ARM7 芯片 W90N740 以太网接口设计及驱动开发

张 东[1]，胡荣贵[2]，徐 海[1]
(1. 电子工程学院 研一队，安徽 合肥 230037；
2. 电子工程学院 网络系 602 教研室，安徽 合肥 230037)

摘 要：介绍了为内部集成 MAC 层控制器的 ARM7 芯片 W90N740 扩展网络接口的方法，网络接口的 PHY 层网卡芯片采用 Davicom 公司的 DM9161。包括硬件电路的连接网卡驱动程序的编写以及模块化加载驱动程序的方法。

关键词：ARM；μClinux；W90N740；网络接口；网络驱动
中图分类号：TP393 **文献标识码**：A **文章编号**：1674-7720(2010)10-0018-04

Design of ethernet interface for ARM7 chip W90N740

ZHANG Dong[1], HU Rong Gui[2], XU Hai[1]
(1. Company Yanyi, Electronic Engineering Institute, Hefei 230037, China;
2. Dept. of Network, Electronic Engineering Institute, Hefei 230037, China)

Abstact: The paper describes the method of expanding the network interface for ARM7 chip W90N740 which integrates MAC layer controller. Network interface card of the PHY-layer chip adopts Davicom's DM9161. The paper introduces hardware circuit connections, developments of network card driver, and the modular load-driven procedures.

Key words: ARM; μClinux; W90N740; NIC; network device driver

嵌入式 Internet 技术就是把 TCP/IP 协议应用到嵌入式设备中，从而使嵌入式设备实现接入 Internet 的功能，是嵌入式系统飞速发展和 Internet 触角不断延伸的必然结果。目前嵌入式 Internet 技术在工业控制、智能家电、消费电子等领域的应用越来越广泛。巨大的市场需求促进了嵌入式技术的飞速发展。网络化、小型化、专用化是嵌入式系统的发展方向，越来越多的应用场合需要将嵌入式系统接入网络。在这一背景下，本文设计实现了基于 ARM7 芯片 W90N740 的以太网接口，编写了 μClinux 下网络设备驱动程序，实现了将嵌入式系统接入网络。

1 接口的硬件设计

1.1 接口芯片

以太网接口控制器主要由 MAC(Media Access Layer)和 PHY (Physical Layer)2 部分组成。PHY 网络芯片与 MAC 控制器之间的连接通过介质无关接口 MII(Media Independent Interface)和简化介质无关接口 RMII(Reduced MII)实现。MAC 层又叫媒体访问层，主要负责逻辑控制且容易集成在处理器内部。如果处理器内部集成了 MAC 控制器，则只需要接入一片 PHY 网卡芯片，如 RTL8021、DM9161 等。否则必须接入包括 MAC 层和 PHY 层控制器的网络芯片，如 RTL8019、CS8900、DM9000 等。目前，很多针对网络控制应用的嵌入式处理器都集成了 MAC 层控制器。

W90N740 是华邦公司(Winbond)生产的 16/32 bit、基于 ARM7 架构的 RISC 芯片，专门为嵌入式及网络应用而设计，内部集成了 2 个 MAC 层控制器。可以用来开发宽带路由器、无线接入点、本地网关以及局域网视频传输等应用产品。本文选用 Davicom 公司的 DM9161 为其扩展网络接口。

DM9161 是单片 PHY 层收发器，在传输介质方面，为适用于 100BASE-TX 快速以太网的 UTP5 和适用于 10BASE-T 的 UTP5/UTP3 提供接口。DM9161 结构框图如图 1 所示。

1.2 硬件连接

硬件连接主要实现 MCU 与网卡芯片之间数据、地址、控制 3 大总线的连接，以及网卡芯片通过隔离变压器与 RJ45 水晶头的连接方法。

W90N740 与 DM9161 引脚连接示意图如图 2 所示。

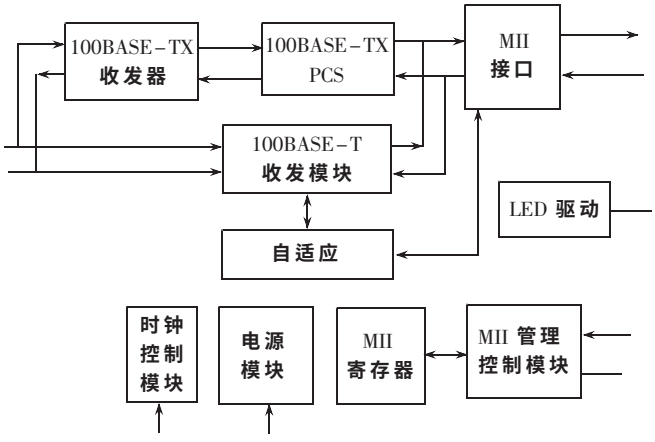

图 1 DM9161 芯片功能模块图

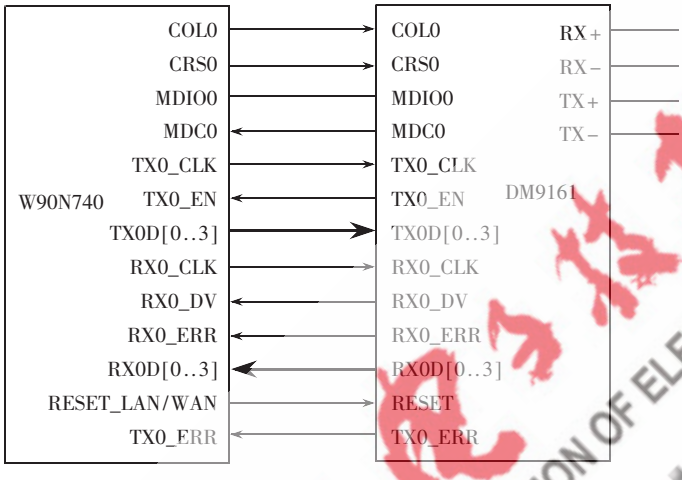

图 2 W90N740 与 DM9161 引脚连接示意图

以太网控制器 DM9161 的接收信号线 RX+，RX-和发送信号线 TX+，TX-连接到隔离变压器(Transformer)，通过隔离变压器连接到水晶头 RJ45。隔离变压器将嵌入式系统与以太网相隔离，以免相互干扰。

2 μClinux 下网卡驱动程序设计

2.1 Linux(μClinux)网络驱动程序的结构

Linux(μClinux)网络驱动程序包括了面向操作系统内核的接口和面向物理层的接口。面向操作系统内核的接口程序，用于发现、检测网卡参数及发送数据例程。面向物理层接口程序主要是中断处理例程。嵌入式网络驱动程序的结构如图 3 所示。

2.2 核心数据结构

网络驱动程序涉及到的核心数据结构主要有 struct net_device 和 struct sk_buff。

2.2.1 struct net_device

struct net_device 结构体是整个网络驱动结构的核心，用来表示 Linux 系统中的一个网络设备接口。其中定义了很多供网络协议接口层调用设备的标准方法，该结构在内核源码树<include/linux/netdevice.h>文件中定义，下面只列出其中主要的成员及其含义。

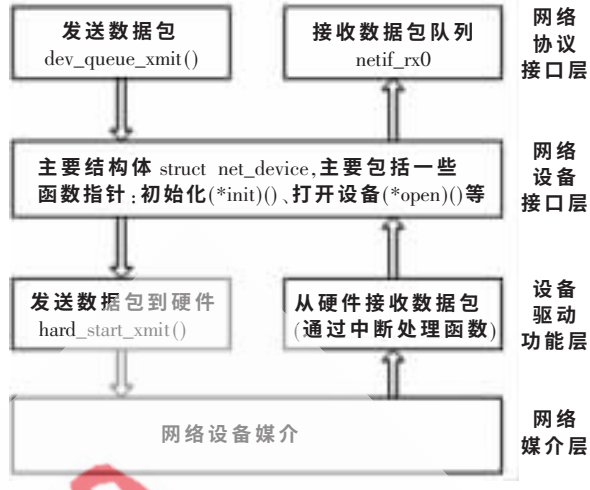

图 3 嵌入式 Linux 网络设备驱动程序结构

(1)全局信息及底层硬件信息

name：网络设备名称，默认是以太网；next：指向全局链表下一个设备的指针，驱动程序中不修改；men_start、mem_end：发送和接收缓冲区的起始、结束位置；base_addr：网络设备的 I/O 地址；irq：中断号，ifconflg 命令可显示和修改；hard_header_len：硬件头的长度，以太网中值为 14；mtu：最大传输单元，以太网中值为 1 500 B；dev_addr[MAX_ADDR_LEN]：硬件(MAC)地址长度及设备硬件地址，以太网地址长度是 48 bit。

(2)主要操作方法

int (*init) (struct net_device *dev)：完成设备初始化并向系统注册，仅调用一次；int(*open)(struct net_device* dev)：设备打开接口函数，当用 ifconfig 激活网络设备时被调用，注册所用的系统资源(I/O 端口、IRQ、DMA 等)，激活硬件并增加使用计数；int (*stop)(struct net_device*dev)：执行 open 方法的反操作；int(*hard_start_xmit) (struct sk_buff *skb，struct net_device *dev)：初始化数据传输；int(*hard_header) (struct sk_buff *skb，struct net_device *dev，unsigned short type，void *daddr，void *saddr，unsigned len)：该函数(在 hard_start_xmi 前被调用)根据之前检索到的源和目标硬件地址建立硬件头。

2.2.2 struct sk_buff

嵌入式 Linux 网络各层之间的数据传送都是通过 sk_buff 结构体来完成。sk_buff 提供一套管理数据缓冲区的方法，是嵌入式 Linux 网络系统高效运行的关键。sk_buff 组织成双向链表的形式。每个 sk_buff 包括一块数据缓冲区和一些控制方法。控制方法可以分成两类，一类控制整个 bufffer 链，一类控制该结构的缓冲区。

sk_buff 结构体的成员主要有：

```
struct sock *sk;                /* 结构体所属的 socket */
struct net_device *dev;         /* 收发数据的网络设备 */
```

```
union h;                    /* 传输层数据包头部 */
union nh;                   /* 网络层数据包头 */
union mac;                  /* 数据链路层数据包头 */
unsigned char *head;        /* 数据缓冲区头指针 */
unsigned char *tail;        /* 数据缓冲区尾指针 */
```

常用的对 sk_buff 结构体进行操作的系统调用主要有：struct sk_buff *alloc_skb (unsigned int size,int gfp_mask)，申请一块 sk_buff；static inline unsigned char *skb_put(struct sk_buff *skb, unsigned int len)，向 sk_buff 添加数据；static inline void kfree_skb(struct sk_buff *skb)，释放一个 skb。

2.2.3 struct w740_priv

为了便于对网络设备数据缓冲区进行管理，驱动程序中定义了一个数据结构 struct w740_priv，该结构体的主要内容是使用二维数组定义了该网络设备的发送、接收缓冲区，并且定义了接收、发送缓冲区描述符数组：

```
volatile RXBD  rx_desc[RX_DESC_SIZE];
volatile TXBD  tx_desc[TX_DESC_SIZE];
volatile char rx_buf[RX_DESC_SIZE][PACKET_BUFFER_SIZE];
volatile char tx_buf[TX_DESC_SIZE][PACKET_BUFFER_SIZE];
```

2.3 驱动程序的编写

网络设备驱动程序的编写，实际上就是为 struct net_device 结构中的函数指针编写实体函数的过程。具体需要由程序员来实现的函数包括：网络接口初始化函数 (*init)(struct net_device *dev)；设备打开和关闭函数 (*open) (struct net_device *dev)，(*stop) (struct net_device *dev)；发送函数 (*hard_start_xmit) (struct sk_buff *skb, struct net_device *dev)以及中断接收函数。

2.3.1 网络设备初始化

如果网络驱动静态编译操作系统内核，则当内核启动时自动调用初始化函数。如果驱动采用模块化加载方式添加到内核，则当运行 insmod 命令加载驱动时调用初始化函数。

编写初始化函数 w740_init(struct net_device *dev)，并将其赋值给网络设备接口的初始化函数指针(*init)。初始化函数 w740_init(struct net_device *dev)主要完成对网络设备接口 struct net_device *dev 的其他成员的初始化。首先调用系统函数 ether_setup (struct net_device *dev)为 dev 结构中的以太网相关项（如 dev->type、dev->mtu 等）赋值，然后将自己编写的实体函数关联到 dev 结构的对应函数指针。

```
dev->open=w740_open;
dev->stop=w740_close;
dev->do_ioctl=w740_do_ioctl;
dev->hard_start_xmit=w740_start_xmit;
……
```

2.3.2 网络接口设备的打开和关闭

网络设备的打开，就是激活网络接口，使它能够接收和发送网络数据。在设备打开函数 w740_open()中主要实现以下功能：

(1)初始化接收、发送缓冲区（通过 w740_priv 结构体）；

(2)设置相关功能寄存器（组），包括 DMA 寄存器组、cam 寄存器组、MAC 寄存器组、终端使能寄存器等，使网卡处于打开状态（具体需要设置的寄存器类型设置方法，可参考芯片手册）；

(3)调用系统函数 request_irq(unsigned int irq,void (*handler)(int irq,void *dev_id,struct pt_regs *regs),unsigned long irqflags,const char *devname,void *dev_id)，为接收中断和发送中断申请中断号并注册中断服务函数。参数 irq 是要申请的中断号，handler 是中断处理函数 irqflags 是中断处理的一些属性；

在 Linux 中，由于支持中断共享，允许不同的设备使用一个中断号，所以在使用 request_irq()申请中断时要指明是否采用中断共享的方式。但在嵌入式系统中，由于接口和功能单一，网络驱动程序中一般不采用中断共享的方式；中断号、I/O 地址也不进行探测，而是在设备打开时直接申请，在设备关闭时使用 free_irq(unsigned int irq,void *dev_id)释放中断。

设备的关闭函数 w740_close()完成 w740_open()的逆过程，不再赘述。

2.3.3 数据包的发送和接收

通过编写函数 static int w740_start_xmit(struct sk_buff *skb, struct net_device *dev) 和 int send_frame (struct net_device *dev ,unsigned char *data,int length) 实现数据包的发送。w740_start_xmit()函数首先完成一些与调试程序相关的打印输出，然后调用 send_frame()。send_frame()函数主要完成以下操作：

(1)调用 static TXBD *get_txbd(struct net_device *dev)函数检查发送数据缓冲区是否满足相应条件。若满足，返回缓冲区描述符，否则停止发送过程；

(2)调用 static void send_txbd(struct net_device * dev, TXBD* txbd,unsigned char *data,int length,int bCopy)完成具体发送过程：函数将内核网络数据缓冲区 skb 中的待发送数据复制到指定的地址空间；关中断；设置 MAC 命令寄存器，将‘发送打开位’置位，启动传送；设置 DMA Receive Frame Control 寄存器，通过 DMA 机制将数据发送到硬件。

当网卡接收到数据包时，会触发中断，通过中断服务程序实现数据包的接收。编写中断服务函数 rx_interrupt(int irq,void *dev_id,struct pt_regs regs)，该函数先获取中断状态寄存器的值，对当前中断状态进行判断，清除中断标志位。然后调用 netdev_rx(struct net_device *dev)

完成数据包的接收工作，netdev_rx()完成以下操作：

(1)调用系统函数 static inline struct sk_buff *dev_alloc_skb(unsigned int length)，为待接收数据分配一块缓冲区(struct sk_buff 结构体)；调用系统函数 skb_put(struct sk_buff *skb, int len)为 skb 分配 len 大小的缓存；调用系统函数 eth_copy_and_sum()将硬件接收到的数据拷贝到 skb 结构体的数据缓存中；

(2)调用系统函数 netif_rx(struct sk_buff *skb)将 skb 中的数据传递给网络协议的上层进行进一步处理。

3 驱动程序的加载

有 2 种方式可以将嵌入式 Linux 网络设备驱动加载到操作系统：一是直接将驱动程序编译进内核；二是通过模块加载的方式。本文采用模块化加载的方式，这种方式更为灵活。

为实现驱动程序的模块加载，需要编写 2 个函数：init_module()和 cleanup_module()。在 init_module()中调用系统函数 register_netdev(struct net_device *dev)将设备注册到操作系统；在 cleanup_module()中调用 unregister_netdev(struct net_device *dev)向系统注销网络设备。

在调用 register_netdev (struct net_device *dev) 时，dev 结构一般填写前面的 11 项，即到 init()，后面的可以暂时不初始化。最重要的是 name 指针和 init()方法。name 指针空(NULL)或者 name[0]为空格(space)，系统就把该设备作为以太网设备处理。以太网设备有统一的命名格式。一定要提供 init()方法，register_device()会调用这个方法来初始化网络设备。

然后调用宏 module_init (init_module) 和 module_exit (cleanup_module)将编写的 2 个函数与模块的加载进行关联。这样，当通过系统命令 insmod()来加载网络驱动模块时，系统会自动调用 init_module()对设备进行注册和初始化；当执行命令 rmmod()时系统调用 cleanup_module()进行设备的注销和驱动模块的卸载。

本文在阐述了嵌入式 Linux 网络设备驱动程序编写方法的同时，结合为 Winbond 公司的 W90N740 芯片设计网络接口并编写驱动程序的实际，详细介绍了嵌入式 Linux 网络驱动程序的实现原理和开发流程，对其他硬件平台的网络接口设计及驱动程序开发有一定的借鉴意义。

参考文献

[1] 毛德操,胡希明.Linux 内核源代码情景分析.浙江大学出版社,2001.

[2] 李俊.嵌入式 Linux 设备驱动开发详解.人民邮电出版社,2008.

[3] 蔡斌,万柳.基于 Linux 的网络设备驱动程序的机制分析[J]. 微计算机应用,2006,27(4).

[4] 李炳龙,陈性元,杜学绘.Linux 网络设备驱动程序分析与设计[J].计算机应用研究,2001,18(10).

[5] 赵洁,丁香乾.嵌入式 Linux 网络驱动程序的开发及实现原理[J].微计算机信息,2008,6-2:64-66.

(收稿日期：2009-12-20)

作者简介：

张东,男,1984 年生,硕士研究生在读,主要研究方向：嵌入式系统。

胡荣贵,男,1966 年生,博士,教授,主要研究方向：嵌入式系统、网络安全。

徐海,男,1984 年生,硕士研究生在读,主要研究方向：嵌入式系统。