嵌入式系统上无操作系统 Telnet 服务器的实现①

胡 奕, 唐莉萍

(东华大学 信息科学与技术学院, 上海 201620)

摘 要: 提出一种在无操作系统的嵌入式系统上实现 Telnet 服务器的方法. 用户可以通过本地计算机的 Telnet 客户端程序登陆到目标嵌入式系统, 完成对嵌入式设备的远程调试与控制. 设计的 Telnet 服务器包含服务器模块和 shell 模块. 服务器模块负责处理客户端连接请求以及选项协商, shell 模块完成对客户端与服务器之间交互数据的处理.

关键词: Telnet; 嵌入式系统; 远程调试控制; 无操作系统

Implementation of Telnet Server for Non-OS Embedded System

HU Yi, TANG Li-Ping

(Communication and Information System, Donghua University, Shanghai 200040, China)

Abstract: An implementation of Telnet server for non-os embedded system is proposed in this paper. User is able to log in to the target embedded system through Telnet client program in local computer, and remote debugging and controlling can be realized. The designed Telnet server is divided into two modules: server module and shell module. Server module handles connection requests of clients and negotiates options, while shell module is responsible for processing the interactive data.

Key words: Telnet; embedded system; remote debugging and controlling; non-os

嵌入式系统的调试是用户在开发嵌入式系统流程中必不可少的环节. 对于远距离或多用户系统的调试, 远程调试是常用的调试方法. 远程调试是通过用户开发主机上的远程调试程序对目标机进行远程调试[1].

GDBServer 是一种常用的远程调试工具[2]. 其原理是在目标嵌入式系统上安装 GDBServer 并运行, 在开发主机上运行 GDB 调试程序, 开发主机通过网络远程登录到目标系统实现远程调试. 由于 GDB 的实现依赖于硬件结构[2], 对于资源有限的嵌入式系统会增加相当大的开销, 因此将 GDB 移植到其他开发板上使用有一定的难度. 文献[3]提出的一种间接访问目标嵌入式系统方法, 即开发主机在本地通过 Telnet 协议远程访问由串口与嵌入式系统直接相连的中间计算机. 这种方法需要设计串口数据与 Telnet 协议数据进行转换的算法, 并对中间计算机进行实时维护, 增加了系统整体的复杂性. 因此, 与在目标嵌入式系统上加装 GDBServer 相比较, 直接在目标机上实现 Telnet 服务器可以减少内存开销, 降低系统的复杂性.

Telnet 协议是 Internet 上广泛应用的一个远程登录协议, 和 TCP/IP 协议相兼容. Telnet 协议具有允许本地主机与访问本地服务器一样访问远程主机中资源的特点. Telnet 协议提供三种基本服务[4-6]: 网络虚拟终端 NVT(Network Visual Terminal)的认识、回显选项协商的机制和对称处理连接的两端, 即不强迫客户机从键盘输入也不强迫客户机在屏幕上显示输出. 其中较常用的服务是回显选项的协商[7,8]. Telnet 协议在数据采集[9], 远程控制[9], 图形界面设计[10]等中有着许多应用.

Telnet 服务器的实现需要遵循基本原则是: Telnet 协议位于应用层, 直接工作在 TCP 层之上, 远程主机服务程序工作在 TCP 的 23 号端口上[11]. 远程登陆服务分为四个过程, 分别是建立 TCP 连接, 客户端程序发

① 收稿时间:2014-02-26;收到修改稿时间:2014-04-08

送 NVT 格式数据, 解析远程主机的 NVT 格式数据到本地, 撤销连接[12,13]. 文献[14]提出一种在 8 位机上实现 Telnet 服务器的方法, 其设计使用精简的 TCP/IP 协议栈, 大大降低了内存使用. 对于功能单一的嵌入式系统, 不使用操作系统, 直接用协议栈控制驱动收发以太网数据包[14,15], 可以更有效地减少系统开销.

本文给出一种在无操作系统的嵌入式系统上实现 Telnet 服务器的方法, 以完成用户远程调试、控制嵌入式系统的功能. 设计的 Telnet 服务器分成两个模块, 服务器模块和 shell 模块. 服务器模块负责处理客户端连接请求以及选项协商, shell 模块完成对客户端与服务器之间交互数据的处理. 用户可以方便地通过 Telnet 客户端直接登录到远程嵌入式系统, 在本地计算机中查看和修改被调试的远程嵌入式目标系统上的内存单元、寄存器以及进程中的变量值等. 同时, 无操作系统也可以减少系统对内存的占用.

1 Telnet协议工作原理

Telnet 协议在 TCP/IP 协议之上, 属于应用层协议, 默认 TCP 端口号为 23.

Telnet 协议定义了数据和命令在网络上的传输方式, 即 NVT, 以适应异构环境. 一次完整的 Telnet 协议数据的交互过程为: 用户数据由本地键盘输入产生, 并被转换成 NVT 格式通过网络发送到远程服务器上. 服务器上伪终端入口点收到数据包之后将其转换成远程服务器规定的数据格式, 发送给上层服务应用程序进行处理. 最后将服务器处理完之后的结果转换成 NVT 格式送回伪终端入口点, 通过网络发送至客户机, 在用户屏幕上输出反馈信息.

Telnet 协议的选项协商机制规定当一方要执行某个选项时必须向另一端发出请求, 若另一端接受该选项, 则选项可以在两端同时起作用, 否则两端保持原来的模式. Telnet 协议中, 字节 0xFF(十进制的 255)叫做 IAC(interpret as command). 这是发送选项协商的第一个字节, 用 IAC 区分收到的字节是数据还是命令. 对于协商给定的选项, 连接的两端都可以发送下面 4 种请求的任意一个请求:

① WILL: 发送方本身将激活选项.

② DO: 发送方希望接收端激活选项.

③ WONT: 发送方本身希望禁止选项.

④ DONT: 发送方希望接收端禁止选项.

此为选项协商的第二个字节. 第三个字节即具体选项内容. 选项协商的原则是对于每个选项的处理都是对称的, 两端中任何一端都可以发出协商请求, 也可以接受或拒绝这个请求. 另外, 如果一端试图协商另一端不了解的选项, 接受请求的一端可简单地拒绝协商.

2 无操作系统Telnet服务器的实现

为了使得用户可以较方便地在本地计算机对嵌入式系统进行远程调试与控制, 本文设计了在嵌入式系统上实现无操作系统的 Telnet 服务器.

基于 Telnet 协议的标准, Telnet 服务器设计成两个独立的模块, 服务器模块与 shell 模块. 服务器模块负责处理来自 TCP 23 号端口的连接请求以及对需要的选项进行协商. Shell 模块进行交互数据的处理操作. 这样设计的优点在于, 远程调试与控制中具体的功能可以直接在 shell 模块中设计实现, 独立于服务器模块. 在基于 Telnet 协议标准的前提下, 可以自由独立地设计 shell 模块中的应用, 不影响 Telnet 协议的基本工作原理. 此外, 即使使用其他网络协议代替 Telnet 协议, shell 模块中的功能仍能正常运行, 具有一定的可扩展性.

2.1 系统设计思路

通常情况下, 在嵌入式系统中进行远程操作, 需要用操作系统进行多任务的管理、内存的分配、线程的切换以及其他一些复杂的功能. 操作系统的使用会占用嵌入式系统较多的内存空间和系统开销. 为了减少内存占用与系统开销, 必须设计一个不使用操作系统、单一任务结构即可支持必要工作的嵌入式系统. 在这个嵌入式系统中, 典型的框架是一个无限循环的结构, 主要功能包括检测中断与轮询设备等. 尽管系统中无操作系统存在, 为了实现以太网数据接收, 配置嵌入式设备的 GPIO(General Purpose Input Output)端口, 串口控制寄存器以及串口读写寄存器等操作, 驱动是必要的. 用户通过驱动自定义这些配置, 并在 Telnet 模块需要用到包含它们即可.

在无操作系统的嵌入式系统中, 将由协议栈控制驱动的读写功能、获取以太网数据包或者发送数据包到以太网中. 在无限循环结构中, 协议栈会询问驱动是否有新的数据包接收. 若有新的数据包到达, 则根据其是 IP 数据包或 ARP 数据包, 交给不同的数据通道

处理. 然后相关的事件被标记, 并将数据包交给上层应用. 若为端口 23 的 TCP 数据包, 则交给 Telnet 服务程序. 最后, 处理后的信息通过驱动发送出去.

2.2 服务器模块

本文设计的 Telnet 服务器具有无操作系统、嵌入式 TCP/IP 协议栈用轮询模式工作的特点. 基于这两个特点, 服务器模块设计需要从三个方面考虑. 第一, 在 TCP 的 23 号端口侦听连接请求. 在 23 号端口建立起正常的 TCP 连接时, 服务器模块进行初始化. 第二, 选项协商机制. 本文 Telnet 服务器的功能是进行远程控制与调试, 回显功能需要协商. 根据相关 RFC 文档的描述, 由 echo 与 SGA(Suppress Go Ahead)这两个选项协商确定哪一端进行回显功能. 因此, 在 TCP 连接建立成功之后, will echo, don't echo 以及 will SGA 这三个选项数据由服务器主动发送给客户端. 考虑到协议栈轮询模式的特点, 即无限循环等待接收新数据包, 主动选项协商不能在服务器的 TCP 状态转换成 ESTABLISHED(连接建立完成的状态)并开始等待接收新数据包时执行. 服务器只有在收到 SYN 包, 同时又收到正确的 ACK 包后执行选项协商操作. 此时双方的 TCP 连接已经建立成功, 选项协商的结果将实时记录. 第三, 对称地处理两端. 因此, 服务器的工作流程可以概括为以下的步骤:

① 侦听 TCP 的 23 号端口, 等待客户的连接请求;

② 接收到客户发送的连接请求(SYN 包), 回复 SYN 与 ACK 包, 将 TCP 状态转换成 SYN_RECEIVED(收到 SYN 包状态);

③ 接收到的正确的 ACK 包, 初始化 Telnet 服务器模块;

④ 发送回显选项协商;

⑤ 处理双方的选项协商并记录结果.

在此工作流程中, 需要解决两个问题. 第一, 收到系统不支持的选项协商问题. 在连接建立完成后, 一些客户端程序如 Linux 下的 Telnet 客户程序会主动发送多种类型的选项进行协商请求, 而本文的回显选项协商只需要 echo 和 SGA 两个选项类型. 对于收到其他类型的选项, 服务器首先判断选项内容是否为 echo 或者 SGA, 若不是, 则忽略其协商请求. 如果用户一次发送多个选项协商数据包, 不断查看下一个选项内容是否为所支持的, 直至数据包中所有选项内容检查完毕. 第二, 服务器的多用户连接问题. 这个问题可以路由表的同步建立来解决. 如果有多个用户同时远程登录, 对嵌入式系统进行控制和调试, 服务器必须分别处理每一个客户. 当一个连接请求到来时, 服务器首先查看路由表中是否有空记录, 如果存在空记录, 则向其添加用户的关键信息, 如 IP 地址、端口号等, 生成指定路由. 服务器发送数据时根据在路由表中查询到指定路由进行通信; 如果在路由表中没有查询到空记录, 则拒绝此连接请求, 并发送错误提示, 显示在客户屏幕上. 当客户连接关闭时, 路由表自动进行相应的更新操作, 生成一个空记录, 以便接受新的连接请求.

2.3 shell 模块

Shell 模块由 Telnet 服务器直接调用, 当 Telnet 服务器运行时, 用户通过 23 号端口发送的数据到达 TCP 层之后, 服务器模块将根据特定字节的内容进行不同的处理. 特定字节即 TCP 数据段部分的第一个字节, 如果该字节的内容为 IAC, 表明这是一个选项协商的数据包, 服务器模块会直接处理并记录协商信息, 并返回相应的信息给客户, 否则这个数据包交给 shell 模块进行操作.

Shell 模块主要有三个功能. 第一是回显用户通过键盘输入的可显字符. 将可显字符打印在用户的屏幕上, 以便用户检查输入的正确性. Shell 根据字符的 ASCII 码值区分可显字符, ASCII 码值的六进制数小于 0x1A 且不为 0x7F 的字符为可显字符. 第二是对不可显字符的处理. 不可显字符主要是键盘上的功能键, 如键盘上的回车、退格、home、end、delete、左/右键等控制类字符. 在 shell 中通过对部分功能键编程实现对输入字符进行的删除、插入等操作, 并将结果显示在客户屏幕上. 第三是解析用户指令. 在无操作系统的远程控制中, 我们自定义了部分系统指令. 当 shell 检测到用户输入完毕指令并按下回车键时, 检查当前一行字符是否系统自定义指令, 若是则调用对应的处理函数; 若不是则返回错误提示到用户屏幕上. shell 处理数据的流程如图 1 所示.

shell 模块作为一个独立模块, 由 Telnet 服务器直接调用, shell 本身并没有发送数据包的接口, 需要调用服务器模块将处理结果返回 Telnet 服务器, 由服务器发送数据到用户. 若上层用到 shell 模块, 只需向 shell 模块提供一个接口, shell 将处理完的数据送入指定接口, 这样处理可以降低系统的复杂性, 提高兼容

性.

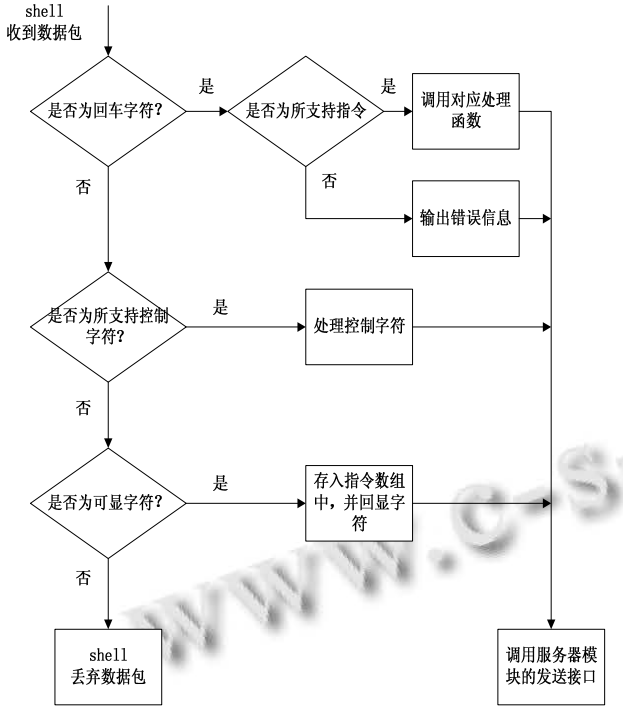

图 1　shell 模块处理数据包流程图

3　远程调试控制指令的设计

本文的远程调试控制主要功能是在本地在线读写内存中指定地址的数据, 比如查看或者设置嵌入式系统中的一些外围设备或者寄存器. 基于这个需求, 我们设计了读和写两种远程控制指令类型. 用户可以通过这两种指令对嵌入式系统内存中特定的地址进行读写, 从而远程调试控制嵌入式系统.

3.1 读指令

读指令形式为 xxxread b/h/w address, 由三个部分组成. 第一部分就是指令的关键字, 用于表明是在 RAM 中还是 ROM 中进行读操作, 如 ramread, 就是在 RAM 中读数据; 第二个参数决定读取数据的长度, b 表示字节, h 表示半字, w 表示字, 分别对应 8 位, 16 位与 32 位; 第三个参数 address 为用户指定的内存地址, 以十六进制形式表示. 在系统运行时, 用户可以通过读指令对已知的内存地址远程查看内容, 如 IP 地址, 外围设备工作状态等重要信息.

3.2 写指令

写指令用于修改某些特定内存的值. 写指令的形式为 xxxwrite b/h/w address data, 分别为功能关键字, 数据类型, 地址, 以及数据. 前三个部分的意义与作用和读指令类似, 第四个参数即用户想要写入的具体数值. 考虑到写指令的合法性, 用户输入写指令后, shell 需要检查指令的第二个参数, 这个参数规定了输入数据的长度, 若输入数据的长度大于规定的长度, 则会被截断, 若小于则自动补零.

4　嵌入式系统远程调试控制的实现

用户通过服务器的 IP 地址使用本地 Telnet 客户端向目标嵌入式系统发起 Telnet 连接, 目标嵌入式系统上的 Telnet 服务器在 23 号端口收到连接请求并处理, 该过程是 TCP 的三次握手连接. 当服务器检测到 23 号端口的 TCP 连接建立成功, 双方的交互数据便可在服务器的 23 号端口进行通信. 之后, 服务器执行选项协商操作, 发送 echo 与 SGA 选项协商数据包给客户端, 每一个选项内容的格式为 (IAC, will/won't/do/don't, option), 共三个字节. 在 Telnet 协议中, IAC, will, won't, do, don't 分别对应十六进制数 FF, FB, FC, FD, FE, 而 echo 与 SGA 选项对应于 0x01 与 0x03. 因此发送该 TCP 数据包时, 数据内容为 will echo, don't echo, will SGA, 如图 2 所示.

以太网头部	IP 头部	TCP 头部	FFFB01	FFFE01	FFFB03

图 2　服务器发送选项协商格式图

客户机收到服务器发出的选项协商包后, 对 echo 以及 SGA 的协商响应也通过发送一个或多个 TCP 数据包告知服务器, 即 do echo, won't echo, do SGA, 如图 3 所示(假设客户端在一个 TCP 数据包中响应了服务器的选项协商).

以太网头部	IP 头部	TCP 头部	FFFD01	FFFC01	FFFD03

图 3　客户端回应选项协商格式图

服务器在 23 号端口接收到客户机发送的选项协商数据包, 去除以太网、IP 以及 TCP 首部, 读取 TCP 数据段的第一个字节为 0xFF, 即 IAC, 标识紧接着两个字节为选项协商数据. 首先查看第三个字节确定选项内容, 其次查看第二个字节, 记录协商结果. 若处理完之后数据段还有剩余数据, 则作如上相同操作, 检测到不支持的选项内容则跳过不作记录.

通过对回显选项的协商，服务器与客户端确定了字符回显的方式，即服务器以一次一个字符的方式进行回显. 远程调试控制的准备工作也就完成了. 用户在本地 Telnet 客户端窗口输入的字符将会作为 TCP 数据包的数据部分发往服务器的 23 号端口，用户输入的字符数据以及服务器处理的结果数据都通过发送 TCP 数据包在 Telnet 协议的两端进行交互通信，其格式如图 4 所示.

以太网头部	IP 头部	TCP 头部	字符数据/处理结果

图 4 双方交互数据格式图

服务器接收到数据包之后，去除首部，仍然先判断数据段首字节是否为 IAC，若不是则表明此为数据包，调用 shell 进行处理. Shell 将进行对可显字符，控制字符以及指令的相应操作. 这样的交互过程实现了基于 Telnet 协议，用户可以在本地远程登录到目标嵌入式系统上进行在线调试控制的目的.

5 实验与总结

5.1 实验验证

为了验证本设计的可行性，本文基于 keil uVision 平台对 stm32f 开发板进行代码编写，包括 TCP/IP 的协议栈以及 Telnet 服务器程序. Keil 平台是基于 C 语言的软件开发系统，包括了 C 编译器、宏汇编、连接器、库管理和一个功能强大的仿真调试器等. 其中 Keil uVision 平台对 ARM 芯片有着良好的支持，因此对于以 ARM 为主控芯片的 stm32f 开发板，可以通过 keil uVision 平台进行软件开发.

在 Windows 与 Linux 下通过 Telnet 客户端程序与下位机进行网络通信，通过 Telnet 协议登陆至下位机 stm32f 开发板上进行远程调试控制. 用户可以通过 keil 软件生成的.map 文件查找到相应变量或寄存器的内存地址，通过读写指令实现了远程调试控制的功能. 如用户需要查看或者修改下位机的 IP 地址时，根据 keil 编译后生成的.map 文件中找到 IP 地址的内存地址，如图 5 所示.

因此下位机 IP 地址的内存地址为 0x20000880，用户可以在 Telnet 客户端中输入 ramread w 20000880 查看 IP 地址，或输入 ramwrite w 20000880 deccd39d 命令修改下位机 IP 地址为 0xdeccd39d(222.204.211.157)，如图 6 所示.

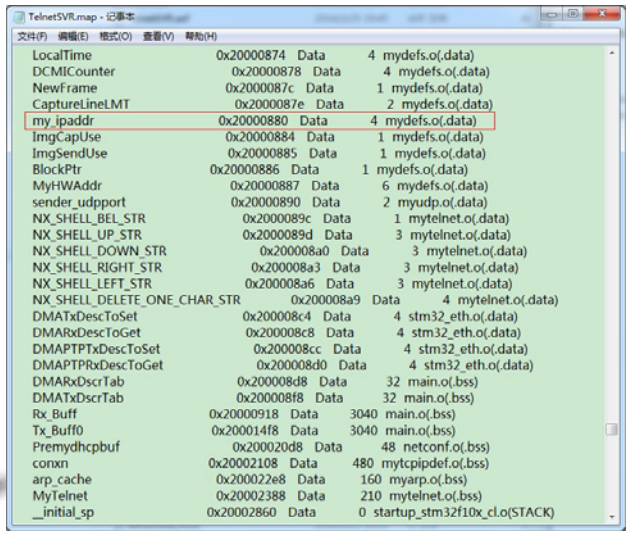

图 5 keil 生成的.map 文件图

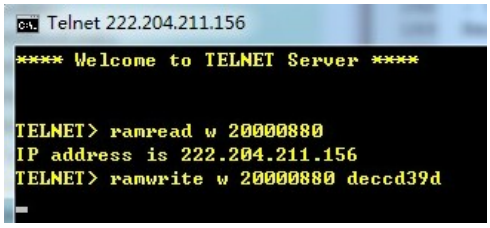

图 6 用户读/写 IP 地址命令图

由于改变了 IP 地址，当前 Telnet 客户端无法与下位机进行通信，因此重新开启新的 Telnet 客户端通过下位机新的 IP 地址完成远程登录，如图 7 所示.

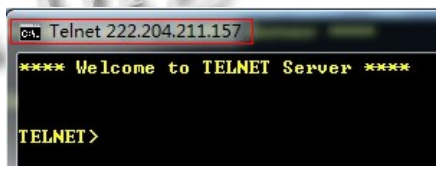

图 7 用户读/写 IP 地址命令图

从 Telnet 客户端窗口的第一栏可以看出，下位机的 IP 地址已经修改成 222.204.211.157. 由此验证了用户可以在本地 Telnet 客户端中远程查看或者修改下位机的 IP 地址.

5.2 总结

本文设计了一种在嵌入式系统上实现 Telnet 服务器的方法. 通过服务器模块和 shell 模块分别完成了 Telnet 协议与数据的处理. 通过直接在目标嵌入式系

统上实现 Telnet 服务器, 用户可以在本地实时地远程调试控制目标, 使得本地操作方便有效; 系统中没有安装外加的调试软件, 也不需要操作系统, 很好地降低了系统复杂性, 减少了整体开销; 在远程调试过程中, 用户无需关心内部代码, 增加了系统的易用性. 通过在 stm32f 开发板上实现了本文的设计, 验证了本方法对于嵌入式系统实现 Telnet 服务器具有可行性以及一定的普适性.

参考文献

1 龚兰兰,刘晓升,朱巧明.远程调试系统的关键技术分析.计算机应用与软件,2010,27(10).

2 郭胜超,吕强,杨季文,钱培德.GDB 远程调试及其在嵌入式 Linux 系统中的应用.计算机工程与科学,2004,26(10): 100–103.

3 吴明琪,马潮.一种软硬结合的嵌入式系统远程调试方法.单片机与嵌入式系统应用,2005(7):15–16,19.

4 Khare R. Telnet: The mother of all (application) protocols. Internet Computing, IEEE, 1998, 2(3): 88–91.

5 Postel J. Telnet protocol specification, request for comments0854, http://www.networksorcery.com/enp/default0702.htm. May, 1983.

6 Postel J. Telnet option specifications, request for comments0855,http://www.networksorcery.com/enp/default0702. htm. May, 1983

7 Postel J, Reynolds JK. Telnet echo option, request for Comments0857. http://www.networksorcery.com/ enp/default0702.htm. May, 1983.

8 张立平,杨尊钢.Telnet 在泵站网络监测中的应用.工业控制计算机,2013,26(5):63–64.

9 丰超,冯瑞芳.Telnet 数据采集方式的实现.商场现代化,2008,28(4):93.

10 许高建,王川林.基于 Telnet 客户端的图形界面设计.农业网络信息,2006(12):78–81.

11 卢爱卿,张会勇,赵征.Telnet 协议的实现原理及应用.计算机工程,2002,28(11):268-269,280.

12 王雷.详解 telnet.黑河科技,2003,(4):63–65.

13 温凤.基于 Telnet 协议的网络设备密码管理系统的设计与实现.四川通信技术,2011,(3):83–85.

14 叶勇杰,高仕斌,黄建敏.Telnet 在 8 位机上的实现.机电工程技术,2006,35(12):41–44.

15 汤宏萍,王竹平.嵌入式 TCP/IP 协议栈的设计与实现.微电子学与计算机,2008,25(6).