基于 MPC8260 和 VxWorks 实现快速以太网通信

引言

摩托罗拉 MPC8260 微处理器芯片因其强大的通信处理能力和多种协议的支持而被广泛应用于通信和网络领域。本文以 MPC8260 为例，探讨在嵌入式系统中实现基于 VxWorks 操作系统的快速以太网通信的方法。

1 快速以太网通信接口的硬件实现

MPC8260 的 3 个 FCC(快速通信控制器)均可通过 MII(媒体独立接口)支持快速以太网，实现以太网中的 MAC(媒体访问控制)层功能。在设计中选用 FCCl 接口作为以太网接口。以太网 PHY(物理层)收发器选用 Intel 公司的 LXT971A，变压器选用裕泰公司的 YL18-2050S，采用 RJ45 标准物理接口。以太网口硬件设计如图 l 所示。

LXT971 A 支持 10BASE-TX、100BASE-TX / FX 等物理连接方式，拥有半双工、全双工、自适应等多种工作模式，拥有内嵌的 LED(发光二极管)驱动显示网口的工作状态，采用通用的 3.3 V 供电。在本设计中，IXT971A 通过 MII 与 MPC8260 的 FCC1 连接，MPC8260 通过 LXT971A 的串行控制管理(MDIO(管理数据输入输出)和 MDC(管理数据时钟))接口对 IXT971A 内部寄存器进行配置，实现工作模式的选择。但由于 LXT971A 是完全自适应的，因此，在实际的工作中不需要对 LXT971A 进行任何配置。需要注意的是，MPC8260 的引脚存在复用的问题，因此，软件必须将所涉及的引脚设为对应的 FCCl 通信口的功能。

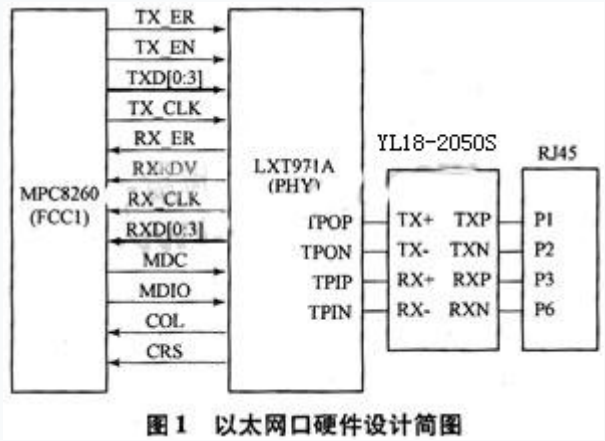

图 1 以太网口硬件设计简图

2 快速以太网驱动的设计与实现

2.1 VxWorks 网络系统结构

VxWorks 支持两种形式的网络驱动：一种是 BSD(Berkeley Software Distribution)驱动；另一种是 END(增强型网络驱动)是 VxWorks 所独有的，其主要特点是增加了一个 MUX 模块来管理 END 设备。目前比较通用的是采用 END 网络驱动程序。在本系统的网络驱动中选用 END 驱动作为以太网驱动方式，其网络系统结构如图 2 所示。

图 2　VxWorks 网络系统结构

收发数据时，网络协议层通过套接字(Socket)获取应用层提供的数据信息，经由 MUX(多路选择)层选择 END 设备。Mux 层将网络协议与网络设备的驱动区隔开来，使得网络协议不依赖于具体的硬件接口，而与硬件有关的代码都封装在 BSP 中，底层由 END 提供驱动。因此，在基于 VxWorks 的以太网通信实现的设计中，主要的工作是进行底层网络设备(硬件)设计和 END 驱动程序的开发。

2.2 END 驱动的实现

以太网在 VxWorks 下作为 END 设备，由通信处理器模块通过 FCC 接口进行管理。END 驱动的工作过程一般分为 3 个步骤完成，即初始化 END 设备、装载 END 设备以及启动 END 设备。风河公司提供了基于大多数嵌入式微处理器的评估板的 BSP 参考源程序，根据实际方案修改其中与网络通信相关的部分程序，即可实现 END 驱动在实际设计中的应用。

2.2.1 END 设备的初始化步骤

为了便于理解网络设备程序的装载过程，首先分析图 3 所示的在 VxWorks 映像启动时的网络初始化顺序。

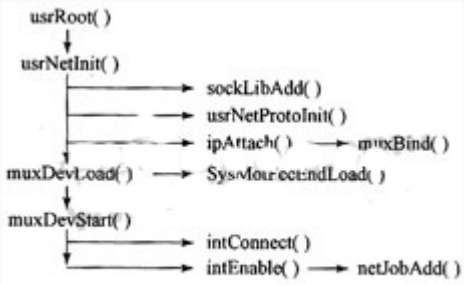

图3　网络初始化步骤

系统启动后，VxWorks 首先执行 usrRoot()函数，安装库程序和创建设备。接着 usrRoot()调用 usr-NetInit()函数完成网络的初始化，包括通过 muxDev-Load()调用 SysMotFccEndLoad()装载设备表 end-DevTbl[]中描述的设备以及调用 ipAttach()完成网络协议的安装等。通过这些过程完成网络装载并使其处于准备接收或发送数据状态。设备装载完成后，muxDevLoad()调用 muxDevStart()启动函数，该函数通过 intConnect()和 intEnable()实现中断服务的注册并打开中断宏。当有数据交换时，打开中断服务程序，调用排列网络任务处理函数 netJobAdd()，指派网络系统任务 tNetTask，由 tNetTask 处理数据接收和发送任务。

2.2.2 相关 BSP 的配置

在 VxWorks 中，网络设备驱动程序装载时，首先要通过对所处 BSP 中文件进行相应的配置才能实现。

1)接口引脚的定义和使能

以太网是由 MPC8260 的以太网控制器通过连接一个外部 PHY 芯片实现的，通过对 MPC8260 的可编程并行口进行寄存器设置，定义这些引脚为以太网接口信号。这部分功能在 sysLib.c 中实现。

另外，需要同时配置读写以太网 PHY 芯片内部寄存器的信号，这可以通过修改 sysFccEnetAddrGet、sysF-ccMiiWr、 sysFccMiiRd 等子过程来实现，详细的配置数据可以参照 MPC8260 并行口寄存器的配置表，这里不一一赘述。

2)以太网物理端口的定义

针对以太网接口，定义使用的 FCC 号、TBD 和 RBD 数目、PHY 物理地址和使用的通信速率操作模式以及用户标识等。这部分功能在 sysMotFccEnd.c 中实现。

```
#define MOT_FCC_NUM 0x1   //使用 FCC1
#define MOT_FCC_TBD_NUM 0x40   //TBD
#define MOT_FCC_RBD_NUM 0x20   //RBD
#define MOT_FCC_PHY_ADDR0 0x00   //PHY 地址
#define MOT_FCC_DEF_PHY_MODE PHY_100BASE_TX
    //使用 100BASE_TX 模式
```

3)添加驱动的调用人口

在 configNet.h 中定义 END 设备驱动程序入口表 END_TBL_ENTRY 结构的数组 end DevTbl[]，将驱动装载函数 SysMotFccEndLoad 的人口点及相关参数添加到 endDevTbl[]。

3 结束语

本文介绍了在 MPC8260 为核心的嵌入式系统中，基于 VxWorks 操作系统的快速以太网通信的硬件设计方法和驱动开发过程，该设计已在实际工作中顺利实现。