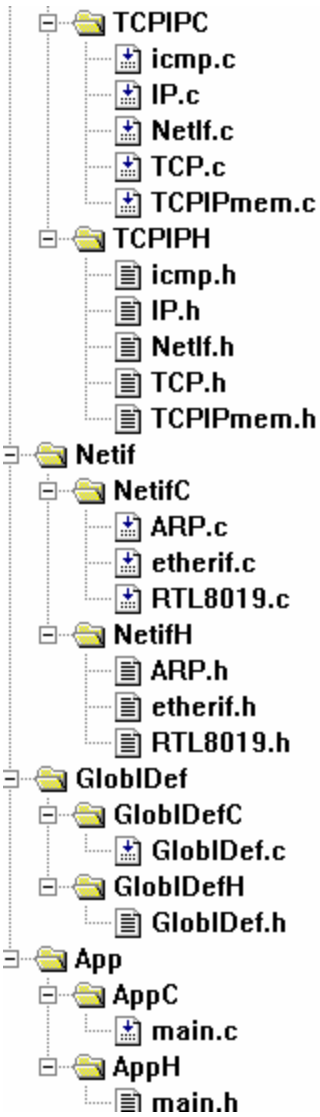
# ZLIP 使用简介

李章林[1]

（ 1 南开大学电子应用实验室，http://www.zlmcu.com，版本：2005-11-28）

## 1 目录结构




：TCP/IP协议栈程序所在目录。
：Icmp 协议。
：IP 层。
：网络接口层。
：TCP 协议层。
：TCPIP 内存管理程序。
：网络接口协议所在目录。
：ARP 协议。
：以太网接口协议。
：RTL8019AS 以太网接口芯片驱动程序。
：全局函数和宏定义所在目录
：应用层协议所在目录
：主程序，这里包含一个如何使用的例子程序。

KeilC 目录下是 KeilC51 的工程文件所在目录。用 KeilC51 打开 Ex1.Uv2。
MCU 目录下是各种类型的 51 单片机的头文件。

## 2 概述

单片机上网技术，是当前的一个热门技术。单片机上网技术中的一个重要部分是在单片上实现 TCP/IP 协议栈。现在可获得的 TCP/IP 源代码一般并不为 51 单片机设计，而 51 单片机和 KeilC51 编译器有其自身的特点：存储类型、函数指针、重入函数等，ZLIP 就是针对这些特点设计的 TCP/IP 协议栈。

ZLIP 设计的目标是：

1) 精简 TCP/IP 协议栈，以减小代码量。ZLIP 目前没有支持 UDP 协议，ICMP 协议也只支持其中的 echo 协议(响应 ping 数据包)。lwIP 是一个功能全面的 TCP/IP 协议栈，但是相对 51 来说代码量较大。

2) 应用层接口简单，以兼容通用的 socket 接口。uIP 有很小的代码量和减小代码量（选择 AVR 为目标器件时，代码为 5K 左右）和 RAM 使用量（100 字节左右）。uIP 采用了不保存需要应答的数据包的 RAM 使用方案，没有和 BSD 的套接字接口兼容，应用层接口较复杂。

3) 针对 KeilC51 编译器设计。所有的外部变量都使用了 xdata 类型，全部指针都为明确存储类型的指针，需要重入的函数已经声明为 reentant，使用 KeilC 的小模式下编译。

使用 12M 晶振、KeilC 编译器、89C55 单片下测试的技术参数如下：

表 1：技术参数

| 代码量（字节） | 外部 RAM 使用量(字节） | 发送速度（字节/秒） |
|---|---|---|
| 14841 | 11068 | 5.892K |

ZLIP 的特点如下：

1)有适中代码量和 RAM 使用量。

2)使用类似 MFC 的 CScoket 的套接字接口，使用方便。

3)支持多 TCP 连接、多网络设备。能方便地移植到多任务操作系统和其它 CPU 下。能方便地替换网络接口协议和网卡驱动设备。

4)支持 ping 命令的响应。

5)为单片机设计：所有的外部变量都使用了 xdata 类型，全部指针都为明确存储类型的指针，需要重入的函数已经声明为 reentant，使用 KeilC 的小模式编译。

# 3 电路图

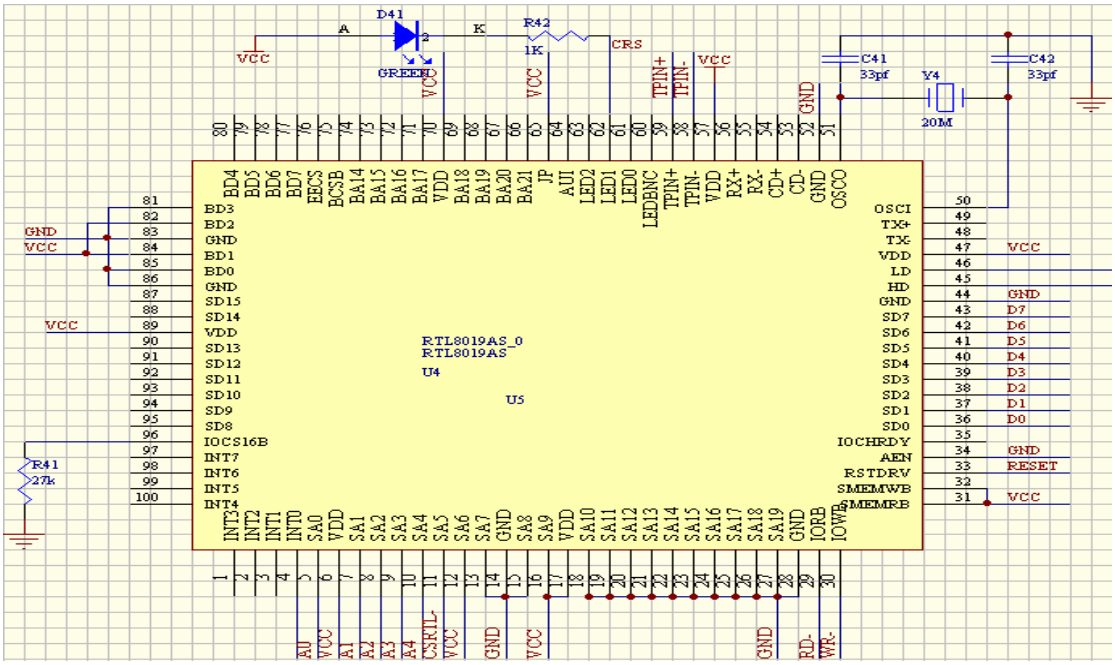



图 1：RTL8019AS 电路左半部分

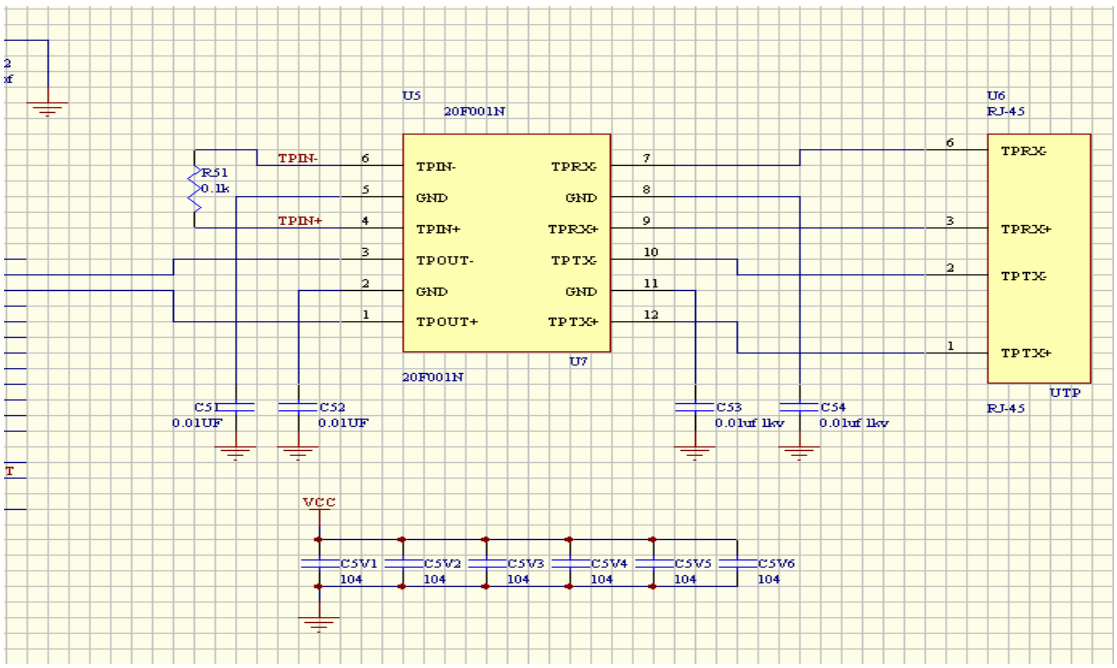



图 2：RTL8019AS 电路右半部分

该程序不能在 KeilC 下软件仿真，因为程序的运行需要外部电路配合。该 51 系统的外部电路主要有：以太网接口芯片 RTL8019AS 电路、外部 RAM 电路。

以太网接口芯片 RTL8019AS 电路图，如图 1 和图 2 表示。A0～A4 接地址线，D0～D7 接数据线，CSRTL 是片选线（低电平有效），RD-和 WR-接读写信号线。

# 4 应用层接口

zlIP 接口函数基本和 BSD 的套接字接口相同。

## 4．1 提供的用户接口函数：

TCPSocket()。

函数原型：socket xdata * TCPSocket(IP_ADDR ScrIP)。

功能：申请一个套接字。ScrIP 是这个套接字的本地 IP 地址。返回 socket 类型指针，如果申请失败返回 NULL。

TCPConnect()。

函数原型：BOOL TCPConnect(socket xdata * pTCB, IP_ADDR DestIP, WORD DestPort,void (code * recv)(void xdata * buf,WORD size),void (code * close)(socket xdata * pSocket))。

功能：向 IP 地址为 DestIP 的服务器的 DestPort 端口发起连接。参数 recv 和 close 用于设置当接收到数据包和对方要求关闭 TCP 连接时应该调用的回调函数指针。连接成功返回 TRUE，否则返回 FALSE。

TCPSend()。

函数原型：BOOL TCPSend(socket xdata * pTCB,void xdata *buf,WORD DataSize)。

功能：发送数据。发送数据的 TCP 连接是套接字指针 pTCB 对应的连接，发送的数据的起始地址为 buf，大小为 DataSize。发送成功返回 TRUE，否则返回 FALSE。

TCPSendEx()

函数原型：BOOL TCPSendEx(socket xdata * pTCB,struct SMemHead xdata *MemHead) 。

功能：快速发送数据。在使用 TCPSend 函数时，你首先需要将数据放入 buf 指向的内存中，然后调用 TCPSend 函数，接着该函数会将 buf 指向的内存区数据拷贝到 TCP 缓冲区中。使用 TCPSendEx 时你首先用 TCPAllocate(DATA_SIZE)获得一个 TCP 缓冲区，然后直接将数据放入 TCP 缓冲区中，从而比 TCPSend 函数少一次数据拷贝，提高发送速度。

参数：发送数据的 TCP 连接是套接字指针 pTCB 对应的连接，发送的数据放在 TCP 缓存 MemHead 中。发送成功返回 TRUE，否则返回 FALSE。

TCPListen()。

函数原型：BOOL TCPListen(socket xdata *pTCB,WORD ScrPort,void (code * accept)(socket xdata *pNewTCB)) 。

功能：使用套接字 pTCB 在 ScrPort 端口监听。参数 accept 是当有客户端向这个监听端口连接成功时调用的回调函数指针。

TCPClose()。

函数原型：void TCPClose(socket xdata *pTCB)。

功能：我方主动关闭连接时调用 TCPClose 函数，它将要求关闭套接字 pTCB 对应的连接。TCPClose 返回以后这个 TCP 连接可能保持，因为另一方还没有发起关闭请求。

TCPAbort()。

函数原型：void TCPAbort(socket xdata *pTCB)。

功能：当使用完这个套接字以后，调用 TCPAbort，将这个套接字释放，还给系统。

## 4．2 使用步骤

使用 ZLIP 时，在你的主程序中(请看示例程序的 main.c 文件)需要做的步骤如下：

1)首先设置一个 25ms 的定时中断函数（示例程序为 Timer 函数）。请在中断函数中调用 NetIfTimer(); ARPTimer(); TCPTimer();三个函数。

2)写 OnReceive 函数，它应该有如下的参数和返回值，函数名可以任意：

void OnReceive1(void DT_XDATA * buf,WORD size) REENTRANT_MUL

在使用 TCPConnect 函数时，OnReceive1 将作为 TCPConnect 函数的一个参数，也就是设置该 socket 的接收函数。当 TCP 连接接收到对方数据时，将自动调用 OnReceive1 函数。buf 指向接收的数据，size 是接收的数据量的大小。你可以在 OnReceive1 中处理接收的数据。当程序中有多个 TCP 连接同时存在时，你需要给每个连接准备一个 OnReceive 函数。

3)写 OnClose 函数，它应该有如下的参数和返回值，函数名可以任意：

void OnClose1(socket DT_XDATA * pSocket) REENTRANT_MUL

类似于 OnReceive 函数，当 TCP 连接的另一方首先向我方发起关闭连接的请求时，系

统将自动调用 OnClose 函数。pSocket 指向将要关闭的 socket。如果你想立即关闭这个连接则在 OnClose 函数中调用 TCPClose 函数。当程序中有多个 TCP 连接同时存在时，你需要给每个连接准备一个 OnClose 函数。

4)写 OnAccept 函数。如果你的程序中用到 TCPListen 函数监听某端口，这时需要写 OnAccept 函数。它应该有如下的参数和返回值，函数名可以任意：

void OnAccept1(socket DT_XDATA *pNewSocket) REENTRANT_MUL

当一个正在 listen 的 socket 接受了对方的连接以后将会自动调用该函数。pNewSocket 是将要获得这个连接的控制权的 socket 指针。一般在 OnAccept 函数中做以下处理：

```
ExAccept = pNewSocket;              //保存 pNewSocket，以后可以用 ExAccept 发送数据
pNewSocket->recv = OnAcceptRecv;   //设置 pNewSocket 的 OnReceive 函数。
pNewSocket->close = OnClose;        //设置 pNewSocket 的 OnClose 函数。
```

当程序中有多个处于 listen 的 socket 时，你需要给每个 socket 准备一个 OnAccept 函数

5)在主程序中做初始化工作：

```
/* init. the order is not important */
NetIfInit();      //初始化网络接口
ARPInit();      //初始化 ARP
TCPInit();      //初始化 TCP
MemInit();      //初始化内存模块
RTLInit(EtherAddr);      //初始化 RTL8019AS，EtherAddr 为以太网地址

/* init Devcie struct and init this device */
/* 初始化一个以太网接口设备，并设置这个设备的发送和接收驱动函数。如果你的系统中以太网接口芯片的驱动不一样，只要替换这里的发送和接口驱动函数就可以了*/
EtherDevInit(&DevRTL,EtherAddr,RTLSendPacket,RTLReceivePacket);

/* add this device to NetIf */
/* 添加一个网络接口设备。参数含义是：该设备的 IP 地址、子网掩码、网关、输入函数指针、输出函数指针、该设备的指针。如果你的系统中有多个网络设备，比如 moden，可以编写 moden 的输入输出函数，使用 NetIfAdd 函数添加这个设备。*/
NetIfAdd(IPAddr,NetMask,GateWay,EtherInput,EtherOutput,&DevRTL);
```

6)启动 25ms 的定时中断

7)使用类似

ExConn = TCPSocket(IPAddr);

语句分配一个 socket，并且绑定这个 socket 的源 IP 地址。

8)

如果我方作为服务器方，监听某一端口则：

TCPListen(ExConn,Port1,OnAccept1);

当另一方向我方 Port1 端口进行连接时，系统自动调用 OnAccpet1 函数。

如果我方作为客户端，向另一方的某个端口进行连接则：

TCPConnect(ExConn,IPAddr2,Port2,OnReceive2,OnClose2);

即向 IP 地址为 IPAddr2 的服务器的 Port2 端口进行连接。在连接成功以后，如果接收到另一方的数据则自动调用 OnReceive1 函数，如果接收到另一方的关闭请求则自动调用 OnClose1 函数。

9)当某个 socket 处于连接状态时，可以使用 TCPSend 或者 TCPSendEx 函数发送数据。

10)需要关闭连接的时候，使用 TCPClose 关闭连接。

11)当一个 socket 不再需要时，使用 TCPAbort 将这个 socket 还给系统。

# 5 移植相关修改

## 5.1 RTL8019AS 的基地址

修改 Netif\RTL8019.h 中的

```
#define RTL_BASE_ADDRESS 0xb000
```

默认的基地址为 0xb000。当单片机访问 0xb000 开始的地址的时候，CSRTL 信号线应给低电平，以选通 RTL8019AS。

## 5.2 TCP 缓冲区大小设置

修改 TCIPIP\TCPIPmem.h 中的

```
#define TCPIP_BUF_SIZE    0x2000
```

默认为 8K，建议大于 4K。缓冲区过小，将会影响发送和接收速度。

## 5.3 多网络设备

如果你的系统中有多个网络设备。修改 TCPIP\NetIf.h 中的

```
#define NET_IF_MAX_NUM      1
```

默认情况下为最多一个设备。

在主程序中使用 NetIfAdd 函数添加网络设备。

## 5.4 TCP 连接数的设置

修改 TCPIP\TCP.h 中的

```
#define TCP_CONNECTION_MAX_NUM        10
```

默认情况下最多支持 10 个 socket 同时工作。

## 5.5 网络接口层协议的最大帧头长度

只有当你的程序使用以太网以外的网络接口协议时，才需要修改。修改 TCPIP\NetIf.h 中的

```
#define NETIF_HEAD_MAX_LEN     14
```

默认是以太网帧头长度，即 14 个字节。

## 5.6 ARP 表的大小设置

修改 Netif\ARP.h 中的

```
#define ARP_ENTRY_MAX_NUM          4
```

默认情况下 ARP 表大小为 4 个记录。当 ARP 表已经满的时候，新的记录将会覆盖最老的那个记录。

## 5.7 响应 ping 命令

如果不希望系统能够响应 ping 命令，则修改 TCPIP\icmp.h 中的

```
#define ICMP_EN  1
```

默认情况下该开关是打开的。如果不需要此功能将其设置为 0

## 5.8 到其它 CPU 的移植

zlIP 虽然为 51 单片机设计，但是也可以被移植到其它的 CPU 上。系统中的 GloblDef\GlobleDef.h 记录了 CPU 的信息，主要修改这个文件。

1) 设置 BYTE,WORD,DWORD,BOOL 等类型的定义
2) 注释掉#define MCU_C51 这一行。注释掉这个选项开关以后将程序从 C51 变为 ANSIC，程序中将没有 C51 特有的关键字。
3) 字节顺序设置。即设置多字节变量的高字节存在于低地址还是高地址。51 单片机的字节顺序和 0x8086CPU 不一样。删除# define HOST_ORDER_AS_NET，如果字节顺序和网络字节顺序不一样。
4) 是否移植到具有多线程的 51 单片机程序中。比如单片上运行了 RTOS51、uc/OS-II、Tiny51 等单片上的多线程操作系统，则需要打开# define MULTI_THREAD 开关，此时程序中几乎所有的函数都声明为 reentrant 类型的。
5) 如果需要运行在调试状态打开# define DEBUG 开关。
6) 对于 IO 和 RAM 不是统一编址的系统需要修改 RTL8019.c 文件中的#define ReadReg(port) (*((BYTE DT_XDATA *)port)) 和 #define WriteReg(port,value) (*((BYTE DT_XDATA *)port) = value)，使程序能够访问 IO 端口。

# 6 后记

公布此源代码，旨在将我的心得和成果和大家共享，共同学习和进步。由于本人水平有限，错误和疏漏之处难免，还请各位同行指正。

## 参考文献

[1] 李章林，张立民. TCP/IP 在 51 单片上的实现特点和方法. “2003 年全国单片机和嵌入式系统年会”论文集.2003 年

[2] (电子文献)Adam Dunkels.uIP - A Free Small TCP/IP Stack[Z]. http://dunkels.com/adam/uip/index.html. 2002-1-15.1

[3] (电子文献)Adam Dunkels.lwIP - News Archive[Z].http://www.sics.se/~adam/lwip/news.html.2001-1-9.

[4] 李章林，张立民. ANSIC 程序到 KeilC51 的移植心得. “2003 年全国单片机和嵌入式系统年会”论文集.2003 年