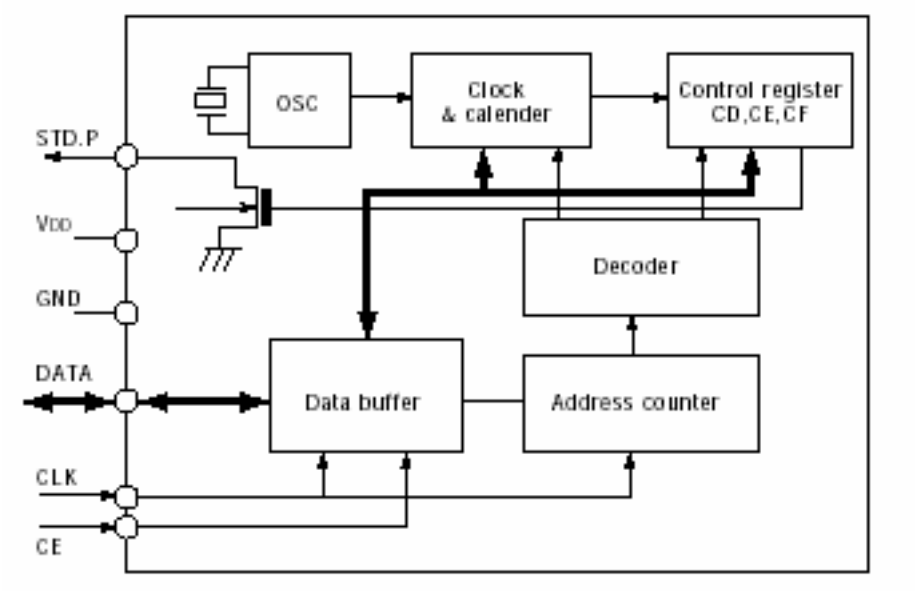


图4-4 RTC4513 block diagram

▪ 驱动资源

drivers/char/rtc.c	rtc4513 driver for XSBase

▪ 设备结点

/dev/rtc	RTC4513

▪ 配置内核

```
Character devices  --->
  <*> XSBase RealTime Clock
```

▪ 设置当前时间(XSBaseR1_date)

用法: ./XSBaseR1_date [YY.MM.DD-hh:mm:ss]

```
[root@XSBASE /]$ XSBaseR1_date
Tue Jun 10 13:47:24 KST 2002
[root@XSBASE /]$ XSBaseR1_date 2002.1.1-20:10:10
```

Reference：RTC4513 Product Data Sheet

4.5 UARTs

PXA255支持3个UARTs，比如全功能UART (FFUART)、蓝牙UART (BTUART)和标准UART (STUART)。

XSBase 用FFUART做控制台，STUART用于红外设备IrDA，BTUART用于接蓝牙模块。

- 驱动资源

drivers/char/serial.c	serial driver for PXA255

- 设备结点

/dev/ttyS0	FFUART(main serial console)
/dev/ttyS1	BTUART(extra serial)
/dev/ttyS2	STUART(IrDA)

- 配置内核

```
Character devices  --->
  <*> Standard/generic (8250/16550 and compatible UARTs) serial support
    [*]   Support for console on serial port
```

4.6 USB

PXA255 支持USB slave。

XSBase 仅仅支持USB-Ethernet。USB-Ethernet驱动的主要功能是模拟从USB到Ethernet数据格式。在这节中介绍怎么通过PING和TFTP使用USB-Ethernet。

主机平台内核补丁是为了在XSBase上使用USB-Ethernet 。

关于在主机平台上编译内核请参考其它文档。这节中介绍怎样仅仅把usbnet.c文件当做一个模块来编译。这种编译方法有可能因内核版本的不同有所不同。

主机平台的USB-Ethernet驱动是usbnet.c文件。我们推荐你拷贝我们提供的drivers/usb/usbnet.c到主机平台的内核，当然你也可以直接修改主机平台内核中的drivers/usb/usbnet.c文件。

假设主机平台内核路径是/usr/src/linux。

```
[root$super]#   cp linux-2.4.18-rmk7-pxa1-XSBASE/drivers/usb/usbnet.c
                /usr/src/linux/drivers/usb
```

```
[root$super ]# cd /usr/src/linux
[root$super linux]# make menuconfig
USB support   --->
<M> Support for USB
<M>    USB-to-USB Networking cable device support (EXPERIMENTAL)
[root$super linux]# make dep
[root$super linux]# make modules
```

当内核编译成功，usbnet.o文件生成在/usr/src/linux/drivers/usb里。

```
[root$super ]# cd /usr/src/linux/drivers/usb
[root$super usb]# insmod usbnet.o
```

当你按上述命令把usbnet当成一个模块导入时，主机平台使用usbnet的条件已经满足。

当你不想用insmod导入那个模块时，可以拷贝usbnet.o到/lib/modules/kernel-version/kernel/drivers/usb。

▪ 驱动资源

```
arch/arm/mach-pxa/usb_ctl.c
  - interrupt routing and overall coordination
arch/arm/mach-pxa/usb_ep0.c
  - PXA USB controller driver
arch/arm/mach-pxa/usb_recv.c
  - receive layer for the PXA USB client function
arch/arm/mach-pxa/usb_send.c
  - xmit layer for the PXA USB client function
arch/arm/mach-pxa/usb-eth.c
  - Ethernet driver for the PXA USB client function
```

▪ 编译内核

```
  System Type  --->
    Intel PXA250/210 Implementations  --->
      <*> PXA USB function support
      <*>    Support for PXA USB network link function
```

▪ 内核启动信息

```
  PXA USB Controller Core Initialized
  USB Function Ethernet Driver Interface
```

▪ USB网络设置

Host PC IP(usb0)：192.168.1.10, XSBase IP(usbf)：192.168.1.5

- XSBase

```
[root@XSBASE /root]$ifconfig usbf up 192.168.1.5
[root@XSBASE /root]$route add default gw 192.168.1.10 usbf
[root@XSBASE /root]$ifconfig
eth0       Link encap:Ethernet  HWaddr 00:12:34:56:78:9A
           inet addr:192.168.100.50  Bcast:192.168.100.255
           Mask:255.255.255.0
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:0 errors:0 dropped:0 overruns:0 frame:0
           TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:100
           RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
           Base address:0x300
```

```
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

usbf      Link encap:Ethernet  HWaddr 58:1F:3F:C0:5C:2E
          inet addr:192.168.1.5  Mask:255.255.255.0
          UP RUNNING  MTU:1500  Metric:1
          RX packets:9 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:644 (644.0 b)  TX bytes:770 (770.0 b)

[root@XSBASE /root]$
```

按上述配置，用USB连接到主机平台上，将在主机平台上看到下面的信息。

- Host PC

```
[root$super ]# tail –f /var/log/message
Apr  8 15:43:01 bedguy kernel: usb.c: registered new driver usbnet
Apr  8 16:24:58 bedguy kernel: usb0: register usbnet 001/005, Linux Device
Apr  8 16:24:59 bedguy kernel: NET4: Linux IPX 0.46 for NET4.0
Apr  8 16:24:59 bedguy kernel: IPX Portions Copyright (c) 1995 Caldera,
Apr  8 16:24:59 bedguy kernel: NET4: AppleTalk 0.18a for Linux NET4.0
```

```
[root$super ]# ifconfig usb0 192.168.1.10
```

用ifconfig设置usb0的IP 192.168.1.10。当XSBase和主机平台连接上，你可以按如下使用USB-Ethernet。

```
[root@XSBASE /root]$ping 192.168.1.10
PING 192.168.1.10 (192.168.1.10): 56 data bytes
64 bytes from 192.168.1.10: icmp_seq=0 ttl=255 time=1.6 ms
64 bytes from 192.168.1.10: icmp_seq=1 ttl=255 time=2.3 ms


[root@XSBASE /root]$ftp 192.168.1.10
Connected to 192.168.1.10.
220 BCB1COOL Server (Proftpd FTP Server) [bedguy]
Name (192.168.1.10:root): bedguy
331 Password required for bedguy.
Password:
230 User bedguy logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

4.7 PCMCIA & CF CARD

PXA255支持16-bit PC卡接口。

XSBase 的PCMCIA卡使用socket 0，CF卡使用socket 1。

插槽通过PSKTSEL来区分。

XSBase支持Prism II 芯片集的主机驱动程序。

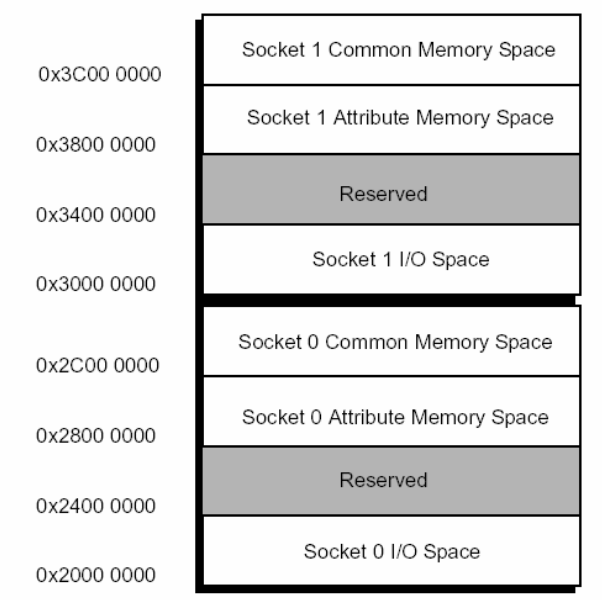


图4-5 16Bit PC CARD Memory Map

▪ 驱动资源

drivers/pcmcia/pxa/XSBASE.c

This is for Socket services approaching PCMCIA socket, detecting number of sockets, sensing insertion and removal, and applying power. It also provides interface with Card services.

drivers/pcmcia/cistpl.c, rsrc,mg.c, bulkmem.c cs.c.

This is for Card services controlling allotment and recovery of system resource such as memory and interrupt when the cards are detected. It also provides interface with Driver services.

drivers/pcmcia/ds.c

This is for Driver services providing interface with Card manager such as Client drivers and cardmgr.

▪ 配置内核

```
General setup   --->
  [*] Support hot-pluggable devices
  PCMCIA/CardBus support   --->
    <*> PCMCIA/CardBus support
    <*>    PXA250/210 support
```

▪ 安装

XSBase 已经在内核中包含了PCMCIA卡的驱动，所以运行cardmgr不需要单独的初始化，卡插入时它会自动检测到。

```
[root@XSBASE /root]$/etc/rc.d/pcmcia restart          // restart
```

▪ 配置工具和监视PCMCIA卡

- cardmgr

它监视PCMCIA sockets的状态，监测卡的插入和拔出并能自动加载合适的客户端驱动程序。

- /var/run/stab

cardmgr 保存每一个socket 信息。

Socket 0 : Wiress LAN CARD and Socket 1 : CF Memory CARD 32M are used for the test.

```
[root@XSBASE /root]$cat /var/run/stab
  Socket 0: Z-Com XI300 11Mb/s WLAN Card
  0       network hostap_cs       0       wlan0
  Socket 1: ATA/IDE Fixed Disk
  1       ide     ide_cs  0       hda     3       0
```

- cardctl

检查sockets的状态。

```
[root@XSBASE /root]$cardctl config
  Socket 0:
    Vcc 5.0V  Vpp1 0.0V  Vpp2 0.0V
    interface type is "memory and I/O"
    irq 24 [exclusive] [level]
    function 0:
      config base 0x03e0
        option 0x41
     io 0xf6000000-0xf600003f [16bit]
  Socket 1:
```

```
    Vcc 3.3V  Vpp1 0.0V  Vpp2 0.0V
    interface type is "memory and I/O"
    irq 38 [exclusive] [level]
    function 0:
      config base 0x0200
        option 0x41 status 0x00 pin 0x00 copy 0x00
    io 0xf7000000-0xf700000f [auto]

[root@XSBASE /root]$cardctl ident
  Socket 0:
    product info: "RFTNC", "RF-TNC1100N", ""
   manfid: 0xd601, 0x0002
    function: 6 (network)
  Socket 1:
    product info: "SunDisk", "SDP", "5/3 0.6"
    manfid: 0x0045, 0x0401
  function: 4 (fixed disk)
```

Reference：Intel® PXA255 Applications Processor Developer's Manual

4.8 MMC(多媒体卡)

PXA255的MMC控制器支持标准的MMC卡和串行外设接口SPI。XSBase255仅仅提供MMC卡的驱动。

- 驱动资源

drivers/mmc/mmc_core.c	MMC subsystem core implementation
drivers/mmc/mmc_pxa.c	Driver for PXA255 MMC controller
drivers/mmc/mmc_block.c	Driver for the block device on the MMC

- 驱动结点

/dev/mmcda	MMC node
/dev/mmcda1	MMC partition 1
/dev/mmcda2	MMC partition 2
/dev/mmcda3	MMC partition 3

- 配置内核

```
General setup   --->
  MMC device drivers   --->
  <*> Multi Media Card support (NEW)
  <*> PXA250 MMC driver (NEW)
  <*> MMC block driver (NEW)
  [*]    MMC partitioning support (NEW)
```

- MMC分区设置

```
[root@XSBASE /root]$fdisk /dev/mmcda
Partition check:
 mmca: mmca1

The number of cylinders for this disk is set to 31360.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
    (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help):p
Disk /dev/mmcda: 1 heads, 1 sectors, 31360 cylinders
Units = cylinders of 1 * 512 bytes
```

```
     Device Boot      Start         End      Blocks   Id  System
/dev/mmcda1               2       31360       15679+  83  Linux

/* If partition is created already, delete that partition. */
Command (m for help): d
Partition number (1-4): 1
Command (m for help): p

Disk /dev/mmcda: 1 heads, 1 sectors, 31360 cylinders
Units = cylinders of 1 * 512 bytes

     Device Boot      Start         End      Blocks   Id  System

Command (m for help):

/* Create partition. */
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (2-31360, default 2): ↵
Using default value 2
Last cylinder or +size or +sizeM or +sizeK (2-31360, default 31360):↵
Using default value 31360

Command (m for help): p

Disk /dev/mmcda: 1 heads, 1 sectors, 31360 cylinders
Units = cylinders of 1 * 512 bytes

     Device Boot      Start         End      Blocks   Id  System
/dev/mmcda1               2       31360       15679+  83  Linux

Command (m for help):

/* Save partition and then exit. */
Command (m for help): w
```

```
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.
[root@XSBASE /root]$
```

▪ MMC 格式化

```
[root@XSBASE /root]$mkfs.ext2 /dev/mmcda1
mke2fs 1.19, 13-Jul-2000 for EXT2 FS 0.5b, 95/08/09
 mmca: mmca1
 mmca: mmca1
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
3920 inodes, 15679 blocks
783 blocks (4.99%) reserved for the super user
First data block=1
2 block groups
8192 blocks per group, 8192 fragments per group
1960 inodes per group
Superblock backups stored on blocks:
        8193

Writing inode tables: done
Writing superblocks and filesystem accounting information: ↵
[root@XSBASE /root]$
```

- 挂起MMC

```
[root@XSBASE /root]$mount /dev/mmcda1 /mnt/ide/
 mmca: mmca1
 mmca: mmca1
[root@XSBASE /root]$ df
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/root                30976     20788     10188  68% /
/dev/mmcda1              15180        13     14384   1% /mnt/ide
[root@XSBASE /root]$ mount
/dev/root on / type jffs2 (rw)
ramfs on /etc/tmp type ramfs (rw)
ramfs on /etc/var type ramfs (rw)
ramfs on /root type ramfs (rw)
/proc on /proc type proc (rw)
none on /dev/pts type devpts (rw)
/dev/mmcda1 on /mnt/ide type ext2 (rw)
[root@XSBASE /root]$
```

网络 5

这章中将介绍bootp、tftp和nfs。

5.1 设置bootp 和 tftp

通过以太网从bootloader下载，在主机平台配置bootp和tftp服务器。

5.1.1 安装Bootp服务器

• 安装 bootp rpm包

bootp rpm 在提供的光盘的PRM目录里。
如果RMP在主机平台已经安装了，那就没有必要再安装一遍。

```
[root$super root]# cd /XSBASE/RPM
[root$super RPM]# rpm –i bootp-2.4.3-7.i386.rpm
```

• 新建bootp 文件

在/etc/xinetd.d目录下新建bootp文件并按如下配置。

/etc/xinetd.d/bootp

```
service bootps
{
        disable                 = no
        socket_type             = dgram
        protocol                = udp
        wait                    = yes
        user                    = root
        server                  = /usr/sbin/bootpd
}
```

• **新建bootptab 文件**

在 /etc下新建bootptab文件并按如下配置。

/etc/bootptab
test:\ ht=1:\ ha=0x123456789A00:\ ip=192.168.1.50:\ sm=255.255.255.0

test： 标记

ht： 硬件类型(1表示以太网)

ha: 硬件地址(这个地址必须和板子的MAC地址相同)

ip ： IP 地址 (如果板子的MAC地址和ha的地址一致的话，ip将被传输给板子).

sm：subnet mask

注意： 板子的ip地址和主机平台的ip地址必须在同一个网段内。

(例如 主机 IP: 192.168.10.100, 板子 IP：192.168.10.xxx)

• **重启 xined**

在主机平台重启xined。

```
[root$super root ]# /etc/rc.d/init.d/xinetd restart
Stopping xinetd :                                          [ OK ]
Starting   xinetd :                                        [ OK ]
```

5.1.2 Tftp

• **安装tftp rpm**

如果RMP在主机平台已经安装了，那就没有必要再安装一遍。

```
[root$super root]# cd /XSBASE/RPM
[root$super RPM]# rpm –i tftp-server-0.17-9.i386.rpm
```

• **新建tftp 文件**

在/etc/xinetd.d下新建tftp文件并按如下配置。

如果这个文件已经存在，你没有必要再新建一次。

/etc/xinetd.d/tftp

```
service tftp
{
        disable                 = no
        socket_type             = dgram
        protocol                = udp
        wait                    = yes
        user                    = root
        server                  = /usr/sbin/in.tftpd
        server_args              = -s /tftpboot
}
```

如果server_args 设置为tftpboot，那么当通过bootloader下载时仅/tftpboot目录下的文件能被传输。

Set up different directory here if preferred.

然后拷贝内核和文件系统到/tftpboot，使用bootloader的tftp命令下载。参考第3.2节。

如完成上述步骤后tftp还不工作，请检查防火墙是否存在。关闭防火墙使用lokkit命令。

5.2 NFS

NFS 能够共享主机平台的部分文件系统。使用超出自己容量的资源变成可能。

• Setting up the host

/etc/exports	
/mnt/nfs	(rw,no_root_squash)

重启 nfs demon

```
[root$super root ]# /etc/rc.d/init.d/nfs stop
[root$super root ]# /etc/rc.d/init.d/nfs start
```

• 板子

```
[root@XSBASE /root]$ portmap
[root@XSBASE /root]$ mount –t nfs 192.168.1.12:/mnt/nfs /mnt
```

把主机平台/mnt/nfs目录挂设在板子的/mnt目录下。

GDB 6

本章将介绍用GDB调试的方法。

6.1 GDB简介

GDB是一个交叉调试工具，用来观察程序内部的运行情况。

GDB的4个主要功能帮助使用者去发现Dugs：

* 运行你的程序，设置所有能影响程序运行的东西。

* 保证程序在指定的条件下停止。

* 当程序停止时，让你检查发生了什么。

* 改变你的程序。那样你就可以试着修正某个 bug 引起的问题，然后继续查找另一个 bug。

6.2 GDB 资源

XSBase使用GDB version 5.3 版本，它在XSBase测试过。其他版本的GDB版本应该和XSBase兼容。

GDB v5.3 可以从互联网上下载得到。

ftp://ftp.gnu.org/pub/gnu/gdb/gdb-5.3.tar.gz

6.3 编译GDB

GDB用ARM体系的GCC编译，因为XSBase的Xscale PXA255芯片是ARM内核。Tool chain 必须在主机平台上重新安装，来编译用于ARM的GDB。否则，请参考手册第3.3节Toolchain building。

6.3.1 编译GDB Server(libthread_db-1.0.so)

1. 解压gdb-5.3.tar.gz.

```
[root$super root]# mkdir /XSBASE/temp
[root$super root]# cp gdb-5.3.tar.gz /XSBASE/temp
[root$super temp]# cd /XSBASE/temp
[root$super temp]# tar xvzf gdb-5.3.tar.gz
```

2. 运行configure 创建 make 文件，执行make。

```
[root$super temp]# cd gdb-5.3
[root$super gdb-5.3]# ./configure --target=arm-linux --prefix=/usr/local/arm-gdb -v
[root$super gdb-5.3]# make
[root$super gdb-5.3]# make install
```

♣ 配置选项

--target： 选择arm-linux 因为XSBase是基于ARM的。

--prefix ： 当你执行make install时选择安装路径

3. 如果编译没有任何错误，arm-linux-gdb将生成在 /usr/local/arm-gdb/bin。

Arm-linux-gdb 运行在主机平台上成为GDB 服务器，用来远程调试XSBase板子。

6.3.2 GDB 客户端编译

1. 移动到GDB目录编译

```
[root$super root]# cd /XSBASE/temp/gdb-5.3
[root$super gdb-5.3]# export PATH=$PATH:/usr/local/arm-gdb/bin
[root$super gdb-5.3]# ./configure --target=arm-linux --host=arm-linux
```

♣ 配置选项

--target： 选择arm-linux 因为XSBase是基于ARM的。

--host： 选择arm-linux 因为XSBase是基于ARM的。

2. XSBase的客户端程序在gdb/gdbserver。

```
[root$super gdb-5.3]# cd gdb/gdbserver
[root$super gdbserver]# make CC=/usr/local/super-arm-linux-R1/bin/arm-linux-gcc
```

3. 如果你按上面的编译会出现下面的错误信息。

```
…
…
linux-arm-low.c:26: sys/reg.h: No such file or directory
```

解决这个错误，请改正gdb/gdbserver/config.h文件。

gdb/gdbserver/config.h

```
#define HAVE_SYS_REG_H 1
-> //#define HAVE_SYS_REG_H 1
```

按上述的注释“define HAVE_SYS_REG_H 1”句子。

4. 再执行make。

```
[root$super gdbserver]# make CC=/usr/local/super-arm-linux-R1/bin/arm-linux-gcc
```

5. 如果没有编译错误gdbserver将生成在gdb/gdbserver目录下。

这个文件是GDB客户端程序，在XSBase下运行。

6.4 GDB 调试

GDB通过XSBase板子的以太网进行调试。因此，必须使主机平台和XSBase之间的TCP/IP协议可用。

首先，检查主机平台和XSBase之间的通信。

默认XSBase 的IP地址是192.168.100.50。

在这章，假设主机平台的 IP是192.168.100.216。

```
[root$super gdbserver]# ifconfig eth0 192.168.100.216
```

6.4.1 TEST 代码

1. 写一段简单的代码用GDB来调试。

/XSBASE/temp/test.c

```
#include <stdio.h>

main()
{
      int i;
      double j;
      char *str = NULL ;

     for(i=0; i<5; i++)
     {
              j = i/2 + i;
              printf("j is %f \n", j);
     }

      strcpy(str,"hello, world");
      printf("str is %s \n", str);

       return;
}
```

这段程序代码是把变量i和变量i/2加到j中，然后拷贝字符串到指针变量str并输出。

下面的输出是希望得到的。

```
j is 0.000000
j is 1.500000
j is 3.000000
j is 4.500000
j is 6.000000
str is hello. world
```

然而，它并没有得到上面的结果。用GDB调试程序。

2. 用arm-linux-gcc编译

```
[root$super temp]# /usr/local/super-arm-linux-R1/bin/arm-linux-gcc -g test.c -o test
```

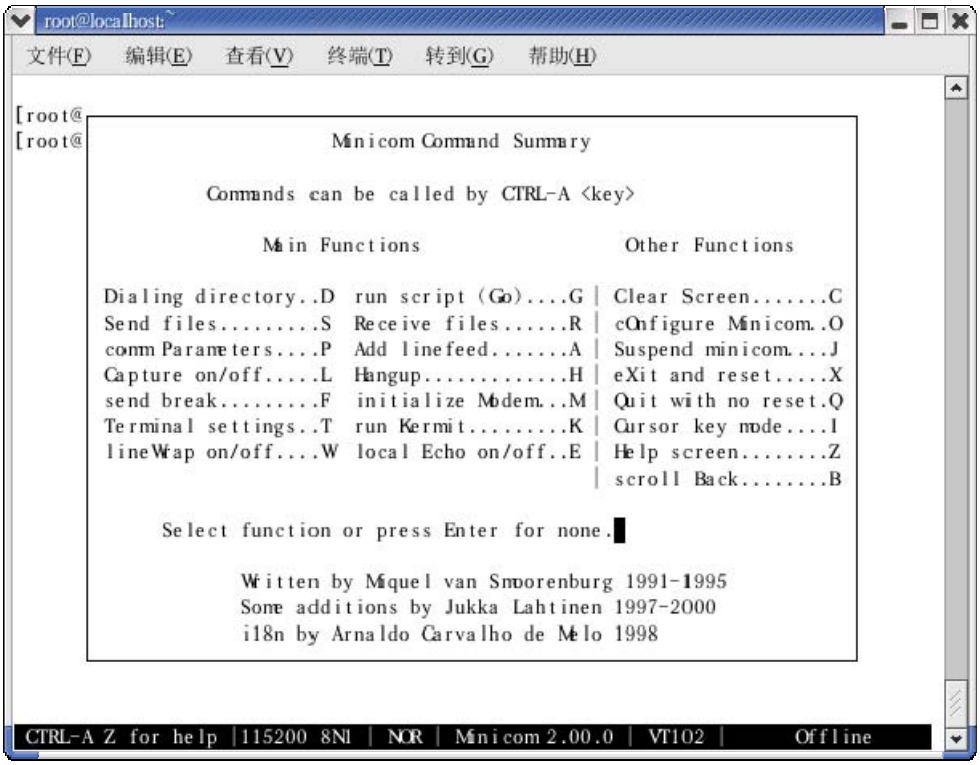
6.4.2 Transfer to XSBase.

下载test和gdbserver到XSBase。

Zmodem和ftp都可以用来下载。

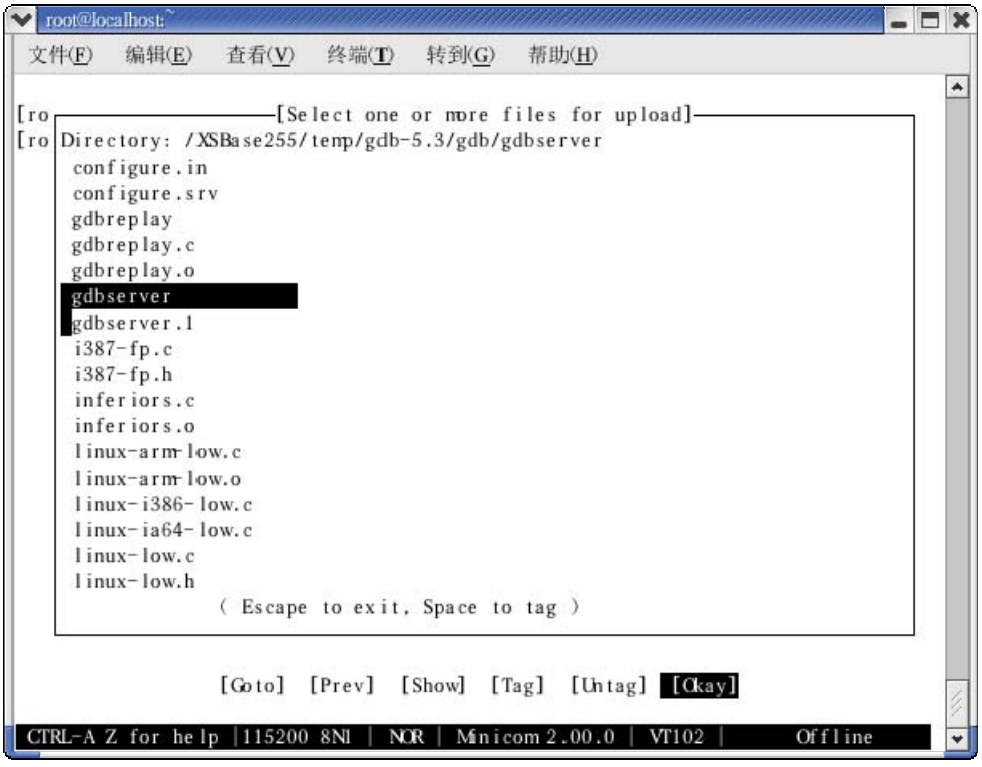
例如我们用zmodem来下载。

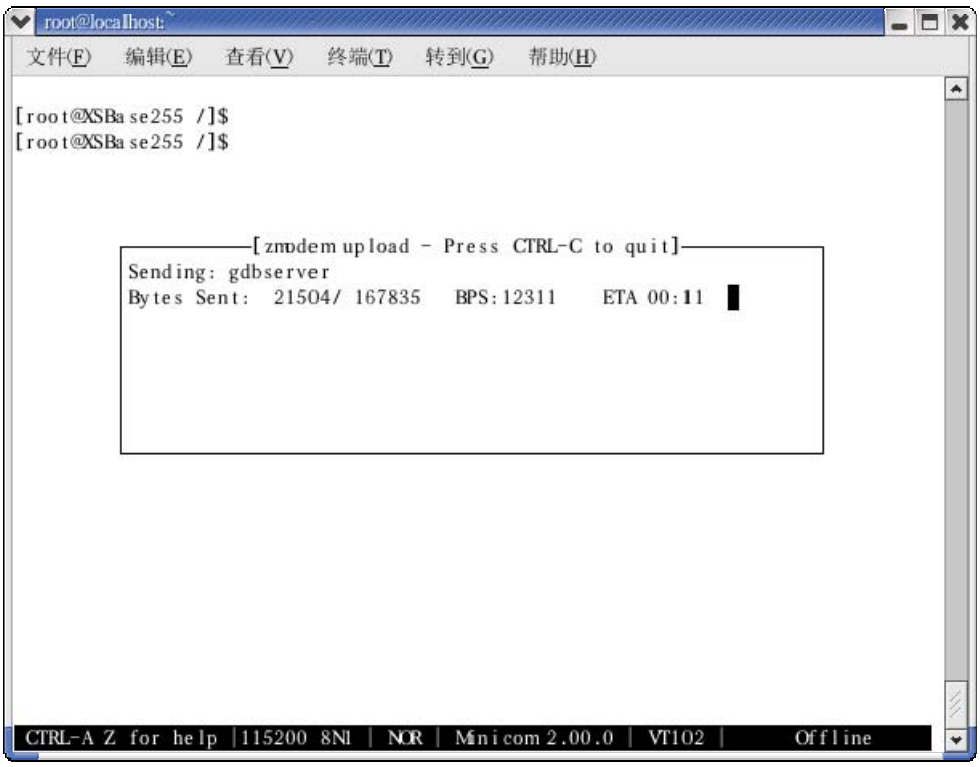
1. 按Ctrl+A然后再按Z键出现minicom配置窗口。



2．按s键选择zmodem。

3.你可以看到主机平台的目录，从/XSBASE/temp/gdb/gdbserver中下载gdbserver。





3. 用同样的方法下载test。

4. 执行test。

```
[root@XSBASE /root]$./test
j is 0.000000
j is 1.000000 pc : [<4009a8f8>]    lr : [<0000845c>]    Not tainted
j is 3.000000
j is 4.000000
j is 6.000000
sp : bffffd4c  ip : bffffd5c  fp : bffffd58
r10: 401342fc  r9 : 40133c88  r8 : 000083d4
r7 : 00000001  r6 : 00008280  r5 : 4001e724  r4 : bffffd94
r3 : 00000068  r2 : ffff7af3  r1 : 0000850d  r0 : 00000000
Flags: nzCv  IRQs on  FIQs on  Mode USER_32  Segment user
Control: 397F  Table: A3E54000  DAC: 00000015
Segmentation fault
[root@XSBASE /root]$
```

结果出现段错误，这不是我们希望得到的结果。

6.4.3 连接

1. 在XSBase板子上运行gdbserver。

```
[root@XSBASE /root]$ ./gdbserver 192.168.100.216:1234 test
Process test created; pid = 80
```

从服务器中等待一个请求。

The IP address of the server is 192.168.100.216 and the port is 1234.

2. 拷贝test程序到主机平台上的/usr/local/arm-gdb/bin目录下，然后执行arm-linux-gdb。

```
[root$super temp]# cp /XSBASE/test /usr/local/arm-gdb/bin
[root$super temp]# cd /usr/local/arm-gdb/bin
[root$super temp] # ./arm-linux-gdb test
GNU gdb 5.3
Copyright 2002 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "--host=i686-pc-linux-gnu --target=arm-linux".
(gdb)
```

3. 连接到XSBase。

```
(gdb) target remote 192.168.100.50:1234
Remote debugging using 192.168.100.50:1234
0x40002980 in ?? ()
```

板子的IP 地址是 192.168.100.50 ，端口是1234。

4. 如果连接成功的话，在XSBase上将出现下面的信息。

```
[root@XSBASE /root]$./gdbserver 192.168.100.216:1234 test
Process test created; pid = 80
Remote debugging from host 192.168.100.216
```

6.4.4 调试

1. 导入symbol文件。

```
(gdb) symbol-file test
Load new symbol table from "test"? (y or n) y

Reading symbols from test...done.
```

导入symbol文件后可以看到代码的内容。

2. 用list命令查看代码的内容。

```
(gdb) list
1       #include <stdio.h>
2
3       main()
4       {
5                       int i;
6                       double j;
7                       char *str = NULL ;
8
9                       for(i=0; i<5; i++)
10                      {
(gdb)
11                              j = i/2 + i;
12                              printf("j is %f \n", j);
13                      }
14
15                      strcpy(str,"hello, world");
16                      printf("str is %s \n", str);
17
18                      return;
19      }
20
```

3. 运行结果不是我们希望的，在第9行代码中设置一个断点来查找出错原因。

```
(gdb) break 9
Note: breakpoint 1 also set at pc 0x83ec.
Breakpoint 2 at 0x83ec: file test.c, line 9.
```

4. 执行代码。

```
(gdb) cont
Continuing.

Breakpoint 1, main () at test.c:9
9                       for(i=0; i<5; i++)
```

在第9行断点暂停。

5. 用单步命令step继续执行。

```
(gdb) step
11                              j = i/2 + i;
(gdb) step
12                              printf("j is %f \n", j);
(gdb) step
9                       for(i=0; i<5; i++)
```

XSBase 输出变量j的值。

```
[root@XSBASE root]$./gdbserver 192.168.100.216:1234 test
Process test created; pid = 161
Remote debugging from host 192.168.100.216
j is 0.000000
```

6. 在下面的单步中选中变量i和j的值。

```
(gdb) step
11                              j = i/2 + i;
(gdb) p i
$1 = 1
(gdb) p j
$2 = 0
(gdb)   step
12                              printf("j is %f \n", j);
(gdb) p j
$3 = 1
```

你会发现当i＝1时j＝1。

通过调试结果分析你发现j=i/2+1是不正确的。

正确的表达式是j = (double)i/2 + I。

7．查找其他的bug，在strcpy(str,"hello, world")这行设置一个断点。

```
(gdb) break 15
Breakpoint 4 at 0x8448: file test.c, line 15.
(gdb) cont
Continuing.

Breakpoint 4, main () at test.c:15
15                          strcpy(str,"hello, world");
```

在第15行断点暂停。

8．查看变量str的值，然后单步运行。

```
(gdb) p str
$4 = 0x0
(gdb) step

Program received signal SIGSEGV, Segmentation fault.
0x4009a8f8 in ?? ()
```

你看到在15行有一个段错误。

变量str的值是0X0。因为它没有指定地址导致出现段错误。

修正这个错误，在15行前面添加str = (char *)calloc(1, sizeof(char))。

9．修正了错误，退出GDB。

```
(gdb) c
Continuing.

Program terminated with signal SIGSEGV, Segmentation fault.
The program no longer exists.
(gdb) quit
```

想了解更多的细节，请参考GDB的文档。

XSBase 硬件 7

本章将介绍XSBase有关的硬件信息。

7.1 XSBase 模块图

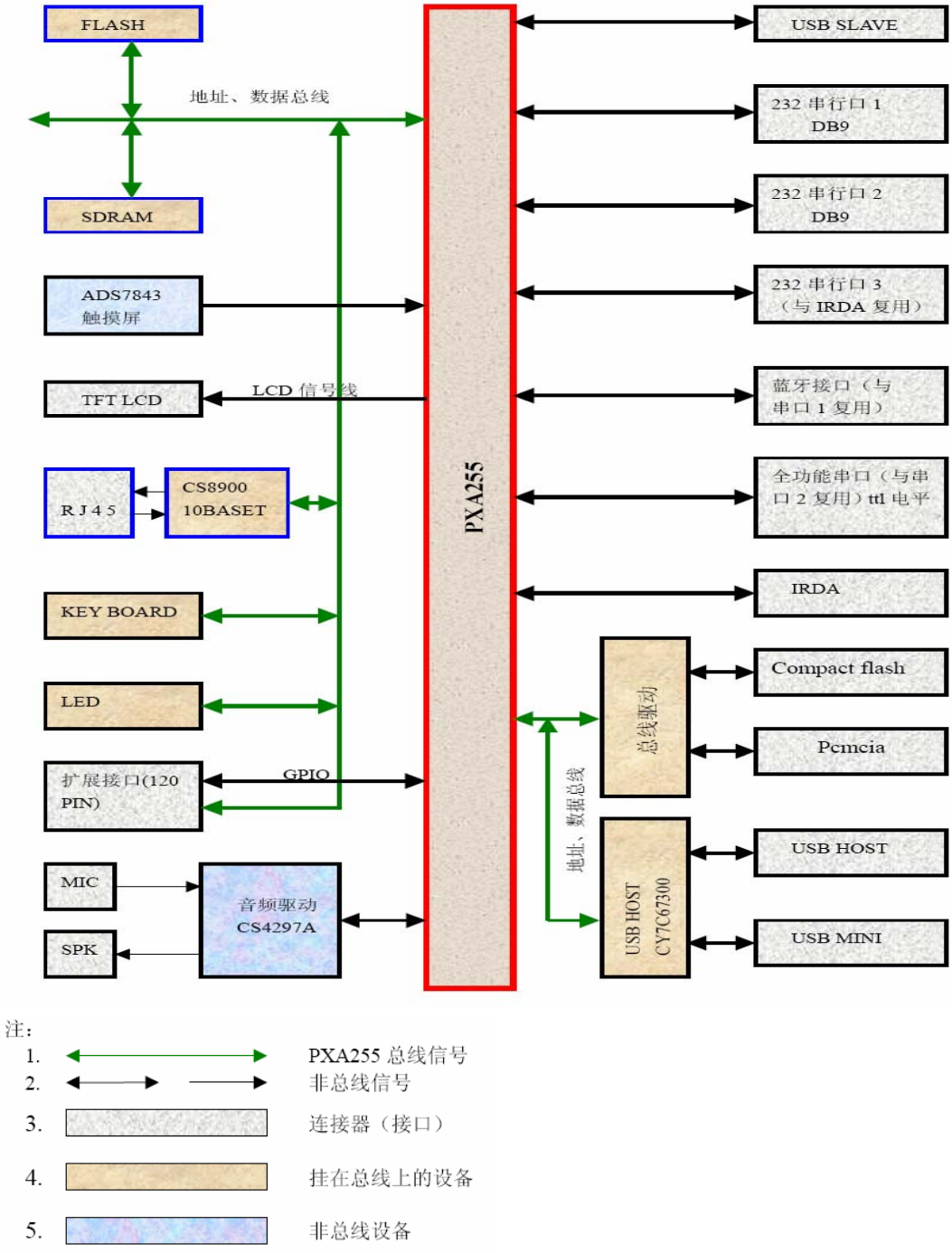


图7-1 XSBase block diagram

7.2 XSBase 内存地址映象

7.2.1 FLASH 内存地址映象

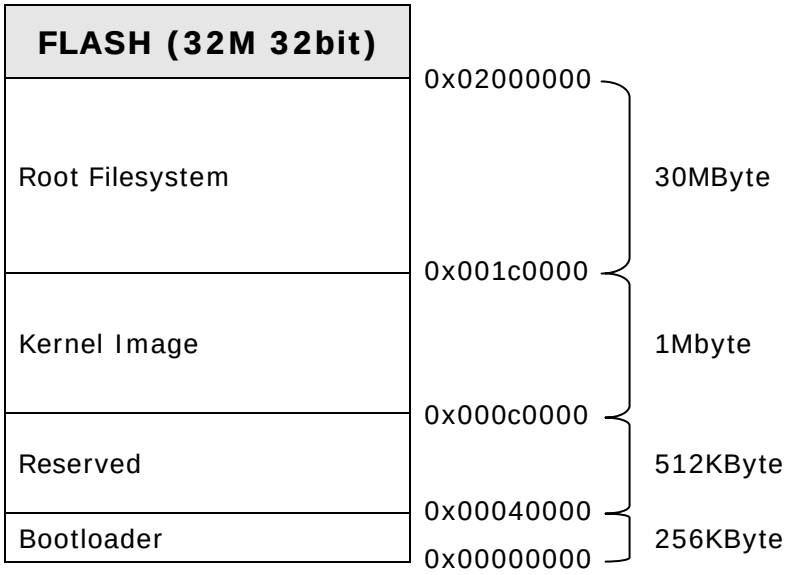


图7-2 Flash Memory Map

7.2.2 SDRAM 内存地址映象

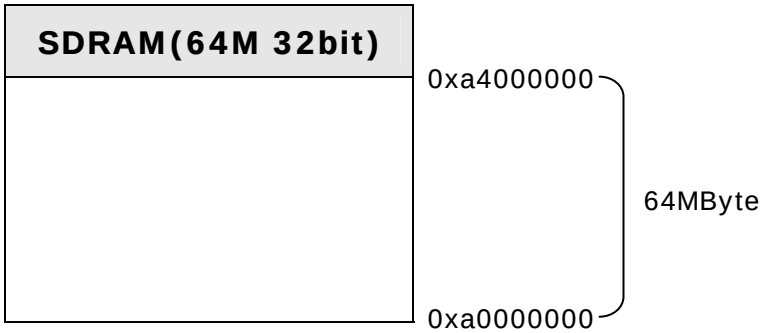


图7-3 SDRAM Memory Map

7.3 XSBase 连接器引脚描述

7.3.1 J1：USB Slave connector（USB Slave接口）

USB Slave connector如下表：

No.	Signal	Description
1	VCC	+5V Supply Voltage.(max. 500mA)
2	UDC-	Data - (Input to computer)
3	UDC+	Data + (Output from computer)
4	GND	Ground.

表7-1 USB Slave connector

7.3.2 J2：Serial [Bluetooth]（蓝牙串口）

蓝牙串口Serial [Bluetooth] 管脚描述如下表

No.	Name	Description
1	BT-RX	Bluetooth UART Receive (In)
2	BT-TX	Bluetooth UART Transmitt (Out)
3	BT-RTS	Bluetooth Data-Termial-Ready (Out)
4	BT-CTS	Bluetooth UART Clear-to-Send (In)
5	GND	Ground

表7-2 Serial(Bluetooth)

7.3.3 J3：Serial [Full Function UART]（全功能异步串口）

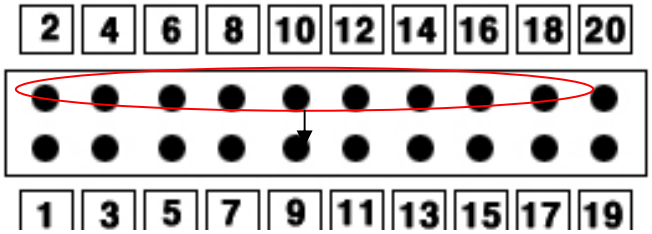
Full Function UART全功能异步串口引脚功能描述如下表：

No.	Name	Description
1	FF-DCD	Full Function UART Data-Carrier-Detect (In)
2	FF-RX	Full Function Receive (In)
3	FF-TX	Full Function Transmitt (Out)
4	FF-DTR	Full Function UART Data-Terminal-Ready (Out)
5	GND	Ground
6	FF-DSR	Full Function UART Data-Set-Ready (In)
7	FF-RTS	Full Function UART Request-to-Send (Out)
8	FF-CTS	Full Function UART Carrier-to-Send (In)
9	FF-RI	Full Function UART Ring-Indicator (In)

表7-3 Serial (Full Function UART)

7.3.4 J4：JTAG connector（JTAG接口）

The pin out of JTAG connector (20 pin)



No.	Name	No.	Name
1	3.3V (Processor I/O Voltage)	11	NC
2	NC(No connect)	12	GND
3	NTRST	13	TDO
4	GND	14	GND
5	TDI	15	System reset (nRESET)
6	GND	16	GND
7	TMS	17	NC
8	GND	18	GND
9	TCK	19	NC
10	GND	20	GND

表7-4 JTAG 20 pin connector

7.3.5 J5:Inverter Power(LCD逆变器电源连接器)

LCD逆变器电源连接器引脚功能如下表

No.	Name	Description
1	GND	GROUND
2	VIN	+5V

表7-5 Inverter Power

7.3.6 J6:TOUCH 4-wire(4线电阻触摸屏连接器)

TOUCH 4-wire (4 pin)4线电阻触摸屏引脚功能描述如下表：

No.	Name	Description
1	YU	Right
2	XL	Bottom
3	YL	Left
4	XR	Top

表7-6 Touch

7.3.7 J7：Power connector(电源接口)

Power connector (3 pin)引脚描述如下表

No.	Name	Description
1	GND	Ground
2	GND	Ground.
3	VIN	+5V Supply Voltage.

表 7-7 Power connector

7.3.8 J8：LCD connector （LCD接口）

The pin out of LCD connector (32 pin)

No.	Name	Description
2	SA-L-PCLK	Data input clock
4	SA-L-LCLK	Input data latch signal
5	SA-L-FCLK	Scan start-up signal
8	LDD11	Display data
9	LDD12	Display data
10	LDD13	Display data
11	LDD14	Display data
12	LDD15	Display data
14	LDD5	Display data
15	LDD6	Display data
16	LDD7	Display data
17	LDD8	Display data
18	LDD9	Display data
19	LDD10	Display data
22	LDD0	Display data
23	LDD1	Display data
24	LDD2	Display data
25	LDD3	Display data
26	LDD4	Display data
28	L-BIAS	LCD AC BIAS
1,6,7,13,20,21,27	GND	Ground
30,31,32	LCD3P3V	Power Supply
3,29	NC	No Connection

表 7-8 LCD connector

7.3.9 J9：MMC Card Socket（MMC卡插槽）

MMC Card Socket (7 pin)引脚功能描述如下表：

No.	Name	Description
1	MMC-CS0	MultiMedia Card Chip Select 0 (Out)
2	MMCMD	MultiMedia Card Command (Bidirectional).
3	GND	Ground
4	DC3P3V	Power Supply
5	MMC-CLK	MultiMedia Card Clock (Out)
6	GND	Ground
7	MMDAT	MultiMedia Card Data (Bidirectional)

表 7-9 MMC Card socket

7.3.10 J11,J12（红外接口、串口选择跳线）

红外接口、串口选择跳线，当选择红外接口时，J11、J12的1、2脚连接；当选择串口时，J11、J12的2、3连接。

No.	Name	Description
1	IR-RX	IRDA RXD
2	SW_IR-RX	PXA255 IRRXD
3	R1OUT	SP3223EEA R1OUT

No.	Name	Description
1	IR-TX	IRDA TXD
2	SW_IR-TX	PXA255 IRTXD
3	T1IN	SP3223EEA T1IN

表7-10 选择跳线

7.3.11 Serial(与红外接口复用)

No.	Name	Description
1	T1OUT	RS-232 发送数据(TXD)
2	T2OUT	未使用
3	R1IN	RS-232 接收数据(RXD)
4	R2IN	未使用
5	GND	Ground

表7-11 Serial

7.3.12 J13, J14 ：Audio（音频）

音频接口包括扬声器接口和麦克风接口，定义如下表

No.	Name	Description
J13	MIC Jack	AC`97 MIC Jack
J14	SPK Jack	AC`97 Speaker Jack

表 7-12 Audio

7.3.13 J15, J16 （扩展接口）

扩展接口为120PIN连接器，其引脚功能定义如下表：

J15

No.	Name	Description
2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52	XA-A[0..25]	PXA255 Address BUS
1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,51,53,55,57,54,56,58	XA-D[0..31]	PXA255 Data BUS
59	DC3P3V	Power Supply
60	GND	Ground

J16

No.	Name	Description
1	ADS7843-BUSY	Touch Screen Controller
3	PCMCIA-IREQ#	PCMCIA Interrupt Request
5	PCMCIA-CD#	PCMCIA Card Detect
7	ADS7843-DO	Touch Screen Controller
9	ADS7843-IRQ	Touch Screen Controller
11	MMC-CLK	MMC Card
13	GP7/48MHz_OUT	Available GPIO
15	MMC-CS0	MMC Card
17	RTC-SDA	Real Time Clock Chip
19	GP10/RTC_CLK	Available GPIO
21	GP11/3.6MHz-OUT	Available GPIO

23	GP12/32KHz-OUT	Available GPIO
25	GP13/MBGNT	Available GPIO
27	GP14/MBREQ	Available GPIO
29	CF-IREQ#	CF Interrupt Request
31	CF-CD#	CF Card Detect
33	DREQ1/GP19	Available GPIO
35	DREQ0/GP20	Available GPIO
37	DVAL0/GP21	Available GPIO
39	DVAL1/GP22	Available GPIO
41	AC_BITCLK/GP28	CS4297A
43	AC_SDIN0/GP29	CS4297A
45	AC_SDOUT/GP30	CS4297A
47	AC_SYNC/GP31	CS4297A
49	SYSCLK/AC_SDIN1/GP32	Available GPIO
53	RESET3P3	RESET (ACTIVE HIGH)
55	RESET3P3#	RESET (ACTIVE LOW)
2,4,6	XA-CS3# ~ 5#	Chip Select
8	XA-RDY	Ready
10	XA-OE#	Output Enable
12	XA-WE#	Write Enable
14	XA-RD_nWR	Read(High) and Write(LOW) Usage : BUS Buffer Control
18,20,22,24	XA-CAS-DQM-0# ~ 3#	Memory Control
26,28,30	XA-RAS-SDCS-1# ~ 3#	Memory Control
32	XA_SDCAS#	Memory Control
34	XA-SDRAS#	Memory Control
36,38	XA-SDCLK-0, XA-SDCLK-1	Memory Control
40	XA-SDCKE-0	Memory Control
44	SIBDO	Synchronous Serial Port Transmit.
46	SIBDI	Synchronous Serial Port Receive
48	SIBCLK	Synchronous Serial Port Clock
50	SIBSYNC	Synchronous Serial Port Frame
52	EXTCLK	Synchronous Serial Port External Clock
56	I2C-SCL	I2C Clock
58	I2C-SDA	I2C Data

59,60	DC3P3V, DC5V	Power Supply
16,42,51,54,57	GND	Ground

表7-13 External Interface

7.3.14 X1：Ethernet（以太网）

以太网连接器采用标准RJ45连接器，引脚功能描述如下表

No.	Name	Description
1	TPOP	Twisted-Pair Output Positive. Differential outputs to the twisted-pair cable.
2	TPON	Twisted-Pair Output Negative. Differential outputs to the twisted-pair cable.
3	TPIP	Twisted-Pair Input Positive. A differential input pair from the twisted-pair cable.
4	NC	Not Connect.
5	NC	Not Connect.
6	TPIN	Twisted-Pair Input Negative. A differential input pair from the twisted-pair cable.
7	NC	Not Connect.
8	NC	Not Connect.

表 7-14 Ethernet

7.3.15 X2 :Serial connector2（串口2）

serial connector采用DB9，其引脚功能描述如下表

No.	Name	Description
1	NC	
2	DB9_T1OUT	RS-232 发送数据(TXD)(from BT-TX)
3	DB9_R1IN	RS-232 接收数据(RXD)(to BT-RX)
4	NC	
5	GND	Ground
6	NC	
7	NC	
8	NC	
9	NC	

表7-15 Serial connector

7.3.16 X3 :Serial connector1（串口1）

No.	Name	Description
1	NC	
2	DB9_T2OUT	RS-232 发送数据(TXD)(from FF-TX)
3	DB9_R2IN	RS-232 接收数据(RXD)(to FF-RX)
4	NC	
5	GND	Ground
6	NC	
7	NC	
8	NC	
9	NC	

表7-16 Serial connector

7.3.17 X4 : PCMCIA Socket（PCMCIA插槽）

PCMCIA Socket (68 pin)引脚功能描述如下表：

No.	Name	Description
1,34,35,68	GND	Ground
30-32,2-6, 64-66,37-41	PCM-D[0..15]	PCMCIA Data Bus
7,42	PCE-1# PCE-2#	PCMCIA Card Enable 1 PCMCIA Card Enable 2
29-22,12,11,8, 10,21,13,14,20, 19,46-50,53-56	PCM-A[0..25]	PCMCIA Address Bus
9	POE#	PCMCIA Output Enable
15	PWE#	PCMCIA Write Enable
44	PIOR#	PCMCIA I/O read
45	PIOW#	PCMCIA I/O write
61	PREG#	PCMCIA Register select
58	PRRSET	PCMCIA Reset
43	PVS1	
57	PVS2	
63	PABVD1	Battery Voltage Detect 1
62	PABVD2	Battery Voltage Detect 2
16	PAIREQ#	PCMCIA Interrupt Request
59	PAWAIT#	PCMCIA Wait

33	PAIOS16#	
60	PAINPACK#	
36	PCD1#	Card Detect 1
67	PCD2#	Card Detect 2
17,51	VCC	Power Supply
18,52	VPP	Power Supply

表 7-17 PCMCIA Socket

7.3.18 X5：CF Socket （CF插槽）

CF Socket (50 pin)引脚功能描述如下表：

No.	Name	Description
1,50	GND	Ground
21-23,2-6, 47-49,27-31	CF-D[0..15]	CF Data Bus
7,32	CFCE-1# CFCE-2#	CF Card Enable 1 CF Card Enable 2
20-14,12-10,8	CF-A[0..10]	CF Address Bus
9	CFOE#	CF Output Enable
36	CFWE#	CF Write Enable
34	CFIOR#	CF I/O read
35	CFIOW#	CF I/O write
44	CFREG#	CF Register select
41	CF-RRSET	CF Reset
33	CF-VS1	
40	CF-VS2	
46	CFA-BVD1	Battery Voltage Detect 1
45	CFA-BVD2	Battery Voltage Detect 2
37	CFAIREQ#	CF Interrupt Request
42	CFAWAIT#	CF Wait
24	CFAIOS16#	
43	CFAINPACK#	
26	CFCD1#	Card Detect 1
25	CFCD2#	Card Detect 2
13,38	VCC	Power Supply

表7-18 CF Socket

7.3.19 X6 :USB MINI CONNECTOR(USB HOST)

No.	Name	Description
1	OTG-VBUS	USB OTG Vbus
2	VM-A	Data - (Input to computer)
3	VP-A	Data + (Output from computer)
4	OTGID	USB OTG ID
5,6,7,8,9	GND	Ground

表7-19 Serial connector

7.3.20 X7 :USB HOST CONNECTOR(A TYPE)

No.	Name	Description
1	USB5V	+5V
2	VM-B	Data - (Input to computer)
3	VP-B	Data + (Output from computer)
4	GND	Ground

表7-20 Serial connector

7.3.21 JP1 : Power Switch （电源开关）

5V Power Slide Switch

7.4 XSBase GPIO MAP

Alternate value

00- The corresponding GPIO pin (GPIO[x]) is used as a general purpose I/O.

01- The corresponding GPIO pin (GPIO[x]) is used for its alternate function 1.

10- The corresponding GPIO pin (GPIO[x]) is used for its alternate function 2.

11- The corresponding GPIO pin (GPIO[x]) is used for its alternate function 3.

Pin	Name	Alternate value	Description
GPIO0	CS8900-INT	00	CS8900 Interrupt
GPIO1	ADS7843-BUSY	00	ADS7843 BUSY
GPIO2	PCMCIA-IREQ	00	PCMCIA Interrupt Request
GPIO3	PCMCIA-CD	00	PCMCIA Card Detect
GPIO4	ADS7843-DO	00	ADS7843 Data Out
GPIO5	ADS7843-IRQ	00	ADS7843 Pen Interrupt
GPIO6	MMC-CLK	01	MMC Clock
GPIO7	Reversed	00	Reversed
GPIO8	MMC_CS0	01	MMC Chip Select 0
GPIO9	RTC_STA	00	RTC4513 DATA
GPIO10	RTC_CLK	00	RTC4513 Clock
GPIO11	Reversed	00	Reversed
GPIO12	Reversed	00	Reversed
GPIO13	Reversed	00	Reversed
GPIO14	Reversed	00	Reversed
GPIO15	SA-CS1	10	Active low chip select 1
GPIO16	CF_IREQ	00	CF Interrupt Request
GPIO17	CF_CD	00	CF Card Detect
GPIO18	RDY	01	Bus Ready
GPIO19	Reversed	00	Reversed
GPIO20	Reversed	00	Reversed
GPIO21	Reversed	00	Reversed
GPIO22	Reversed	00	Reversed
GPIO23	Reversed	00	Reversed
GPIO24	Reversed	00	Reversed
GPIO25	Reversed	00	Reversed
GPIO26	Reversed	00	Reversed
GPIO27	Reversed	00	Reversed

GPIO28	AC_BITCLK	10	AC97 bit_clk
GPIO29	AC_SDIN0	01	AC97 Sdata in0
GPIO30	AC_SDOUT	10	AC97 Sdata_out
GPIO31	AC_SYNC	10	AC97 sync
GPIO32	Reversed	00	Reversed
GPIO33	SA-CS5	10	Actice low chip select 5
GPIO34	FF-RX	01	FFUART receive
GPIO35	FF-CTS	01	FFUART Clear to send
GPIO36	FF-DCD	01	FFUART Data carrier detect
GPIO37	FF-DSR	01	FFUART data set readyT
GPIO38	FF-RI	01	FFUART Ring Indicator
GPIO39	FF-TX	10	FFUART transmit data
GPIO40	FF-DTR	10	FFUART data terminal Ready
GPIO41	FF-RTS	10	FFUART request to send
GPIO42	BT-RX	01	BTUART receive data
GPIO43	BT-TX	10	BTUART transmit data
GPIO44	BT-CTS	01	BTUART clear to send
GPIO45	BT-RTS	10	BTUART request to send
GPIO46	IR-RX	01	ICP receive data
GPIO47	IR-TX	10	ICP trandmit data
GPIO48	SA-POE	10	Out Put Enable for Card Space
GPIO49	SA-PWE	10	Write Enable for Card Space
GPIO50	SA-PIOR	10	I/O Read for Card Space
GPIO51	SA-PIOW	10	I/O Write for Card Space
GPIO52	SA-PCE-1	10	Card Enable for Card Space
GPIO53	SA-PCE-2	10	Card Enable for Card Space
GPIO54	SA-PSKTSEL	10	Socket Select for Card Space
GPIO55	SA-PREG	10	Card Address bit 26
GPIO56	SA-PWAIT	01	Wait signal for Card Space
GPIO57	SA-IOIS16	01	Bus Width select for I/O Card Space
GPIO58	LDD0	10	LCD data pin 0
GPIO59	LDD1	1010	LCD data pin 1
GPIO60	LDD2	10	LCD data pin 2
GPIO61	LDD3	1	LCD data pin 3
GPIO62	LDD4	10	LCD data pin 4
GPIO63	LDD5	10	LCD data pin 5
GPIO64	LDD6	10	LCD data pin 6
GPIO65	LDD7	10	LCD data pin 7

GPIO66	LDD8	10	LCD data pin 8
GPIO67	LDD9	10	LCD data pin 9
GPIO68	LDD10	10	LCD data pin 10
GPIO69	LDD11	10	LCD data pin 11
GPIO70	LDD12	10	LCD data pin 12
GPIO71	LDD13	10	LCD data pin 13
GPIO72	LDD14	10	LCD data pin 14
GPIO73	LDD15	10	LCD data pin 15
GPIO74	SA-L-FCLK	10	LCD Frame clock
GPIO75	SA-L-LCLK	10	LCD line clock
GPIO76	SA-L-PCLK	10	LCD Pixel clock
GPIO77	L-BIAS	10	LCD AC Bias
GPIO78	SA-CS2	10	Active low chip select 2
GPIO79	SA-CS3	10	Active low chip select 3
GPIO80	SA-CS4	10	Active low chip select 4

表7-16 GPIO MAP

技术支持 8

可以通过以下途径获得技术支持：

1. 亿道XScale技术论坛：www.XSBase.com
2. 专用E-MAIL技术支持：info@emdoor.com
3. 提供电话、传真咨询的技术支持

更多信息，欢迎访问亿道电子网站 http://www.emdoor.com

深圳市亿道电子技术有限公司

总部：深圳市深南大道6013号中国有色大厦506室
电子邮件：info@emdoor.com
电　　话：+86-755-83474891/2/3/4/5/6
传　　真：+86-755-83474895
公司主页：www.emdoor.com

上海分部：上海市普陀区金沙江路1066号申汉商务大厦C座1103
电　　话：+86-21-62650520,62643621
传　　真：+86-21-62655790

北京分部：北京市海淀区知春路22号知音商务写字楼512室
电　　话：+86-10-62375506,62375508
传　　真：+86-10-62376767