文章编号:1674 - 2869(2011)06 - 0089 - 05

基于 SPI 接口的无线网卡设备驱动设计

盛李立[1],王 忠[1],王春丽[1],王 浩[2]
(1.武汉工程大学计算机科学与工程学院,湖北 武汉 430074;
2.武汉大学电子信息学院,湖北 武汉 430072)

摘 要:介绍了 Marvell 88W8686 无线网卡芯片在嵌入式 linux 下的驱动设计与实现.该系统主要应用于心电监护仪和小型手持数据采集系统.首先搭建嵌入式开发的软硬件环境,对 linux 内核进行裁剪,然后研究网络驱动工作原理,接着对 Marvell 88W8686 无线网卡的驱动进行修改和交叉编译,最终移植到 ARM 平台上,建立嵌入式无线局域网.

关键词:ARM;嵌入式 linux;无线网卡;802.11g;设备驱动
中图分类号:TU111.1 **文献标识码**:A **doi**:10.3969/j.issn.1674-2869.2011.06.021

0 引 言

最近几年,在国内外医疗市场上,WIFI 医疗设备的应用正以前所未有的速度增长.众所周知,相比于传统的有线网络,无线局域网的应用价值体现在:可移动性、布线容易、组网灵活和成本优势,另外,无线网络通信范围不受环境条件的限制.本文设计了一套基于 Marvell 88W8686 无线网卡芯片的无线通信系统,通过对 SPI 无线网卡的 Linux 设备驱动的深入理解和分析,成功地移植在 S3C2440 ARM 处理器上[2].实现了嵌入式系统的无线局域网接入.利用该平台,可以进一步设计完善医用心电监护仪和手持数据采集系统的无线数据传输,使得控制人员能够远离数据采集现场,而通过远程终端来控制现场数据和各种控制信号,获得较好的便利性,同时较好地解决了安全性问题.

1 硬件系统构成

1.1 Marvell 88W8686 无线网卡芯片介绍

无线网卡是无线局域网(WLAN)的重要组成部分,WLAN 的物理层及 MAC 层是用无线网卡的硬件及其软件完成的,而 LLC 层以上各层均由计算机软件来实现.WLAN 包括进行通信的网络接口卡(简称无线网卡)和接入点/桥接器(AP/网桥).其中,无线网卡提供了最终用户设备(手持设备)与接入点/桥接器之间的接口[6].

Marvell 88W8686 无线网卡芯片集成了一片 ARM 兼容 DECPU,包含 SDIO 和 SPI 以确保与多种主机系统互用 DE 高速串行主机接口,以及先进 DE802.11 a/b/g RF 收发器,完全与蜂窝网络相容.该芯片兼容多种通信标准如 L3,TCP/IP,UMA 和 IMS.

1.2 系统构成

主控制器采用 S3C2440[5],主频高达 400 ~ 533 MHz,基于 ARM920T 内核(16/32-bit RISC CPU)的高性能 CPU,独立的 16 kB 指令和 16 kB 数据 cache,MMU 虚拟内存管理单元,使得程序运行以及数据存储更加高效,并可以支持 Wince,Linux 等多种主流操作系统,其集成了以下片上外设:LCD 控制器(支持 STN 和 TFT)、SPI 控制器、SDIO 控制器、USB 控制器、NAND FLASH Controller 等.

操作系统采用 Linux 2.6.28;Bootloader 采用 vivi;根文件系统采用 ramdisk.系统启动后挂载 yaffs2 文件系统.系统的硬件平台采用广州友善之臂公司的 mini 2440 开发板,外围接口包括 1 个 SPI 接口和一根 IO 外部中断线.硬件电路:mini 2440开发板通过 SPI0 连接 Marvell 88W8686;mini 2440 开发板 con4 接口图如图 1 所示.

收稿日期:2011 - 01 - 25
作者简介:盛李立(1985 -),男,湖北荆州人,硕士研究生 . 研究方向:嵌入式系统与无线网络.
指导教师:王 忠,教授,硕士,硕士研究生指导老师.研究方向:图像处理.

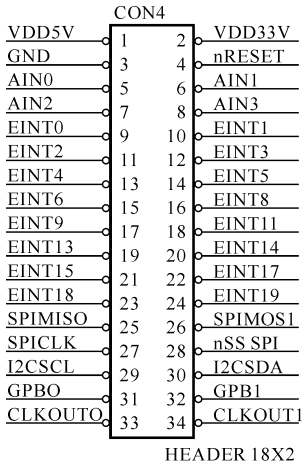

图 1 开发板 mini 2440 接口 con4 图

Fig 1 Con4 Interface of mini 2 440 Development Board

SPI 接口使用 SPI0，中断使用 EINT1，复位使用 nRESET，唤醒使用 GPB0，电源使用 VDD33V，地使用 GND.

无线路由器采用武汉大学网络与信息技术实验室开发的 WHU-VANET，支持 802.11 a/b/g 标准、TCP 协议和 TFTP 协议.

整个系统硬件框图如图 2 所示，图 3 为 Marvell 88W8686 硬件原理图.

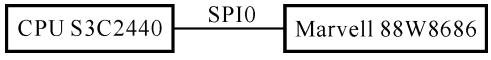

图 2 系统硬件框图

Fig 2 System Hardware Diagram

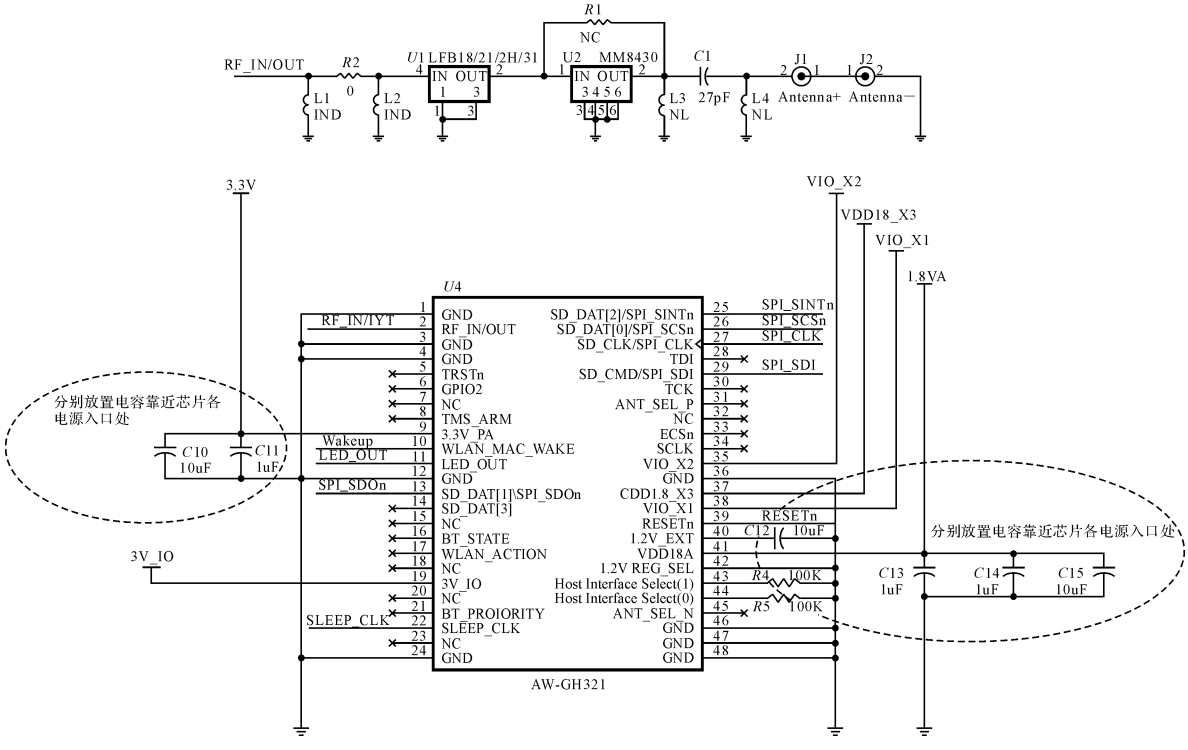

图 3 Marvell88W8686 硬件原理图

Fig 3 Marvell88W8686 Hardware Diagram

2 Marvell 88W8686 无线网卡芯片驱动

Marvell 88W8686 无线网卡芯片驱动实现的基本步骤如下：

- 基于 mini 2440 开发板的 Linux 2.6.28 操作系统的移植；
- 编写基于 mini 2440 开发板的 SPI 接口驱动；
- 编写 Marvell 88W8686 无线网卡芯片的驱动程序；
- 联合 WHU-VANET 无线路由器进行调试和性能测试.

2.1 Linux 2.6.28 操作系统的移植

2.1.1 下载安装交叉编译环境 在本文中，使用的交叉编译器是 arm-linux-gcc 4.3.2，其具体下载地址请参考文献[7]，下载后，进行解压安装到根目录.

2.1.2 获取 Linux 内核源代码 有很多方式可以获取 Linux 内核源代码，在本文中，是直接在 Fedora9 平台上互联网，直接在命令行通过 wget 命令获取到最原汁原味的 Linux-2.6.28 的源代码[8]，然后进行解压，接着针对 mini 2440 开发板进行对解压后的内核进行以下修改：

① 指定交叉编译变量

本文中，移植的目的是让 Linux-2.6.28.9 可以在 mini 2440 上运行.

首先，要使得 Linux-2.6.28.9 的缺省目标平台成为 ARM 的平台. 修改总目录下的 Makefile 中的 ARCH 指定平台为 ARM_CROSS_COMPILE 为解压安装后的 arm-linux-gcc 4.3.2 开发工具[2].

② 修改机器码

首先很关键的一点，内核在启动时，是通过 bootloader 传入的机器码(MACH_TYPE)确定应启动哪种目标平台的，友善之臂已经为 mini 2440 申请了自己的机器码为 1999，故需要修改 ARCH/arm/tools/mach_types 中的 ARCH_S3C2440 的机器码 362 为 1999，与 supervivi 传入的机器码参数一致即可！

③ 修改时钟源频率

修改 arch/arm/mach-s3c2440/mach-smdk2440.c 文件下的时钟源频率，因为 mini2440 开发板上的晶振是 12 MHz，故需要修改原 SMDK2440 目标板上的晶振 16.934 4 MHz 为 12 MHz.

④ 修改 Nand Flash 分区

系统默认的分区不是所需的，所以要自己修改，除此之外，还有 Nand Flash 的结构信息需要增加填写，以便能够适合系统自带的 Nand Flash 驱动接口.

⑤ 修改 DM9000 驱动程序

参考 Linux-2.6.18 内核中的 dm9000.c 源文件来修改 Linux-2.6.28 中的 DM9000 的驱动程序，主要是在 dm9000.c 的 dm9000_init()函数中添加 dm9000 寄存器初始化操作.

⑥ 获取 yaffs2 源代码

在命令行输入 # git clone git://www.aleph1.co.uk/yaffs2，可以下载到最新的 yaffs2 源代码. 然后解压为内核打上 yaffs2 补丁. 在命令行输入 #./patch-ker.sh c /opt/mini 2 440/linux-2.6.28.9 成功的为内核打上补丁.

⑦ 编译测试

在 Linux 源代码根目录下执行

#make s3c2410_defconfig

对内核进行默认配置，然后在命令行输入：

#make menuconfig

为内核添加对 DM9000 的支持，SPI 驱动的支持，同时添加上对 yaffs2 文件系统和无线扩展工具的支持.

⑧ 烧写到开发板运行测试

在命令行输入：#make zImage

生成内核映像文件，然后通过 wget 下载到 mini 2440 开发板上运行.

2.2 SPI 接口驱动编写

从 marvell 官方网站下载 SPI 接口的驱动程序 src_gspi8686.tar.gz，此驱动程序是基于 PXA270 的，所以需要移植到 mini 2440 平台上，主要有以下工作要做：

1)对 src_gspi8686 里 io 文件中的 gspi.c 和 gspi.h 这两个文件修改，针对各个具体函数，按照 s3c2440 SPI 的时序来编写 SPI 驱动，在本设计中，具体需要修改的部分如下：

① 首先定义 SPI 字符设备的在内核中的主设备号，代码如下：

static int major = 240;

② 内核提供了操作字符设备的函数接口，由 file_operations 结构封装[4].

驱动程序就是要实现这些只有函数名，而无函数体的函数接口，以便响应用户的系统调用，在这个结构中的每一个字段都必须指向驱动程序中实现特定操作的函数，对于不支持的操作，对应的字段可置为 NULL 值. 针对本设计中的 SPI 字符设备驱动，需要实现的函数接口是：gspihost_open()，gspihost_release()，其他函数接口置空.

• gspihost_open()函数完成以下操作：调用 try_module_get()函数来获得 SPI 驱动模块，同时将网卡的私有数据域置空.

• gspihost_release()函数完成以下操作：调用 module_put()函数来释放 SPI 驱动模块，同时将网卡的私有数据域置空.

③ 完成读写数据和寄存器操作，具体需要实现的函数如下：

• gspi_write_data_direct()根据 SPI 的写时序，首先使能 mini 2440 的 SPI0 时钟，然后片选中 Marvell 88W8686，根据 Marvell 88W8686 的数据手册，写操作的地址偏移量是 0x8000，然后将具体的数据通过 SPI 的 MOSI 管脚把数据发送给 Marvell 88W8686 芯片. 至此实现了 gspi_write_data_direct()的函数体.

• gspi_write_reg()和 gspi_write_data()函数的实现，均是调用 gspi_write_data_direct()函数，只是传递的参数有差别.

• gspi_read_data_direct()根据 SPI 的读时序，首先使能 SPI0 时钟，然后片选中 Marvell 88W8686，根据 Marvell 88W8686 的数据手册，将请求的操作的地址偏移量通过 mini 2440 的 SPI0 的 MOSI 发送给 Marvell 88W8686 芯片，然后 Marvell 88W8686 芯片将具体地址的数据通过

SPI0 的 MISO 管脚把数据上传给主机 mini 2440. 至此实现了 gspi_read_data_direct()的函数体.

• gspi_read_reg()和 gspi_read_data()函数的实现,均是调用 gspi_read_data_direct()函数,只是传递的参数有差别.

④ GSPI 模块在使用中断前要先请求一个中断通道(或者中断请求 IRQ),然后在使用后释放该中断. 在本系统中,mini 2440 的 SPI0 使用的是轮询发送中断接收,中断接收时,使用的是外部中断线 1,通过调用 request_irq()函数来完成 SPI0 中断的注册. 与此相反的是调用 gspi_unregister_irq()函数来注销中断.

⑤ 实现 gspihost_init_hw()函数,此函数主要完成以下操作:首先初始化 mini 2440 的 SPI0 的管脚;片选管脚初始化;设置 SPI0 的速率,模式,格式等;完成外部中断线 1 管脚即 EINT1 的初始化.

⑥ 实现 gspihost_module_init()函数,此函数主要完成以下操作:调用 ioremap()函数将 SPI0 和外部中断线 EINT1 所使用的 GPIO 的地址空间映射到内核的虚拟地址空间,便于系统访问;同时调用 register_chrdev()来向内核注册此 SPI 字符设备.

至此,基本上完成了 SPI 字符设备驱动程序的设计,待编译调试无误后既可下载安装此 SPI 模块到内核中去运行.

2)修改 Makefile 编译修改过的代码,编译生成 gspi.ko 和 gspi8xxx.ko.

2.3 Marvell 88W8686 驱动程序分析与理解

Marvell 88W8686 驱动程序调用流程如图 4 所示.

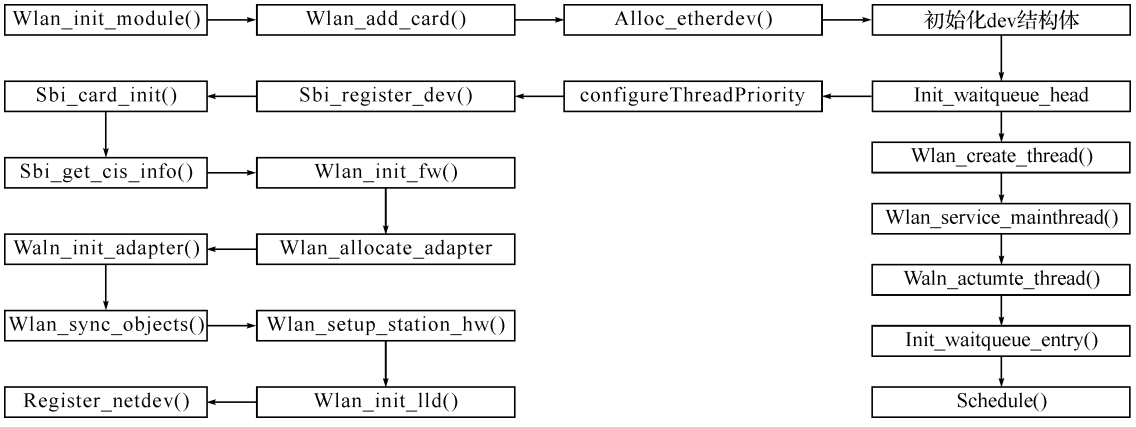

图 4 Marvell 驱动程序流程图

Fig 4 Marvell driver flow chart

在图 4 中:系统首先调用 Wlan_init_module()函数初始化 gspi8xxx.ko 模块,然后调用 Wlan_add_card()函数在系统内核中增加一个网卡设备,同时分配 wlan_priv 结构和初始化这个设备,再调用 Alloc_etherdev()函数分配一个以太网设备并且注册此以太网设备,再调用 inti_waitqueue_head()函数初始化等待队列,同时调用 wlan_create_thread()函数来创建 wlan_service_main_thread 主线程,并且调用 configureThreadPriority()函数来配置该线程的优先级. 其实 wlan_service_main_thread 主线程相当于一个中断服务程序,该线程处理由固件产生的接收事件或者接收的数据,并且发送从内核来的数据. 接着调用 sbi_register_dev()来注册此设备,根据网卡和中断获得的数据来填充此设备的私有数据结构. 调用 sbi_get_cis_info()函数来获得 CIS 表,接着调用 wlan_init_fw()函数来初始化固件,并且调用 register_netdev()函数来注册此网络设备. 至此,整个模块的初始化流程到此结束. 然后内核调用 schedule()函数来进行进程调度,当 wlan 驱动有中断时,就会进入 wlan_service_main_thread()函数进行判断,并且调用相应的函数进行处理.

3 编译与 ping 测试

(1)将编译生成的 gspi.ko 模块,Marvell 88W8686 无线网卡驱动模块 gspi8xxx.ko,helper.bin 及 gspi.bin 下载到 ARM 板中;

(2)修改文件执行权限,在命令行输入:chmod +x *;

(3)加载 spi 接口驱动 gspi.ko,在命令行输入:insmod gspi.ko;

(4)加载网卡设备驱动及上载芯片固件,在命令行输入:insmod gspi8xxx.ko helper_name=./helper.bin fw_name=./gspi.bin.

(5)加载驱动完成后,查看网卡信息,然后扫描可用的无线网络,再连接到特定的 AP,最终配置以太网信息并 ping 测试. 如图 5 所示:ping 测试

成功，说明系统运行正常，WLAN 通信正常.

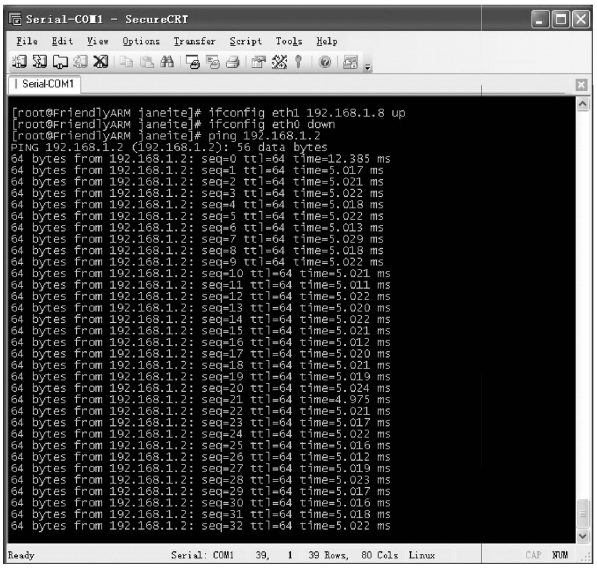

图 5　ping 结果图

Figure 5　Ping Results Diagram

4　测试结果与分析

4.1　测试方法

使用 UDP 方式 SOCKET 发送数据，模拟生命监护仪发送过程，数据发送量为 8 kBytes/s. 工作方式采用：连接 AP、发送睡眠时缓冲数据、发送唤醒时数据、断开连接、睡眠循环的工作方式，以达到低功耗的要求.

4.2　测试结果

4.2.1 实测数据　　进各阶段的占用时间通过 linux 系统自带时间函数求得，单位为 1 s，精确度为 0.01 s，测试结果如图 6 所示.

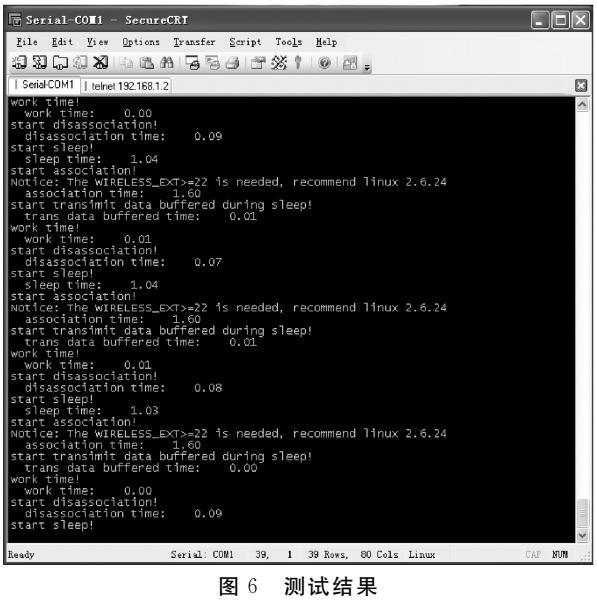

图 6　测试结果

Figure 6　Test Results Diagram

由图 6 可知，各阶段所耗时间为：

a）连接时间：1.60 s

b）休眠 1 s 时缓存的 8 kBytes 发送时间：0.01 s

c）工作时 16 kBytes 数据（假设为 2 s 内的数据量）发送时间：0.01 s

d）断开连接时间：0.08 s

e）休眠时间：1.04 s

4.2.2　技术指标计算　　根据以上测试时间及数据量等数据，可以计算以下指标：

① 工作平均功耗：3.3 V * (170 mA * (1.60 s + 0.01 s + 0.01 s + 0.08 s) + 1.8 mA * 1.04 s)/(1.60 s + 0.01 s + 0.01 s + 0.08 s + 1.04 s) = 3.3 V * 63.35 mA = 209.01 mW

② 休眠功耗：3.3 V * 1.8mA = 5.94 mW

③ 平均数据量：(8 kBytes + 16 kBytes)/(1.60 s + 0.01 s + 0.01 s + 0.08 s + 1.04 s) = 8.76 kBytes/s

④ 最大传输延时：1.04 s + 1.60 s + 0.01 s = 2.65 s

⑤ 发射功率最大是：18 dbm

⑥ 最大切换时间为，断开连接时间加上连接 AP 时间：0.08 s + 1.60 s = 1.68 s

通过以上计算，所有数据均满足生命监护仪和实际使用场合所规定的无线网卡技术指标.

5　结　语

WIFI 医疗是一个潜力巨大的市场，快速发展的市场，将有力地推动 WIFI 医疗设备的开发. 本文从工程应用的角度出发，研究并移植了 Linux 下的 Marvell 88W8686 无线网卡芯片的设备驱动，以此为基础可以构建诸如便携式的心电监护仪的无线传输系统，打破传统的床旁监测的单一模式，为各级医疗机构提高医疗服务水平提供了良好的平台.

参考文献：

[1]　王标，郭敏，单保慈. 基于 ARM 的无线网卡设备驱动设计[J]. 现代电子技术，2009(7)：101-103.

[2]　孙天泽，袁文菊. 嵌入式设计及 Linux 驱动开发指南——基于 ARM9 处理器[M]. 北京：电子工业出版社，2005：28-35，115-125.

[3]　刘峥嵘. 嵌入式 Linux 应用开发详解[M]. 北京：机械工业出版社，2005：22-35.

[4]　[美]Corbet J. Linux 设备驱动程序[M]. 3 版. 北京：中国电力出版社，2006.

[5]　Samsung. S3C2440A 32-BIT CMOS MICROCONTROLLER USER'S MANUAL Revision 1[EB/OL]. http://www. mcuoL. com/dounLoad/254/2179. htm, 2004.

（下转第 97 页）