Emtronix


# ETR232i 嵌入式网络模块编程手册

感谢您购买英创信息技术有限公司的产品：**ETR232i 嵌入式网络模块。**

**由于 ETR232H 和 ETR232i 在软件操作方面完全兼容，因此本手册也完全适用于 ETR232H。**

ETR232i 是一款以 R1610C 为核心、以网络数据通讯为特色的嵌入式 PC 模块，其外形尺寸仅为 74mm×53mm；配有 Flash、串口、以太网接口、GPIO、精简 ISA 总线、实时时钟、LCD 接口、矩阵键盘接口等板载资源；采用 BC3.1 作为开发调试工具；支持 RS232/RS485 数据通讯、常规 TCP/IP 应用、GPRS/CDMA 远程数据通讯、NAT 路由、无线网关、FTP 服务器、Web 服务器等多种应用；可用于通讯管理、工业控制、GPRS/CDMA 数据终端、仪器仪表等众多领域。

本手册主要是对 ETR232i 的驱动程序及相应的测试程序作详细的介绍，其相关的软件程序代码在所提供的应用光盘<software>目录中，按照 ETR232i 的不同应用，分为以下几个子目录：

| | |
|---|---|
| ..\\startup | 程序起步的 Demo 例程 |
| ..\\drivers | ETR232i 相关的硬件接口函数及相应的测试例程 |
| ..\\UART | ETR232i 串口驱动程序及相应的测试例程 |
| ..\\LCD | ETR232i LCD 汉字、图形显示驱动程序及相应的测试例程 |
| ..\\Ethernet | 基于以太网的 TCP/IP 协议库 API 接口函数及相应的测试例程 |
| ..\\PPPnet | 基于 PPP 通讯的 TCP/IP 协议库 API 接口函数及相应的测试例程（主要是针对 GPRS 的应用测试例程） |

本手册将以上述几部分驱动程序为单元逐一介绍具体内容。

随着用户需求的不断发展，我们提供的驱动程序也在进行相应的更新调整，用户可访问英创公司网站或直接与英创公司联系，以及时获得 ETR232i 的最新驱动程序。

此外，英创公司针对模块的使用编写有《ETR232i 嵌入式网络模块数据手册》；针对开发评估底板的使用编写有《ETR232i 嵌入式网络模块开发评估底板手册》。这三个手册都包含在英创为用户提供的产品开发光盘里面，用户也可以登录英创公司的网站下载阅读。

用户还可以访问英创公司网站或直接与英创公司联系以获得 ETR232i 的其他相关资料。

英创信息技术有限公司联系方式如下：

地址：成都市高新区高朋大道 5 号博士创业园 B 座 402# 邮编：610041

联系电话：028-86180660 传真：028-85141028

网址：http://www.emtronix.com 电子邮件：support@emtronix.com.cn

# 目　　录

# 1、C 语言入门

本节主要是为那些初次接触 C 语言或对 BC 开发环境不太熟悉的用户提供的，如果你对此已有一定的了解，可跳过本节。

本节中说明的测试程序，在英创公司提供的光盘中<startup>子目录下。

## 1.1 第一个应用程序

作为应用程序设计的第一步，我们将使用 BC31 集成开发环境，编写一个名为 HELLO.CPP 的源程序并把它编译连接成可执行文件 HELLO.EXE：

- 将你的开发调试主机系统置于 DOS 提示符并进入你的工作目录下。
- 键入命令 BC，使系统进入 BC 集成开发环境，按 F3 打开一个名为 HELLO.CPP 的文件。这时，系统进入 BC 编辑环境。无论用户是否使用 BC 的 C++ 特性，我们建议用户的文件名都以 CPP 作为扩展名，这有利于程序的调试。
- 通过键盘输入程序如下：

```
#include    <stdio.h>
#include    <conio.h>
#include    <time.h>
#include    <dos.h>

int main( )
{
    struct   time t;

    clrscr( );                    //清屏
    printf( "Hello!\n" );
    for( ; ; )
        {
        gettime(&t);
        printf(" %2d:%02d:%02d\r", t.ti_hour, t.ti_min, t.ti_sec);
```

```
        if( kbhit() )       break;
        }
    return 0;
}
```

- 按 F2 键将程序存盘，再按 F9 对程序进行编译连接，生成 EXE 可执行文件。
- 按 ALT+X 退出 BC 集成开发环境。
- 在 DOS 提示符下，键入：td –rp hello 程序将被自动下载至 ETR232i 下，并弹出源码调试窗口。
- 按 F9 键，程序将运行。如果接有 LCD 屏，可在 LCD 屏上看到“Hello！”及动态刷新的当前时间串。
- 如在 ETR232i 上接有键盘，按任意键程序将退出。在任意时候，可在 PC 上按 CTRL+BREAK 强制中断程序运行。按 ALT+X 将退出 TD 调试状态。

## 1.2 建立工程文件

建立工程文件可把多个程序模块方便地组合在一起进行编译连接，是设计专业程序的基本手段，这主要是因为通过建立工程文件可合理安排程序结构，快速调试程序错误，所以我们建议用户采用工程文件这一工具来设计自己的应用程序。

下面我们将建立一个工程文件 MYPRJ.PRJ，具有.PRJ 的扩展名。

- 将你的开发调试主机系统置于 DOS 提示符下。
- 键入命令 BC，使系统进入 BC 集成开发环境，按 ALT+P 打开一个名为 MYPRJ.PRJ 的工程文件。这时按 Insert 键，向工程文件中添加 TIMESTR.CPP 及 HELO.CPP。
- 模块 TIMESTR.CPP 定义了一个获取时间串的函数。
- 模块 HELO.CPP 包含了主函数 main( )，通过#include “timestr.h”，可以调用 TIMESTR.CPP 模块中提供的获取时间串的函数。
- 按 F2 键将程序存盘，再按 F9 对程序进行编译，自动对工程文件中两个源文件分别进行编译，然后连接，生成 MYPRJ.EXE 可执行文件。
- 利用 TD，可以将 MYPRJ.EXE 下载到 ETR232i 下进行调试运行。

## 1.3 文件读写

利用操作系统提供的文件系统设计应用程序是嵌入式系统有别于一般单片机系统的一个显著特点。任何一个文件在使用之前和使用之后，必须要进行打开和关闭，这是因为操作系统对于同时打开的文件数目是有限制的。因此在使用文件前应打开文件，才可对其中的信息进行存取。用完之后需要关闭，否则将会出现一些意想不到的错误。BC 提供了打开和关闭文件的函数：fopen()函数和 fclose()函数。在文件打开后，可根据需要调用相应的文件读写函数 fread()函数和 fwrite()函数。

我们提供的文件操作的例子程序，FILEIO.PRJ 包含了 TIMESTR.CPP 和 FILEIO.CPP。模块 TIMESTR.CPP 定义了一个获取时间串的函数。模块 FILEIO.CPP 包含了主函数 main( )，实现了显示当前时间，并将当前时间转换成文本格式存盘。在 FILEIO.CPP 定义了一个存盘函数，代码主体如下：

```
int SaveFile( char* FileName, char *str )
{
    FILE *fp;
    fp = fopen(FileName, "wt" );                    // 打开文件，格式为可写的文本文件
    if( fp==NULL )
        {
        printf( "Cannot Open this File!\n" );
        return -1;
        }
    fprintf( fp,"%s", str );                        // 向文件中写入串
    fclose( fp );                                   // 关闭文件
    return 0;
}
```

# 2、硬件接口驱动程序

与本节相关的程序在光盘中的<drivers>目录下。

## 2.1 硬件设置的接口函数定义

ETR232i 提供了可对低层硬件操作的接口函数，其定义在 ETR232i.H 中，ETR232i.CPP 是对 ETR232i 提供的低层操作 C/C++接口函数的实现，各个函数的定义如下：

(1) unsigned int GPIODirCfg( unsigned char Dir )

函数功能描述：设置 8 位通用数字 IO(从 D[0]位到 D[7]位)的输入输出方向。D[i]=0: 定义为输出；D[i]=1:定义为输入。D[0]、D[1] …… D[7]缺省定义为输入。

输入参数：Dir 中为 1 的位，表示对应的 GPIO 为输入；否则为输出。

(2) void GPIOWrite( unsigned int Value )

函数功能描述：设置 GPIO 的输出位电平。

输入参数：Value 为输出值。只有方向设置为输出的，才是有效的。

(3) unsigned char GPIORead( )

函数功能描述：读取 GPIO 的输入状态。

输入参数：无

返回值：返回字节中，只有方向设置为输入的，才是有效的。

(4) unsigned int GPIOSetBit( int BitIdx，int L10 )

函数功能描述：对通用的数字 IO 进行一位输出值设置。

输入参数：

int BitIdx 选择需设置的输出位的序号，其范围是从 0 … 9。

int L10 设置的输出值，0 或 1。

(5) void GPRS_STB( int L10 )

函数功能描述：设置 GPRS_STB 位，GPRS_STB 为一位数字输出，可用于启动

GPRS 模块。

另外如果没有 GPRS 应用，GPRS_STB 信号可作为通用数字输入输出来使用。

输入参数：

int L10　　设置的输出值，0 或 1。

(6) unsigned int GPRS_PWR( )

函数功能描述：读取 GPRS_PWR 位状态，GPRS_PWR 为数字输入，可用于检查 GPRS 的上电状态。

另外如果没有 GPRS 应用，GPRS_PWR 信号可以作为通用数字输入来使用。

输入参数：无

返回值：

0　GPRS 没有上电

1　GPRS 已经上电

(7) void GPRS_AUX( int L10 )

函数功能描述：设置 GPRS_AUX 位，GPRS_STB 为一位数字输出，可用于对 GPRS 模块电源进行管理，L10 为 0 开通 GPRS 电源，L10 为 1 关闭 GPRS 电源。

另外如果没有 GPRS 应用，GPRS_AUX 信号可作为通用数字输出来使用。

输入参数：

int L10　　设置的输出值，0 或 1。

(8) unsigned int NB_Delay( unsigned int milliseconds )

函数功能描述：延时 milliseconds 毫秒。可用于替代 C 函数 delay(…)。注意 ETR232i 的定时器结构与标准 PC 不一致，不能使用 delay()来进行延时操作。

(9) unsigned long NB_clock( unsigned int milliseconds )

函数功能描述：返回系统当前的 Tick 值。可用于替代 C 函数 clock(…)。

(10) int SetTMR1( unsigned int Divider )

函数功能描述：设置 Timer1 时钟的分频数。ETR232i 的 Timer1 输入时钟频率为 1MHz。Timer1 为占空比为 50%的方波，Divider 代表了分频数。需要注意的是：输出的方波信号一个周期会定时中断两次（上升沿、下降沿），Divider 分频数和实际产生波形的周期有一个 2 倍的关系。如：Divider = 20000，则产生周期为 40ms 的方波。

(11) int SetWDT( )

函数功能描述：打开 Watchdog，时间间隔约为 1.3 秒。注意：由于 Watchdog 的时间较小，所以在使用的过程中，要进行适时加载。

(12) int ReloadWDT( )

函数功能描述：加载 Watchdog

(13) int SysReboot( )

函数功能描述：重新启动系统，相当于 PC 机的热启动。

## 2.2 硬件接口函数应用例程

### 2.2.1 Watchdog 的使用

用于测试 WatchDog 函数，WDT.PRJ 由 ETR232i.CPP 和 WDT.CPP 构成，ETR232i.CPP 提供低层操作的 C/C++接口函数，WDT.CPP 包含了主函数 main（），先调用 EnableWDT( )函数启动 WatchDog，时间间隔为 1.3 秒，在循环中不断调用 ReloadWDT( ) 重载 watchdog，如果将循环中的延时时间调整超过 1.3 秒，系统将被重新启动。

### 2.2.2 GPIO 操作

**（1）通用的数字 IO**

实现对 8 位通用数字 IO 输入输出操作。GPIO.PRJ 由 ETR232i.CPP 和 GPIO.CPP 构成，ETR232i.CPP 提供低层操作的 C/C++接口函数，GPIO.CPP 包含了主函数 main（），在 main()中通过调用低层的操作函数，首先把 GPIO 设置为输出，然后设置不同的输出，可通过 ETR232i 评估板上的 LED 来观察 GPIO 输出的相应变化。

例如：

```
GPIODirCFG( 0 );          //设置 8 位数字 IO 的方向为输出
GPIOWrite( 0xff );        //设置 GPIO[7..0]=0xff,将 8 位输出为高
NB_Delay（1000）;         //延时 1 秒
GPIOWrite( 0 );           //设置 GPIO[7..0]=0x00, 将 8 位输出为低
```

**（2）利用 GPIO 接矩阵键盘**

在英创公司提供的评估底板中，利用 8 位 GPIO 可支持接 4×5 的矩阵键盘，其中 GPIO1、GPIO3、GPIO5、GPIO7 作为键盘的输入，GPIO0、GPIO2、GPIO4 编码输出 7 种状态，在我们提供的评估底板只接了 5 位输出，所以接 4×5 的矩阵键盘，驱动程序是可支持接 4×7 的矩阵键盘。键盘的驱动程序定义在 PC_KEY.CPP 中，PC_KEY.CPP 模块利用 C++的构造函数对使用的几位 GPIO 进行初始化操作以及相关中断的安装，所以用户在使用时只需要将 PC_KEY.CPP 包含在 PRJ 文件中，即可直接调用 C 函数 bioskey( )、 getch( )等进行键盘的操作。相应的测试程序请参见 KEYTST0.PRJ。

**（3）利用 GPIO 作实时时钟扩展**

ETR232I V8 及更早版本的主板上没有时钟芯片，可以利用 GPIO 进行时钟扩展。而在目前提供的 ETR232I V11 版本中，主板上已经有了时钟芯片，就没有必要再用 GPIO 来进行时钟扩展了，可直接调用系统函数 settime( ) setdate( ) gettime( ) getdate( )对实时时钟进行设置和读取的操作。

### 2.2.3 中断程序

ETR232i 提供三类可调用的中断资源：1）一类是可直接调用的 DOS 和 BIOS 的软中断；2）ETR232i 的精简 ISA 扩展总线中共有 2 个中断资源，可用于支持用户扩展的专用硬件单元；3）ETR232i 单元定时器 TMR1 主要是用于实现 1 毫秒至 50 毫秒间隔的定时中断。软中断功能一般是通过调用标准 C 运行库函数来使用的，这里不再赘述。本节主要介绍用户可直接操作的 ETR232i 硬件中断（包括定时器中断）如下表所示：

| 中断名称 | 中断号 | 中断优先级 | 备注 |
|---|---|---|---|
| IRQ5 | 0x0D | 5 | 扩展总线外部中断 |
| IRQ6 | 0x0E | 5 | 扩展总线外部中断 |
| TMR1IRQ | --- | 1 | 芯片内部 TMR1,ETR232i 内部定时中断 |

各个中断对应的中断服务程序定义为：

```
void interrupt IRQ5_ISR(__CPPARGS);
void interrupt IRQ6_ISR(__CPPARGS);
void interrupt TMR1_ISR(__CPPARGS);
```

```
int InstallISR( int intno);          //该函数的功能是置中断。输入参数为中断号。
int UninstallISR( int intno );    //该函数的功能是恢复中断。
```

本节内容主要介绍这三类中断资源的程序调用方法。

**（1）系统定时器与软中断的调用**

与 PC 兼容系统定时器一样，ETR232i 的系统定时器（Timer0）也是每秒中断约 18.2 次，既中断周期约为 55ms，通常称为一个 tick。为了让应用程序系统定时器执行各种定时任务，每次系统定时中断均会调用软中断 int 0x1C，这样应用程序可简单的加载软中断 0x1C，方便地进行各种定时操作了。加载定时任务的方法就是直接加载中断 0x1C 的中断服务程序，而中断 0x1C 服务程序通常是通过管理存储器变量来达到定时的目的。

```
void CaptureTime( )
{
    oldhandler = getvect(0x1C);
    setvect(0x1C, Service);
}

void ReleaseTime( )
{
    setvect(0x1C, oldhandler);
```

```
}

void interrupt Service( __CPPARGS )
{
    if(Mask == 0 )
    {
        Times++;
        if(Times == 2)      Flash(0);
        if(Times == 8)    { Flash(1);   Times=0 }
     }
}

void Flash( int type)
{
    if(type == 0 )        LCD_FillBar(cuX, cuY, cuX+7, cuY+1, 1);
    else                    LCD_FillBar(cuX, cuY, cuX+7, cuY+1, 0);
}
```

调用函数 CaptureTime( )，设置中断 0x1C 的中断服务程序为 Service（）,每 55ms 中断一次，执行该中断服务程序 Service（），通过管理变量 Times，在屏幕上显示不同的图形，以达到图形光标闪烁的效果。

注意：中断 0x1C 是以 tick（55ms）为节拍工作的，因此定时的最小单位为一个 tick，对于更长的定时，可以通过对 tick 计数来实现，而对于更短时间的定时操作则需要利用系统的 Timer1 来实现了。

**（2）内部定时器中断的使用**

ETR232i 内部包含的一个定时器 Timer1 可供应用程序使用，Timer1 的输入时钟为 1MHz，可直接调用 SetTMR1( unsigned int Divider) 函数来设置 Timer1 的分频数，Timer1 的分频输出已接到中断输入 TMR1IRQ（内部）以支持定时中断功能。

TMRDEMO.PRJ 工程文件由 ETR232i.CPP、TMRISR.CPP 及 TMRDEMO.CPP 构成。

ETR232i.CPP 定义了低层的接口函数，TMRISR.CPP 提供了定时中断服务程序，TMRDEMO.CPP 包含了主函数 main( )。TMR1IRQ 为内部中断，其时钟输出已接到了相应的中断输入，不需要外接线。TMRDEMO.CPP 首先设置 Timer1 输出一个周期为 10ms 的方波信号（N=10000），设置 GPIO 为输出模式，然后启动定时中断服务程序。定时中断服务程序 TMTISR.CPP 以 toggle 方式设置 GPIO 的输出，并让内部计数值加 1。用户可参见源代码以了解详细的过程。

**（3）外部硬件中断的使用**

在实际应用中，一般使用硬件中断的方法是编写中断服务程序，在程序初始化中安装中断程序，在程序退出时卸载中断程序。中断服务程序一般通过存储器变量与上层应用程序交换数据。

ISRDEMO.PRJ 工程文件由 ETR232i.CPP 、 TMRISR.CPP 、 ISR.CPP 和 ISRDEMO.CPP 构成，其中模块 ISR.CPP 提供了在扩展总线的 3 个外部中断函数的编写方法。我们提供的中断服务程序例子中，定时中断如 3.2 节所描述那样工作，把周期（10ms）变化的 GPIO 通过 ETR232i 评估板连接到 2 个外部输入上，从而触发外部中断。外部中断每中断一次，分别对各自的计数值加 1，主程序 ISRDEMO 将在 LCD 上显示这些计数值。用户在实际的运用中，可根据需要修改中断服务程序。

# 3、TCP/IP 协议库及其应用

ETR232i TCP/IP 通讯软件是一组可在 ETR232i 环境中调用的软件运行函数，通过在 ETR232i 以太网接口实现基于 TCP/IP 或 UDP/IP 协议的通讯功能。我们提供了六个测试程序。这些测试程序放置在光盘中<Ethernet>子目录下。

测试的方法是将我们提供的测试程序下载到 ETR232i 运行，而另一端在 PC 机上运行我们提供的基于 WinSocket 的 VC 源码及相应的 Windows 运行程序。我们提供的 VC 用户不仅可利用此程序对 ETR232i 的网络功能进行测试，还可在此基础上进行修改，以构成自己的上位机程序。

有关 VC 程序的介绍已超出了本手册的范围，目前市面上有大量相关参考书可供用户在设计上位机程序时参考。此外 TCP/IP 通讯是与操作系统、编程工具无关的，用户也可采用其他编程工具，如 VB、Delphi、C++Builder 等来设计自己的网络通讯程序。

在对网络程序进行调试时，需注意的是对程序调试的网络环境的选择，网络程序的调试网络环境应该选择没有大量广播包，ping 包的小规模网络（LAN），否则调试过程中，由于没有及时地处理这些数据包，而让系统缓冲区（buffer）被占满，从而导致网络失效。

## 3.1 TCP/IP 协议库函数原型定义

嵌入式网络模块 TCP 通讯软件是一组可在英创嵌入式网络模块环境中调用的软件运行函数，通过其以太网接口实现基于 TCP/IP 或 UDP/IP 协议的通讯功能。客户的应用程序通过在 BC 集成编程环境下直接调用该通讯软件的各个函数，并把其库函数文件连入客户应用程序的工程文件中，即可实现完整的网络通讯。

为了同时适应在单任务 DOS 操作系统及 RTOS 下设计网络通讯程序，英创公司的嵌入式网络模块 TCP 通讯软件采用直接面对连接的方法构成，并针对工业控制的特点进行了封装，构成了若干个可直接使用函数，使用户能以最快的速度操作使用英创嵌入式网络模块的网络资源。

以下对各个函数作详细介绍：

（1）int InitEthernetNet( char* IPString=NULL, char* MSKString=NULL,int PHY_SPD=0 );

功能描述：初始化以太网接口。

输入参数：

char* IPString

本地 IP 地址串，形式为：“n1.n2.n3.n4”； IP 串为空，则系统按缺省 IP 地址(“192.168.201.22”)进行初始化，若 IP 为“0.0.0.0”,将发送 DHCP 请求报文，请求自动分配 IP，如果网络环境中没有运行 DHCP 服务器，将超时退出，初始化失败。

char* MSKString

用户根据需要设置子网掩码,形式为：“m1.m2.m3.m4”。当 MSKString 串为空时，系统将按 IP 地址的类型设置其子网掩码。用户也可根据实际应用需要，设置相应的子网掩码，比如为 B 类 IP“162.168.201.22”，设置 C 类子网掩码 “255.255.255.0”。用户可通过子网掩码的有效设置，来提高网络 IP 地址的利用率。

int PHY_SPD

用户可根据实际通讯情况，把 100M 网络强制设置为 10M 网络，有效地降低对网络物理连线的要求。

PHY_SPD = 0：缺省设置值，自动协商网络类型（10M/100M 自适应）

PHY_SPD = 10：强制设置为 10M 网络类型

PHY_SPD = 100：强制设置为 100M 网络类型

返回值：

若正确返回值为 0，若初始化失败返回值<0。

备注：

- 当采用 DHCP 获取自动 IP 时，同时也获得子网的掩码、网关 IP 以及 DNS 服务器的 IP。
- 如果在调用该函数之前没有设置网关，则系统为无网关类型。
- 在采用参数对网络进行初始化时，要求所配置的网络参数为有效参数，即网关 IP 和本地 IP 需是同一网段的，否则将导致网络初始化失败，函数将返回错误代码 –10。

（2）int TermEthernetNet( );

功能描述：关闭以太网接口。

返回值：若正确返回值>=0，若失败返回值<0。

参数：无

（3）int MyPort( )

功能描述：用于产生随机的端口号。

返回值：返回随机端口号，其范围为 2000—9000。

（4）int ConnOpen ( char * to, char * protoc , int lp , int rp , int flags) ;

功能描述：建立基于 TCP 或 UDP 通讯服务。

输入参数：

char* to

应用程序若以服务器模式工作，则填“*”(如在基于 UDP 的组播程序设计时,就填”*”); 以客户端模式工作，则填相应的服务器 IP 地址，如“192.168.201.97”。

protoc

定义传输层协议，可取：”TCP/IP”或 ”UDP/IP”

int lp

本地端口号。在服务器模式应用中，lp 为知名端口号。以客户端模式打开连接时，建议 lp 的值通过调用函数 int MyPort( )随机产生，应避免每次打开连接时使用相同的端口号，因为在同时打开多个连接时，采用相同的端口号，有可能造成服务器端对各个连接处理出错。

int rp

若应用程序工作在服务器模式下，rp 应置为 0，若应用程序工作在客户端模式下，rp 为连接对方端口号，即服务器的知名端口号。

int flags

一般置为 0。但在单任务的运行环境中（如 DOS 环境），对 TCP 连接宜采用无阻塞方式打开连接，这时 flags 应置为 NONBLOCKOPEN。对无阻塞方式打开的连接，应用程序需通过调用 ConnIsEstablished( int conno )来确定连接是否建立。因为，即使连接未真正建立，ConnOpen( )也能返回正常值。对 UDP 连接，如果是一般的无连接应用，flags 为 NONBLOCKOPEN；如果是点对点应用，flags 应设为 UDP_P2P。在程序中连接号实际也起到 handler

的作用。

返回值：

>＝0 表示正确返回，其值为连接号，应用程序操作 TCP/IP 通讯接口时，都需使用该连接号；若失败返回值<0。

（5）int ConnClose( int conno,  int flags );

功能描述：关闭连接。

输入参数：

int conno 连接号；

int flags flags 置为 0，对连接进行阻塞模式的关闭操作，即等关闭操作完成后才退出关闭操作。如果 flags 等于 1，对连接进行无阻塞模式的关闭操作，程序仅仅是启动关闭操作即退出。在该函数中，flags 的缺省值为 0。

返回值：

0 正常关闭；

EBADF 无效连接号，关闭操作未执行。

备注：

- 对 TCP 连接来说，只有连接关闭正常，才能保证所有的数据收发是可靠的。
- 在网络出现异常时，对连接进行关闭操作，可能会有 2 分钟左右的延时。

（6）int ConnRead( int conno, char* buff, int len );

功能描述：从指定连接号读取网络接口接收到的数据。对于 TCP 连接，若该函数调用频率低于对端发送的频率，此时可能已收到几个数据包，且其数据总长度小于接收缓冲区 buff 的长度，则这些数据将被一次性读入 buff，以最大限度释放内部缓冲区资源，而从应用层看会出现所谓的“合包”现象。

输入参数：

int conno 连接号；

char* buff char 型指针变量，为存放读出数据的缓冲区 buff 地址。

int len 缓冲区的字节长度，对 TCP 连接，最大缓冲区长度应不大于 1460 字节；对 UDP 来说，最大缓冲区长度应不大于 1472 字节。

返回值：

0 表明对端已执行了关闭连接或连接出现异常错误，此时应关闭本连接。

>0 正常返回，返回值为实际读取的字节数；

EBADF 无效连接号，读取操作未执行。

EWOULDBLOCK 对无阻塞方式打开的连接，表示无数据，可重复调用该函数。

ETIMEOUT 超时错误，由应用程序决定后续处理。

EMSGSIZE 数据太长，表明接收缓冲区太小。

（7）int ConnWrite( int conno, char* buff, int len，int PushFlg=0 );

功能描述：发送缓冲区中的数据到指定连接号网络接口。

输入参数：

int conno 连接号。

char* buff char 型指针变量，为存放发送数据的缓冲区 buff 地址。

int len 缓冲区的字节长度，对 TCP 连接，最大缓冲区长度应不大于 1460 字节；对 UDP 来说，最大缓冲区长度应不大于 1472 字节。

int PushFlg：

=0 为普通发送

=1 为急迫发送，仅对 TCP 连接有效。

返回值：

>=0 正常返回，返回值为实际发送的字节数；

EBADF 无效连接号，发送操作未执行。

ETIMEOUT 超时错误，由应用程序决定后续处理。

EMSGSIZE 数据太长，发送数据长度超过了内部缓冲区大小。

备注：

在普通发送时，如果需要连续发送的数据包小于最大缓冲区长度，而且在连续发送之间没有作相应的延时，TCP/IP 则有可能将连续的包作合包处理后发送，以提高网络的传输效率，所以在接收方收到的是已经合过的包。在众多工业应用中，可以通过设置急迫标志，并在数据包与数据包之间作短暂的延时（如 1ms）就可以使

接收方收到与发送方一致的包。从而避免在应用层对 TCP 数据流进行“断句”分析的麻烦。

（8）int ConnIsEstablished( int conno );

功能描述：检查连接是否建立。

输入参数：连接号。

返回值：

1：连接已建立；

0：连接还未建立。

（9）int ConnCanSend( int conno, int len );

功能描述：检查是否能发送指定长度的数据。

输入参数：

int conno 连接号；

int len 需发送数据的长度。

返回值：

1：接已能发送指定长度的数据；

0：连接不能发送指定长度的数据。

（10）int ConnHasData( int conno );

功能描述：检查是否有数据可读取。

输入参数：连接号。

返回值：

1 连接已存在可读取的数据；

0 连接无可读取的数据。

（11）int ConnIsFinished( int conno );

功能描述：可使用该函数检查该连接的对端是否进行了关闭连接的操作。若对端已进行了关闭操作，系统应立即调用 ConnClose( int conno )关闭该连接。

输入参数：连接号

返回值：

1 连接已关闭；

0 连接未关闭。

（12）int ConnIsFatal( int conno );

功能描述：可使用该函数检查连接的状态。

输入参数：连接号

返回值：

1 连接出现异常，如连接复位（RST）或超时。

0 连接 OK

备注：应用程序在调用该函数检查到连接出现了异常，应立即关闭该连接，即连接方停止数据的传输，以释放诸如缓冲区之类的资源。

（13）int JoinMulticastGroup ( char* MulticastIP );

功能描述：

本系统加入组播(且组播 IP 地址被 MulticastIP 唯一标示)，使本系统能收到向本组播 IP 地址发送的信息。(注:不能同时加入两个及以上的组播，要加入新的组播必须先退出上一组播)

输入参数：

MulticastIP 组播的 IP 地址.

返回值:

<0 该函数调用失败

>=0 调用成功并返回此连接标示号

备注：组播 IP 地址的设置需在打开网络连接之前进行。

（14）int LeaveMulticastGroup ( );

功能描述：退出已加入的组播

返回值：若成功返回 0; 若失败返回 –1

备注：退出组播需在相应网络连接关闭以后进行。

（15）int SetGateWayIP( char* GateWayIP ) ;

功能描述：

在不同局域网之间通信时需要用到此函数设置网关地址。使本机可以通过网关路由到外网的远端主机。

输入参数：

char* GateWayIP 网关的 IP 地址.

返回值:

<0 该函数调用失败

=0 该函数调用成功

备注：

- 设置网关需在网络初始化之前进行。
- 若采用 DHCP 自动获取 IP，可不调用该函数。

（16）int GetIP( int conno, unsigned char* IPCode );

功能描述：获取连接对方的 IP 地址

输入参数：

int conno 由 ConnOpen( )打开的连接号.

unsigned char* IPCode 获取此连接的对方的 IP 地址

返回值:

<0 该函数调用失败

=0 该函数调用成功

备注：此函数只对已建立的连接有效。

（17）int GetOWNIP( unsigned char* IPCode );

功能描述：获取本地的 IP 地址

输入参数：

unsigned char* IPCode 本地的 IP 地址

返回值:

<0 该函数调用失败

=0 该函数调用成功

（18）int FTP_getput（char *host, char *file , int mode, char *FTPdir, char *FTPusername, char *FTPpassword, unsigned long timeout, int FTPMode）；

功能描述：

以 FTP 客户端方式向远端 FTP 服务器发送文件或从远端 FTP 服务器获取文件。

输入参数：

char *host 远端主机的 IP 地址，如“192.168.201.34”。

char *file 被操作的文件名，如“myfile.txt”。

int mode =0; 从远端服务器获取 ASCII 文件

=1；从远端服务器获取 2 进制文件

=2；向远端服务器发送 ASCII 文件

=3；向远端服务器发送 2 进制文件

char *FTPdir 所操作的文件在 FTP 服务器上的存储路径，NULL 表示当前目录

char *FTPusername 登录时使用的用户名，如“guest”

char *FTPpassword 登录时使用的密码，如“888”

unsigned long timeout 定义的 timeout 时间，单位为毫秒，缺省值为 5 分钟

int FTPMode 登录 FTP 的模式，=0 为标准模式

=1 为 passive 模式

返回值:

!=0 该函数调用失败

=0 该函数调用成功

（19）int FTP_putstream（char *host, char *file , char *FTPdir, char *FTPusername, char *FTPpassword, char* stream, unsigned int slen, unsigned long timeout, int FTPMode）；

功能描述：

以 FTP 客户端方式将数据流发送到远端 FTP 服务器文件。

输入参数：

char *host 远端主机的 IP 地址，如“192.168.201.34”。

char *file 被操作的文件名，如“myfile.txt”。

char *FTPdir 所操作的文件在 FTP 服务器上的存储路径，NULL 表示当前目录

char *FTPusername 登录时使用的用户名，如“guest”

char *FTPpassword 登录时使用的密码，如“888”

char *stream 数据流

unsigned int slen 数据流长度

unsigned long timeout 定义的 timeout 时间，单位为毫秒，缺省值为 5 分钟

int FTPMode 登录 FTP 的模式，=0 为标准模式

=1 为 passive 模式

返回值:

!=0 该函数调用失败

=0 该函数调用成功

（20）int FTP_getstream（char *host, char *file , char *FTPdir, char *FTPusername, char *FTPpassword, , char* stream, unsigned long timeout, int FTPMode）；

功能描述：

以 FTP 客户端方式从远端 FTP 服务器读取文件内容。

输入参数：

char *host 远端主机的 IP 地址，如“192.168.201.34”。

char *file 被操作的文件名，如“myfile.txt”。

char *FTPdir 所操作的文件在 FTP 服务器上的存储路径，NULL 表示当前目录

char *FTPusername 登录时使用的用户名，如“guest”

char *FTPpassword 登录时使用的密码，如“888”

char *stream 读取远端 FTP 服务器文件的数据

unsigned long timeout 定义的 timeout 时间，单位为毫秒，缺省值为 5 分钟

int FTPMode 登录 FTP 的模式，=0 为标准模式

=1 为 passive 模式

返回值:

>0 远端 FTP 服务器读取数据的长度

（21）int FTP_server( void* pTsk，unsigned long timeout, char* FTPusername, char* FTPpassword )

功能描述：

FTP 服务器运行程序，FTP 的实现是以非阻塞方式和阻塞方式相结合为特征的，当未与客户端建立连接时，处于非阻塞方式，函数可返回；当与客户端建立连接后，FTP 服务器转入阻塞模式，这时需等待 FTP 请求完成后，函数才会退出。

输入参数：

void* pTsk 系统保留，总设置为空指针 NULL。

unsigned long timeout 超时时间，单位为毫秒，若 FTP 服务器在 mseconds 内未接受任何服务请求，则函数自动退出。

char* FTPusername 设置 FTP 服务器的用户名，若 FTPusername 为空指针，则采用缺省用户名“guest”。

char* FTPpassword 设置 FTP 服务器的密码，若 FTPpassword 为空指针，则采用缺省密码“888”。

返回值:

= 0 等待 FTP 连接请求，可再次调用该函数。

= 1 已有 FTP 连接请求，等待连接建立，需再次调用该函数。

= 2 FTP 连接完成，可再次调用该函数。

（22）void NetPackagePro( )

功能描述：

处理低层的网络数据包，如广播包、ping 包等。应用程序如果在一段时间内没有操作网络，应调用该函数，使得网络数据包能够得到及时处理，释放网络内部资源，从而避免网络阻塞。

（23）int EthernetLinkTest( )

功能描述：

用于检查以太网物理连接 link 状态。

返回值：

0： 物理连接正常 link OK。

-30: 物理连接出错 link fail。

（24）int Ping( char* IPStr, unsigned long Milliseconds )

功能描述：

采用 ICMP 协议查询远端的服务器主机是否工作。

输入参数：

char* IPStr 远端主机的 IP 地址，如：“192.168.201.121”

unsigned long Milliseconds

等待远端服务器主机应答的最长时间，单位为毫秒，时间一般设置在 100—3000 毫秒之间。

返回值：

=0： 远端服务器主机有应答，说明远端服务器主机处于运行状态。

!=0： 无相应的服务器主机应答。

（25）int SetReloadWDTInNet( void(*)Reload() )

功能描述：

把 WatchDog 定时器的加载函数传给网络运行库，使之能在网络函数运行期间及时加载 WatchDog。

输入参数：

void(*)Reload() WatchDog 加载函数指针。

备注：在网络初始化成功后，此函数只需调用一次即可。

（26）int ARPHost( char* IPStr, unsigned long Milliseconds )

功能描述：

采用 ARP 包来查询远端的服务器主机是否工作。

输入参数：

char* IPStr　远端主机的 IP 地址，如：“192.168.201.121”

unsigned long Milliseconds

等待远端服务器主机应答的最长时间，单位为毫秒，时间一般设置在 100—1000 毫秒之间。

返回值：

>=0：　远端服务器主机有应答，说明远端服务器主机处于运行状态。

<0：　无相应的服务器主机应答。

## 3.2 TCP 服务器演示程序

TCPSVR.PRJ 是 TCP/IP Ethernet Server 的例子，建立一个基于 TCP 的服务器。将接收到的数据串，倒序后又发送给客户端。

TCPSVR.PRJ 工程文件包括有以下模块：

ETR_TCP.LIB　提供 TCP/IP 协议库 API 函数

UTILITY.CPP　提供读取 Ethernet 的 MAC 地址函数

ETR232i.CPP　提供 ETR232i 硬件接口函数

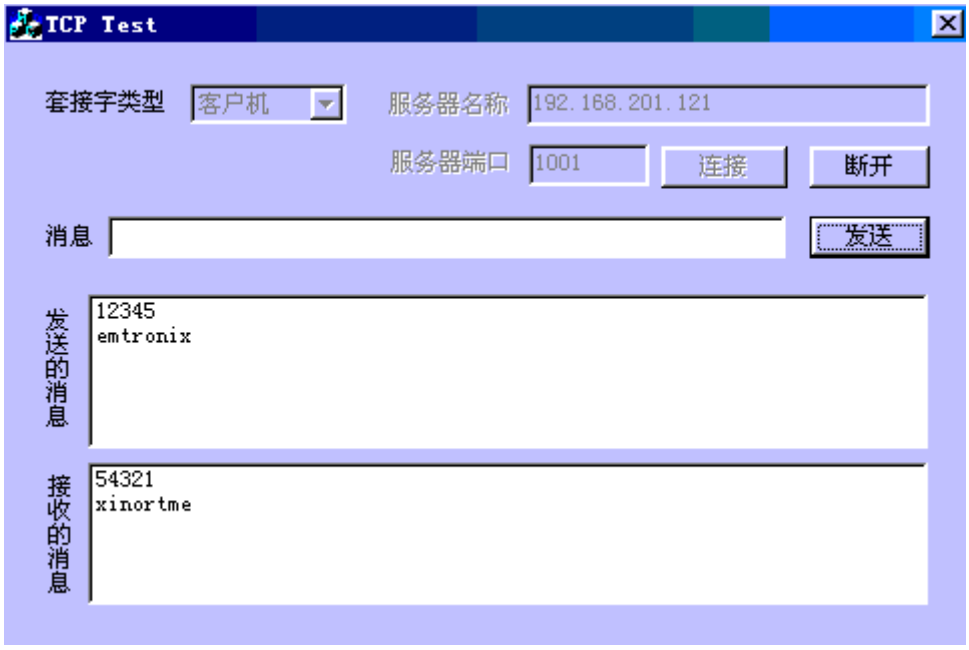
DS1302.CPP　提供 ETR232i 实时时钟的操作函数

TCPSVR.CPP　PRJ 文件的主函数

在 TCPSVR.CPP 中包含主函数 main( )，首先调用 InitEthernet( char* IPStr )初始化以太网接口，其中的参数 IP 地址可以通过命令行参数的形式输入，如果没有带参数，则采用缺省的 IP 地址(“192.168.201.22”)进行初始化,接着打开一个端口号为 1001 的无阻塞的 TCP(“*”)连接，等待连接建立后，可以实现数据接收显示并将接收数据倒序后再向外发送的功能，同时可以接收响应网络命令，设置实时时钟和读取实时时钟。

在对该程序进行测试时，在另一端的 PC 机上运行 tcptest.exe，将出现对话框，选择“客户机”，在“服务器名称”栏目中填入为嵌入式模块分配的 IP 地址（如：192.168.201.22），然后点击“连接”。TCP 连接一旦建立，原灰化的消息框将被激活，填入字符串点击发送后，数据将发送到嵌入式模块，嵌入式模块收到数据后，会将数据顺序颠倒后通过局域网返回到测试 PC，在接收数据区中显示出来。如果在消息框中，不填入任何字符，点击发送后，将发

送设置实时时钟命令以及实时时钟到嵌入式模块，嵌入式模块收到后，设置 ETR232i 的实时时钟；如果在消息框中，填入字符“Time”，点击发送后，嵌入式模块收到该命令后，将读取 ETR232i 上的实时时钟，生成相应的串返回到测试 PC。




有关 TCP/IP 通讯接口函数的详细使用方法请参见 3.1 节的说明。

## 3.3 TCP 客户端演示程序

TCPCLNT.PRJ 是 TCP/IP Ethernet Client 的例子，以客户端模式打开 TCP 的连接，主动申请和服务器端建立连接，一旦建立后，可接收处理服务器端发送的数据，如果没有接收到对端的数据，每间隔一个固定的时间主动发送一次数据，即所谓的心跳测试，在该例程中定义的心跳时间为 10 秒。

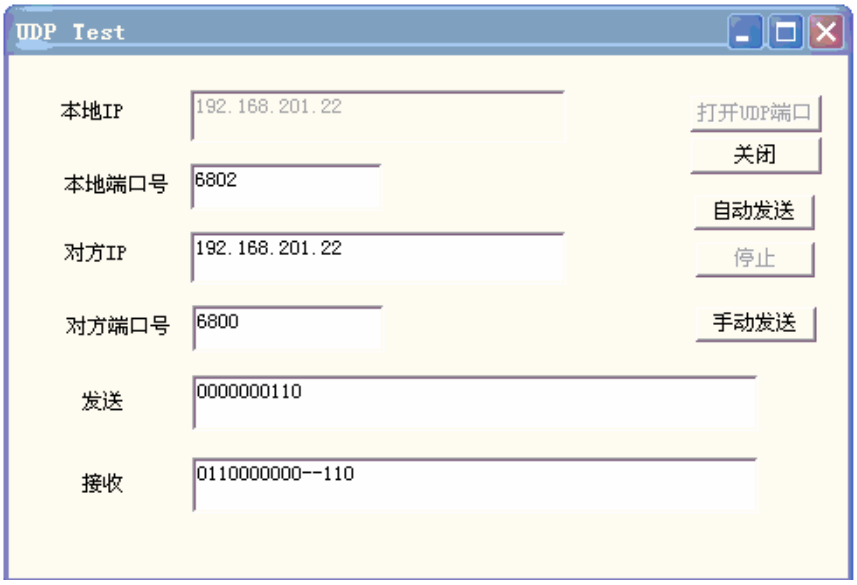
TCPCLNT.PRJ 工程文件由 TCPCLNT.CPP 及 ETR_TCP.LIB 构成。在 TCPCLNT.CPP 中包含主函数 main( )，首先调用 InitEthernet( char* IPStr )初始化以太网接口，其中的参数 IP 地址可以通过命令行参数的形式输入，如果没有带参数，则采用缺省的 IP 地址(“192.168.201.22”)进行初始化,接着打开一个 TCP 的连接，需带有目的 IP 地址的连接，目的 IP 地址可以通过命令行参数的第二项输入，如果没有带有参数，则采用缺省值“192.168.201.3”。命令行参数输入的方法如：TCPCLNT 192.168.201.26 192.168.201.16。命令行参数的第一项为本地 IP 地址，第二项为目的 IP 地址。打开一个连到对方 IP，端口号为 1001 的无阻塞 TCP 连接，等待连接建立后，程序进入到主循环。

在另一端 PC 机上运行我们提供的 TCPTest 程序，选择“服务器”，服务器端口不变，还是选择“1001”，然后点击“侦听”。TCP 连接一旦建立，原灰化的消息框将被激活，填入字符串点击发送后，数据将发送到嵌入式模块，嵌入式模块收到数据后，会将数据通过局域网返回到测试 PC，在接收数据区中显示出来。如果测试 PC 不主动发送数据，每隔 10 秒测试 PC 将收到嵌入式模块发送数据，并在接收数据区中显示出来。

## 3.4 UDP 服务器演示程序

UDPSVR.PRJ 工程文件由 UDPSVR.CPP、UTILITY.CPP 及 ETR_TCP.LIB 构成。在 UDPSVR.CPP 中包含主函数 main( )，首先调用 InitEthernet( char* IPStr )初始化以太网接口，其中的参数 IP 地址可以通过命令行参数的形式输入，如果没有带参数，则采用缺省的 IP 地址(“192.168.201.22”)进行初始化，接着打开一个端口号为 6800 的非阻塞模式（NONBLOCKOPEN）的 UDP 连接，对 UDP 连接，如果是一般的无连接应用，flags 为 NONBLOCKOPEN；如果是点对点应用，flags 应设为 UDP_P2P。连接打开后，可实现数据接收显示功能，并把接收到的字符串倒序后又发送到对端。

ETR232i 端运行 UDPSVR.EXE，另一端在 PC 机上运行 UDPTest 测试程序。将出现对话框，选择填好对端的 IP 以及端口号，然后点击“打开 UDP 端口”，其他灰化的按钮将变亮，此时再点击“自动发送”，程序将自动发送以序列号生成的字符串，并显示在发送消息框内；嵌入式模块收到数据后，会将数据顺序颠倒后通过局域网返回到测试 PC，在接收数据区中显示出来，同时显示 UDP 实际接收到的数据包的个数。



## 3.5 UDP 客户端演示程序

UDPCLNT.PRJ 是 UDP/IP Ethernet Client 的例子，以客户端模式打开 UDP 的端口，一方面可接收发往该 UDP 端口的数据包，如果没有接收到的相应数据包，另一方面又每间隔一个固定的时间主动往对端 UDP 端口发送一次数据，即所谓的心跳测试，在该例程中定义的心跳时间为 10 秒。

UDPCLNT.PRJ 工程文件由 UDPCLNT.CPP、ETR232i.CPP 及 ETR_TCP.LIB 构成 。在 UDPCLNT.CPP 中包含主函数 main( )，首先调用 InitEthernet( char* IPStr )初始化以太网接口，其中的参数 IP 地址可以通过命令行参数的形式输入，如果没有带参数，则采用缺省的 IP 地址(“192.168.201.22”)进行初始化,接着打开一个 UDP 的连接，需带有目的 IP 地址的连接，目的 IP 地址可以通过命令行参数的第二项输入，如果没有带有参数，则采用缺省值 “192.168.201.16” 。 命 令 行 参 数 输 入 的 方 法 如 ： UDPCLNT 192.168.201.26 192.168.201.16。命令行参数的第一项为本地 IP 地址，第二项为目的 IP 地址。打开一个本地端口号为 6800 的无阻塞 UDP 连接，对端的端口号为 6802。连接打开后，可接收发往该 UDP 端口的数据包，同时有可每间隔一个固定的时间主动向目的端（目的 IP、目的端口）发送数据。

在另一端 PC 机上运行我们提供的 UDPTest 程序，将出现对话框，选择填好对端的 IP 以及端口号，然后点击“打开 UDP 端口”，其他灰化的按钮将变亮。可接收到对端发送的数据，同时也可选择“自动发送”，主动发送数据到 ETR232i。

## 3.6 FTP 服务器演示程序

FTPSVR.PRJ 提供的是 FTP 服务器程序，PC 机上运行 cuteFTP。FTP 连接建立后，可以下载或上传文件进行测试。

FTPSVR.PRJ 工程文件由 FTPSVR.CPP、REDAINI.CPP 和 ETR_TCP.LIB 构成。在 FTPSVR.CPP 中包含主函数 main( )，首先读取配置文件 netconf.ini 的内容，从中获得所需配置的本地 IP、掩码以及网关地址，再调用 InitEthernet( … )初始化以太网接口，接着进入主循环，调用 FTP 服务器运行程序 int FTP_server( NULL )，FTP 的实现是以非阻塞方式和阻塞方式相结合为特征的，当未与客户端建立连接时，处于非阻塞方式，函数可返回；当与客户端建立连接后，FTP 服务器转入阻塞模式，这时需等待 FTP 退出后，函数才会退出。函数 FTP_server( NULL )输入参数在单任务时，选择 NULL。返回值为 0：等待 FTP 连接

打开； 1：等待 FTP 连接建立；2：一次 FTP 连接完成，可再次调用该函数。Ftp_Server 的用户名=“guest” 密码=“888”。

## 3.7 支持多连接的服务器演示程序

TCPSVR1.PRJ 提供的是一个支持同时打开多连接的 TCP 服务器运行程序。 工程文件包括有：

| | |
|---|---|
| ETR_TCP.LIB | 提供 TCP/IP 协议的接口函数 |
| UTILITY.CPP | 提供读取 Ethernet 的 MAC 地址函数 |
| TCPSERV.CPP | 提供打开、建立 TCP 连接的管理程序 |
| TCPSVR1.CPP | PRJ 文件的主循环 |

在 TCPCLNT.CPP 中，其通讯方式是以服务器模式方式（或称被动方式）打开 TCP 连接，并等待连接建立。在该模块中定义了一个 TCPServerManager 的类，来实现以上功能。采用 C++的类来设计程序有助于得到更加稳定的程序结构。class TCPServerManager 类中定义了连接的三种状态： CLOSED 表示关闭、START 表示连接打开、OPEN 表示连接已经建立。提供的公共函数有：

TCPServerManager ( int MaxNumConn);

功能描述：

此为 class TCPServerManager 的构造函数，对各类变量进行初始化。定义了同时打开的连接数，由所带参数 MaxNumConn 来定，其大小不超过 MAXNCONN。并将各个连接的初始状态置为 CLOSED。由于网络的资源有限，总共提供的连接数为 20 个，在本应用案例中，MAXNCONN 定义为 8，即最多同时打开 8 个以太网连接。

int ThisConnHasData( )

功能描述：

用于检查连接是否接收到有数据，其返回值 >=0 表明接收到网络数据，该返回值为收到数据的连接编号。

int CloseAll( );

功能描述：关闭所有已打开的连接。

该程序的测试方法和 TCPSVR 完全一致。

# 4、PPP 及 TCP/IP 协议库及其应用

ETR232i PPP/TCP/IP 通讯软件是一组可在 ETR232i 环境中调用的软件运行函数，通过在 ETR232i 接口实现基于 PPP 的 TCP/IP 或 UDP/IP 协议的通讯功能。我们提供了一个利用 PPP 协议的 GPRS 应用的测试程序。测试程序放置在光盘中<PPPNet>子目录下。

## 4.1 PPP TCP/IP 协议库函数原型定义

嵌入式网络模块 PPP TCP 通讯软件是一组可在英创的嵌入式网络模块环境中调用的软件运行函数，通过在嵌入式网络模块串口 COM2 实现 PPP 通讯协议，并进一步实现在其上的 TCP/IP 协议的通讯功能。客户的应用程序通过在 BC 集成编程环境下直接调用嵌入式网络模块 PPP TCP 通讯软件的各个函数，并把其库函数文件连入客户应用程序的工程文件中，即可实现完整的网络通讯。

为了同时适应在单任务 DOS 操作系统及 RTOS 下设计网络通讯程序，英创公司的嵌入式网络模块 TCP 通讯软件采用直接面对连接的方法构成，并针对工业控制的特点进行了封装，构成了若干个可直接使用函数，使用户能以最快的速度操作使用英创嵌入式网络模块的网络资源。

以下对各个函数作详细介绍：

（1） int InitPPPNet(int PortNum=0, char* BAUD=NULL);

功能描述：初始化网络接口。

输入参数：

int PortNum

用于 PPP 连接的串行端口号，设置为 0 表示使用 COM1；设置为 1 表示使用

COM2。

char* BAUD

用于配置 PPP 连接的串行端口的波特率。

“9600” ----- 波特率为 9600bps

“19200” ----- 波特率为 19200bps

“38400” ----- 波特率为 38400bps

“57600” ----- 波特率为 57600bps（缺省配置）

“115200”----- 波特率为 115200bps

如 BAUD 串为空 NULL，按缺省的波特率（57600bps）进行配置。

返回值：

若正确返回值为 0，若初始化失败返回值<0。

备注：

此函数调用时可以不填参数而使用缺省值。

（2） int TermPPPNet( );

功能描述：关闭网络接口。

返回值：若正确返回值>0，若失败返回值<0。

输入参数：无

（3）int MyPort( )

功能描述：用于产生随机的端口号。

返回值：返回随机端口号，其范围为 2000—9000。

（4）int ConnOpen ( char * to, char * protoc , int lp , int rp , int flags) ;

功能描述：建立基于 TCP 或 UDP 通讯服务。

输入参数：

char* to

应用程序若以服务器模式工作，则填“*”(如在基于 UDP 的组播程序设计时,就填”*”); 以客户端模式工作，则填相应的服务器 IP 地址，如“192.168.201.97”。

protoc

定义传输层协议，可取：”TCP/IP”或 ”UDP/IP”

int lp

本地端口号。在服务器模式应用中，lp 为知名端口号。以客户端模式打开连接时，建议 lp 的值通过调用函数 int MyPort( )随机产生，应避免每次打开连接

时使用相同的端口号，因为在同时打开多个连接时，采用相同的端口号，有可能造成服务器端对各个连接处理出错。

int rp

若应用程序工作在服务器模式下，rp 应置为 0，若应用程序工作在客户端模式下，rp 为连接对方端口号，即服务器的知名端口号。

int flags

一般置为 0。但在单任务的运行环境中（如 DOS 环境），对 TCP 连接宜采用无阻塞方式打开连接，这时 flags 应置为 NONBLOCKOPEN。对无阻塞方式打开的连接，应用程序需通过调用 ConnIsEstablished( int conno )来确定连接是否建立。因为，即使连接未真正建立，ConnOpen( )也能返回正常值。对 UDP 连接，如果是一般的无连接应用，flags 为 NONBLOCKOPEN；如果是点对点应用，flags 应设为 UDP_P2P。在程序中连接号实际也起到 handler 的作用。

返回值：

>＝0 表示正确返回，其值为连接号，应用程序操作 TCP/IP 通讯接口时，都需使用该连接号；若失败返回值<0。

（5）int ConnClose( int conno, int flags );

功能描述：关闭连接。

输入参数：

int conno 连接号；

int flags flags 置为 0，对连接进行阻塞模式的关闭操作，即等关闭操作完成后才退出关闭操作。如果 flags 等于 1，对连接进行无阻塞模式的关闭操作，程序仅仅是启动关闭操作即退出。在该函数中，flags 的缺省值为 0。

返回值：

0 正常关闭；

EBADF 无效连接号，关闭操作未执行。

备注：

- 对 TCP 连接来说，只有连接关闭正常，才能保证所有的数据收发是可靠的。

- 在网络出现异常时，对连接进行关闭操作，可能会有 2 分钟左右的延时。

（6）int ConnRead( int conno, char* buff, int len );

功能描述：从指定连接号读取网络接口接收到的数据。对于 TCP 连接，若该函数调用频率低于对端发送的频率，此时可能已收到几个数据包，且其数据总长度小于接收缓冲区 buff 的长度，则这些数据将被一次性读入 buff，以最大限度释放内部缓冲区资源，而从应用层看会出现所谓的“合包”现象。

输入参数：

int conno 连接号；

char* buff char 型指针变量，为存放读出数据的缓冲区 buff 地址。

int len 缓冲区的字节长度，对 TCP 连接，最大缓冲区长度应不大于 1460 字节；对 UDP 来说，最大缓冲区长度应不大于 1472 字节。

返回值：

1 表明对端已执行了关闭连接或连接出现异常错误，此时应关闭本连接。

>0 正常返回，返回值为实际读取的字节数；

EBADF 无效连接号，读取操作未执行。

EWOULDBLOCK 对无阻塞方式打开的连接，表示无数据，可重复调用该函数。

ETIMEOUT 超时错误，由应用程序决定后续处理。

EMSGSIZE 数据太长，表明接收缓冲区太小。

（7）int ConnWrite( int conno, char* buff, int len，int PushFlg=0 );

功能描述：发送缓冲区中的数据到指定连接号网络接口。

输入参数：

int conno 连接号。

char* buff char 型指针变量，为存放发送数据的缓冲区 buff 地址。

int len 缓冲区的字节长度，对 TCP 连接，最大缓冲区长度应不大于 1460 字节；对 UDP 来说，最大缓冲区长度应不大于 1472 字节。

int PushFlg：

=0 为普通发送

=1 为急迫发送，仅对 TCP 连接有效。

返回值：

>=0 正常返回，返回值为实际发送的字节数；

EBADF 无效连接号，发送操作未执行。

ETIMEOUT 超时错误，由应用程序决定后续处理。

EMSGSIZE 数据太长，发送数据长度超过了内部缓冲区大小。

备注：

在普通发送时，如果需要连续发送的数据包小于最大缓冲区长度，而且在连续发送之间没有作相应的延时，TCP/IP 则有可能将连续的包作合包处理后发送，以提高网络的传输效率，所以在接收方收到的是已经合过的包。在众多工业应用中，可以通过设置急迫标志，并在数据包与数据包之间作短暂的延时（如 1ms）就可以使接收方收到与发送方一致的包。从而避免在应用层对 TCP 数据流进行“断句”分析的麻烦。

（8）int ConnIsEstablished( int conno );

功能描述：检查连接是否建立。

输入参数：连接号。

返回值：

1：连接已建立；

0：连接还未建立。

（9）int ConnCanSend( int conno, int len );

功能描述：检查是否能发送指定长度的数据。

输入参数：

int conno 连接号；

int len 需发送数据的长度。

返回值：

1：接已能发送指定长度的数据；

0：连接不能发送指定长度的数据。

（10）int ConnHasData( int conno );

功能描述：检查是否有数据可读取。

输入参数：连接号。

返回值：

1 连接已存在可读取的数据；

0 连接无可读取的数据。

（11）int ConnIsFinished( int conno );

功能描述：可使用该函数检查该连接的对端是否进行了关闭连接的操作。若对端已进行了关闭操作，系统应立即调用 ConnClose( int conno )关闭该连接。

输入参数：连接号

返回值：

1 连接已关闭；

0 连接未关闭。

（12）int ConnIsFatal( int conno );

功能描述：可使用该函数检查连接的状态。

输入参数：连接号

返回值：

1 连接出现异常，如连接复位（RST）或超时。

0 连接 OK

备注：应用程序在调用该函数检查到连接出现了异常，应立即关闭该连接，即连接方停止数据的传输，以释放诸如缓冲区之类的资源。

（13）int GetIP( int conno, unsigned char* IPCode );

功能描述：获取连接对方的 IP 地址

输入参数：

int conno 由 ConnOpen( )打开的连接号.

unsigned char* IPCode 获取此连接的对方的 IP 地址

返回值:

<0 该函数调用失败

=0 该函数调用成功

备注：此函数只对已建立的连接有效。

（14）int GetOWNIP( unsigned char* IPCode );

功能描述：获取本地的 IP 地址

输入参数：

unsigned char* IPCode 本地的 IP 地址

返回值:

<0 该函数调用失败

=0 该函数调用成功

（15）int FTP_getput（char *host, char *file , int mode, char *FTPdir, char *FTPusername, char *FTPpassword, unsigned long timeout, int FTPMode）;

功能描述：

以 FTP 客户端方式向远端 FTP 服务器发送文件或从远端 FTP 服务器获取文件。

输入参数：

char *host 远端主机的 IP 地址，如“192.168.201.34”。

char *file 被操作的文件名，如“myfile.txt”。

int mode =0; 从远端服务器获取 ASCII 文件

=1；从远端服务器获取 2 进制文件

=2；向远端服务器发送 ASCII 文件

=3；向远端服务器发送 2 进制文件

char *FTPdir 所操作的文件在 FTP 服务器上的存储路径，NULL 表示当前目录

char *FTPusername 登录时使用的用户名，如“guest”

char *FTPpassword 登录时使用的密码，如“888”

unsigned long timeout 定义的 timeout 时间，单位为毫秒，缺省值为 5 分钟

int FTPMode 登录 FTP 的模式，=0 为标准模式

=1 为 passive 模式

返回值:

!=0 该函数调用失败

=0 该函数调用成功

（16）int FTP_putstream（char *host, char *file , char *FTPdir, char *FTPusername, char *FTPpassword, char* stream, unsigned int slen, unsigned long timeout, int FTPMode）;

功能描述：

以 FTP 客户端方式将数据流发送到远端 FTP 服务器文件。

输入参数：

char *host 远端主机的 IP 地址，如“192.168.201.34”。

char *file 被操作的文件名，如“myfile.txt”。

char *FTPdir 所操作的文件在 FTP 服务器上的存储路径，NULL 表示当前目录

char *FTPusername 登录时使用的用户名，如“guest”

char *FTPpassword 登录时使用的密码，如“888”

char *stream 数据流

unsigned int slen 数据流长度

unsigned long timeout 定义的 timeout 时间，单位为毫秒，缺省值为 5 分钟

int FTPMode 登录 FTP 的模式，=0 为标准模式

=1 为 passive 模式

返回值:

!=0 该函数调用失败

=0 该函数调用成功

（17）int FTP_getstream（char *host, char *file , char *FTPdir, char *FTPusername, char *FTPpassword, , char* stream, unsigned long timeout, int FTPMode）;

功能描述：

以 FTP 客户端方式从远端 FTP 服务器读取文件内容。

输入参数：

char *host 远端主机的 IP 地址，如“192.168.201.34”。

char *file 被操作的文件名，如“myfile.txt”。

char *FTPdir 所操作的文件在 FTP 服务器上的存储路径，NULL 表示当前目录

char *FTPusername 登录时使用的用户名，如“guest”

char *FTPpassword 登录时使用的密码，如“888”

char *stream 读取远端 FTP 服务器文件的数据

unsigned long timeout 定义的 timeout 时间，单位为毫秒，缺省值为 5 分钟

int FTPMode 登录 FTP 的模式，=0 为标准模式

=1 为 passive 模式

返回值:

>0 远端 FTP 服务器读取数据的长度

（18）int FTP_server( void* pTsk，unsigned long timeout, char* FTPusername, char* FTPpassword )

功能描述：

FTP 服务器运行程序，FTP 的实现是以非阻塞方式和阻塞方式相结合为特征的，当未与客户端建立连接时，处于非阻塞方式，函数可返回；当与客户端建立连接后，FTP 服务器转入阻塞模式，这时需等待 FTP 请求完成后，函数才会退出。

输入参数：

void* pTsk 系统保留，总设置为空指针 NULL。

unsigned long timeout 超时时间，单位为毫秒，若 FTP 服务器在 mseconds 内未接受任何服务请求，则函数自动退出。

char* FTPusername 设置 FTP 服务器的用户名，若 FTPusername 为空指针，则采用缺省用户名“guest”。

char* FTPpassword 设置 FTP 服务器的密码，若 FTPpassword 为空指针，

则

采用缺省密码“888”。

返回值:

= 0 等待 FTP 连接请求，可再次调用该函数。

= 1 已有 FTP 连接请求，等待连接建立，需再次调用该函数。

= 2 FTP 连接完成，可再次调用该函数。

（19）void NetPackagePro( )

功能描述：

处理低层的网络数据包，如广播包、ping 包等。应用程序如果在一段时间内没有操作网络，应调用该函数，使得网络数据包能够得到及时处理，释放网络内部资源，从而避免网络阻塞。

（20）int Ping( char* IPStr, unsigned long Milliseconds )

功能描述：

采用 ICMP 协议查询远端的服务器主机是否工作。

输入参数：

char* IPStr 远端主机的 IP 地址，如：“192.168.201.121”

unsigned long Milliseconds

等待远端服务器主机应答的最长时间，单位为毫秒，时间一般设置在 100—3000 毫秒之间。

返回值：

=0： 远端服务器主机有应答，说明远端服务器主机处于运行状态。

!=0： 无相应的服务器主机应答。

（21）int SetReloadWDTInNet( void(*)Reload() )

功能描述：

把 WatchDog 定时器的加载函数传给网络运行库，使之能在网络函数运行期间及时加载 WatchDog。

输入参数：

void(*)Reload()　WatchDog 加载函数指针。

备注：在网络初始化成功后，此函数只需调用一次即可。

（22） int SetupPPPLink（unsigned long　timeout）；

功能描述：作为 PPP 的客户端，启动 PPP 连接，进行 PPP 认证，并检查 PPP 状态。

输入参数：

unsigned long timeout

PPP 客户端启动 PPP 连接时设定的一个超时时间，时间单位为秒，缺省值为

300，即 5 分钟。

| 返回值 | 描　述 |
|---|---|
| 0 | PPP 打开、PPP 认证成功，进入网络层协议状态。 |
| <0 | PPP 打开、PPP 认证失败。 |

（23） int GetPPPState（）；

功能描述：获取 PPP 状态

| 返回值 | 描　述 |
|---|---|
| 1 | PPP 打开、PPP 认证成功，进入网络层协议状态。 |
| 0 | PPP 处于协商状态 |
| -1 | PPP 处于关闭状态，可再重新建立 PPP 连接 |

（24） int SetPPPUsername（char* NewName ）；

功能描述：设置新的用户名称。

输入参数：char* NewName

需要设置的新的用户名称，如果 NewName 为 NULL，则沿用缺省的用户名，“guest”

备注：

PPP 库只保留 NewName 的指针，因此用户应把 NewName 作为全局或静态量，切忌作为自动变量放在堆栈里。

（25） int SetPPPPassword（char* NewWord ）；

功能描述：设置新的用户密码。

输入参数：

char* NewWord 需要设置的新的用户密码，如果 NewWord 为 NULL，则沿用

缺省的用户密码，“888”

备注：

PPP 库只保留 NewWord 的指针，因此用户应把 NewWord 作为全局或静态量，切忌作为自动变量放在堆栈里。

（26） int ForcePPPLinkDown（）；

功能描述：强制执行 PPP 终止操作，使 PPP 返回关闭状态。

（27） int SendATCmdString( char* ATCmdString );

功能描述：发送 AT 指令。

输入参数：

char* ATCmdString 需发送的 AT 指令串。

返回值：

>0 发送 AT 指令的长度。

<0 发送 AT 指令失败。

（28） int GetATCmdEcho( );

功能描述：获取 AT 指令的响应值。

返回值：

>0 AT 指令的响应值

-1 没有响应值

（29） int SetSerMuxUp( );

功能描述：安装多路转换协议，并创建 3 个虚拟逻辑通道。该函数应在发送“AT+CMUX=0”并返回“OK”，延时 2 秒后调用。

返回值：

0 安装多路转换协议成功，并成功创建 3 个虚拟逻辑通道。

<0 安装多路转换协议失败。GPRS 模块将在 5 秒后回到 AT 命令状态。

## 4.2 GPRS 应用

PPP 协议的 GPRS 应用的测试程序在光盘“PPPNet”目录下，该目录中提供了两个测试程序：

GPRSTST.EXE：用于测试 ETR232i 利用 GPRS 拨号上网的功能。

GPRS232.EXE：ETR232i 利用 GPRS 拨号上网实现 RS232 接口的数据和远端数据服务中心进行数据的透明传输。

下面就对这两个程序进行详细的说明。

### 4.2.1 GPRS 自动拨号上网

GPRSTST 实现了 GPRS 自动拨号上网的功能，该程序运行后，如果 LCD 屏上显示“IP=**.**.**.**”，即表示 GPRS 拨号上网成功。

GPRSTST.PRJ 工程文件包含以下文件：

| | |
|---|---|
| ETR_PPP.LIB | 提供 PPP TCP/IP 协议的接口函数 |
| ETR232i.CPP | 提供 ETR232i 硬件接口函数 |
| GPRS.CPP | 提供了进行 GPRS 自动拨号上网的函数 |
| GPRSTST.CPP | PRJ 文件的主函数 |

下面重点介绍 GPRS.CPP 模块所实现的功能。

利用 GPRS 无线上网有以下几个步骤：1）GPRS 无线模块上电；2）通过发送 AT 指令与中国移动网的节点服务器建立连接；3）在连接建立后，进行相应的 PPP 配置，PPP 配置成功后，即可获得节点服务器配置 IP 地址，进行网络通讯了。为了方便用户的使用，我们对 GPRS 无线上网的这几个步骤进行封装，其中利用 C 语言的功能强大的函数指针变量以及有限状态机的处理机制，提供给用户一个单一的函数 PPP_Running( )，该函数每执行一个步骤就退出，并返回当前的状态，以便于应用程序的设计。该函数实现的功能包括 GPRS 自动拨号上网，上网后若出现断线，PPP 链接重新建立，以保证设备一直在线。用户只需根据函数的返回值来判断 GPRS 是否上网，以便于应用程序进行所需的 TCP/IP 通讯。

函数的定义包含在 GPRS.H 中，在 GPRS.CPP 中实现。GPRS.H 中定义了两个函数：

PPPGPRSState PPP_Running( )

功能描述：实现 GPRS 自动拨号上网，不断调用该函数，可保证 GPRS 一直在线。

返回值：返回进行 GPRS 拨号上网状态。

0： GPRS 模块上电操作

1： 初始化 PPP 网络相关参数。

2： 发送“ATE0”指令操作。

3： 发送“AT+CGATT?”指令，用于检查模块是否附着到 GPRS 网络

4： 发送“AT+CGATT=1”指令，用于设置模块附着到 GPRS 网络

5： 发送“AT+CGDCONT=1,"IP","CMNET ""，设置 GPRS 通讯参数。

6： 发送“ATD*99***1#”，拨号到 GPRS 节点服务器。

7： 进行 PPP 配置、协商。

8： 表明 GPRS 上网、PPP 配置成功，应用程序可进行 TCP/IP 通讯。

9： 断开 GPRS 网络，关闭 PPP 网络，同时关闭了 GPRS 模块电源。

void PPP_ReStart( )

功能描述：重新启动 PPP 配置。该函数主要针对在进行 TCP/IP 数据通讯过程中出现异常时的应用，可通过调用该函数重新启动 GPRS 操作。

返回值：无。

GPRSTST 程序在运行时，将在 LCD 屏上显示“ RIdx= ”，根据这个序号对应以上的说明，用户可判断 GPRS 拨号上网的状态，直到 LCD 屏上显示“IP=**.**.**.**”，表明 GPRS 拨号上网成功。

### 4.2.2 GPRS 数据透明传输

**程序介绍**

程序 GPRS232 实现了通过串口数据和远方数据中心之间的透明传输。GPRS232.PRJ 文件包括有以下模块：

| | |
|---|---|
| ETR_PPP.LIB | 提供 PPP TCP/IP 协议的接口函数 |
| ETR232i.CPP | 提供 ETR232i 硬件接口函数 |
| READINI.CPP | 提供从配置文件中读取参数接口函数 |
| GPRS.CPP | 提供了进行 GPRS 自动拨号上网的函数 |
| TCPCLNT.CPP | 提供打开、建立 TCP 连接的管理程序 |
| RS232X3.CPP | 提供 ETR232i 串口驱动程序 |
| GPRS232.CPP | PRJ 文件的主循环 |

GPRS.CPP 模块已经在上一节中作了详细的介绍，这里主要介绍 TPCLNT.CPP 和 GPRS232.CPP。

在 TCPCLNT.CPP 中，其通讯方式是以客户端方式（或称主动方式）打开 TCP 连接，并等待连接建立，并提供相应的读写数据的接口函数。在该模块中定义了一个 TCPClient

的类，来实现以上功能。采用 C++的类来设计程序有助于得到更加稳定的程序结构。class TCPClient 类中定义了连接的三种状态：CLOSED 表示关闭、START 表示连接打开、OPEN 表示连接已经建立。提供的公共函数有：

void Init( struct SERVER_ID* pSvrID );

功能描述：

带入初始化参数，即通过结构参数 pSvrID 设置远端数据中的 IP 地址和端口号。

int SetTimeoutCnt( long NumTick );

功能描述：

用于设置 timeout 的时间，其参数 NumTick 为设置超时时间的 Tick 值，1 个 Tick 值约为 55ms，如果 NumTick 则表明不用设置超时。

int IsTimeout( );

功能描述：

检查是否超时，函数返回 1 表明已经超时，返回 0 没有超时。

CONN_STATE Running( );

功能描述：

作为 TCPCliend 的主要运行函数。包括三项处理：

1) 连接处于 CLOSED 状态，则打开一个新的连接，连接打开成功后，其状态由 CLOSED 转换到 START。
2) 连接处于 START 状态，则检查连接是否建立，同时检查超时时间，如果连接建

立，其状态由 START 转换到 OPEN；如果时间超时连接还未建立，则关闭连接，

状态又回到 CLOSED。
3) 连接处于 OPEN 状态，则检查连接是否有异常，或是被对方关闭，如果没有检

查到则一直保持 OPEN 状态，否则将关闭 TCP 连接，其状态回到 CLOSED。

该函数返回 TCP 连接的状态。

int WriteData( char* DatBuf, int DatLen )

功能描述：

通过 TCP 连接进行发送数据。该函数对于连接状态处于 OPEN 时才有效。

int ReadData( char* DatBuf, int DatLen )

功能描述：

检查 TCP 连接是否收到数据。该函数对于连接状态处于 OPEN 时才有效。

int Close( );

功能描述：关闭已打开的连接。

GPRS232.CPP 定义了该工程文件的主循环，利用各个模块所提供的接口函数，可进行 GPRS 自动拨号上网，和远端网络数据中心建立 TCP 连接，以实现数据透明传输的功能。在该模块程序中，1）第一步进行的是初始化操作，包括从配置文件中读取配置参数，以及对串口的初始化操作等；2）进入程序的主循环，首先调用 PPP_Running( ) 自动进行 GPRS 拨号上网操作，直到该函数返回 PPP 的状态值为 PPPLINKUP，表明 GPRS 拨号上网已成功。循环再进入到下一步操作，和远端服务中心建立 TCP 的连接，直接调用 TCPManager->Running( )，直到该函数返回连接的状态为 OPEN。此时主循环一方面检查网络对端是否有数据，如果有将接收的数据通过 RS232 发送出去；另一方面检查 RS232 串口是否接收到数据，如果有将接收到的串口通过 TCP 连接发送到对端服务中心。从而实现了 RS232 和远端数据中心之间的数据透明传输。同时需要说明的还有，在该例程中，还有一个心跳包的处理，如果在设置的心跳时间内一直没有数据的通讯传输，程序将主动发送一个心跳测试包到远端的服务中心，在该程序中的心跳测试时间设置为 400 个 Tick 值，约为 20 秒。具体实现的方法请参见该程序代码。

**程序测试**

在进行测试时，和串口相连的设备可是一台 PC1 机，在该 PC 机上运行串口调试助手程序（在提供光盘“测试工具”目录下）或者是 Windows 的超级终端程序；而另一端远端数据中心可在另一台 PC2（已连入互联网的 PC2）上运行我们公司提供的 VC 测试程序

TCPTest（在提供光盘“测试工具”目录下）。

1. 让 PC2 具有 Internet 互联网可访问的 IP 地址，有以下两种方法：

   1）互联网静态的 IP 地址。

   2）通过 ADSL 或 Modem 拨号上的互联网，获取的是动态的互联网 IP 地址，运行 >ipconfig，即可获得其动态 IP 地址。

2. 配置文件 config.ini 介绍

   在运行 GPRS232 程序时，须带上配置文件 config.ini，该文件主要用于配置一些通讯参数。

   [Config]

   RemoteIP=61.157.22.196

   用于定义互联网上需连接的服务器的 IP 地址(即 PC2 的互联网 IP)。

   RemotePort=1001

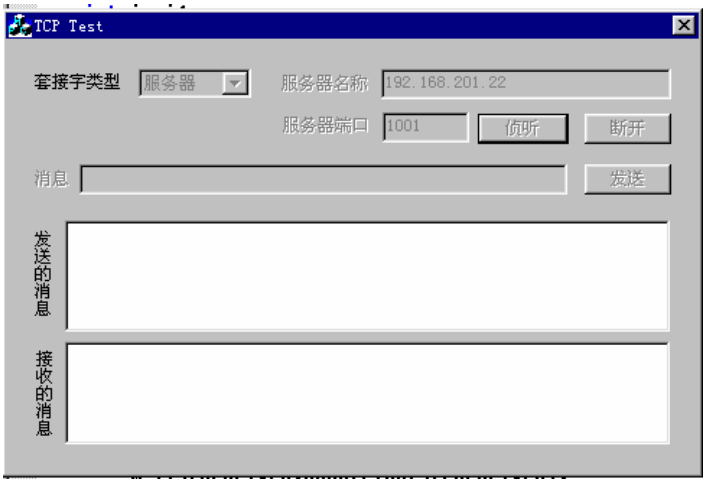
   用于定义在互联网上远端服务器的端口号。

   BaudIdx=12

   用于定义 ETR232i 和设备（PC1）RS232 通讯的波特率。选择 1 对应的波特率为 115.2kbps，2 对应的波特率为 57.6kbps，3 对应的波特率为 38.4kbps，6 对应的波特率为 19.2kbps，12 对应的波特率为 9600bps，以此类推。

3. “远方服务数据中心”测试程序 TCPTest 简介

   TCPTest 是运行在 PC2 上的程序，在运行该程序之前，首先须使 PC2 连上互联网，PC2 上网后，运行 TCPTest 程序，选择为“服务器”模式，并点击侦听，其中的消息框将处于灰化状态。如下图：



如果 ETR232i 与 PC2 之间 TCP 连接建立成功，灰化的消息框将变亮，此时就可通过消息框对 ETR232i 发送字符串，同时接收设备通过 ETR232i 发送的数据。接收到的数据将显示在接收的消息框内。

# 5、串口驱动程序

ETR232i 嵌入式串行通讯程序能实现 RS232/RS485 串行通讯数据的收发，我们提供了两个测试程序。测试的方法是将测试程序下载到 ETR232i 运行，而另一端在 PC 机上运行串口调试助手程序或者是 windows 提供的超级终端程序。

## 5.1 串口驱动 API 函数

ETR232i 提供了 3 个异步串口 COM1、COM2、COM3，在缺省状态下 COM1 串口的最高波特率为 115.2kbps。其中 COM1 与 PC 机的串口完全兼容，IO 映射访问。COM2 和 COM3 来自于 RDC1610 内部。为了方便用户掌握串口操作的具体细节，各个串口的低层驱动函数的定义和说明均以源代码的方式提供给客户，共包括 6 个函数，下面就各个函数的定义作一说明。这几个函数的定义在 RS232X3.h，函数的说明在 RS232X3.cpp。

（1） int InitUART( int ComIdx, int BaudIdx ， int Parity)

功能描述：初始化串口

输入参数：

ComIdx：串行端口号设置，可选值如下表

| COMIdx | 助记符 | 简要说明 |
|---|---|---|
| 0 | COM1 | 16C550，端口地址为 0x3F8 |
| 1 | COM2 | 来自 CPU 内部，16C550 兼容 |
| 2 | COM3 | 来自 CPU 内部，16C550 兼容 |

BaudIdx: 通讯波特率设置，用户可根据需要进行配置，本演示程序中采用值为 12（9600bps）。现将常用可选值列于下表，供用户参考：

| BaudIdx | 实际设置的波特率 |
|---|---|
| 1 | 115.2 kbps |
| 2 | 57.6 kbps |
| 3 | 38.4 kbps |
| 6 | 19.2 kbps |
| 12 | 9600 bps |

| | |
|---|---|
| 24 | 4800 bps |
| 48 | 2400 bps |
| 96 | 1200 bps |

Parity：用于选择串口通讯的奇偶校验。

0 无校验

1 奇校验

2 偶校验

注：本初始化函数中对数据通讯格式的设置为：8 bit,1 stop,no Parity，用户也可根据自己的需要在源程序中对配置进行必要的调整。

（2） int InstallISR( int ComIdx )

功能描述：置中断。

输入参数：

ComIdx：串行端口号设置,定义同 InitUART。

返回参数：

0：安装中断成功。

（3） int UninstallISR( int ComIdx )

功能描述：恢复中断。

输入参数：

ComIdx：串行端口号设置,定义同 InitUART。

返回参数：

0：恢复中断成功。

（4） int PutOutputData( int ComIdx, char abyte )

功能描述：将待发送数据置入输出数据缓冲区中。

输入参数：

ComIdx：串行端口号设置,定义同 InitUART。

abyte：待发送数据字节

返回参数：

0：输出数据缓冲区未满，写数成功。

-1：输出数据缓冲区已满，写数失败。

（5） void StartSend( int ComIdx )

功能描述：启动中断,开始数据发送过程

输入参数：

ComIdx：串行端口号设置,定义同 InitUART。

（6） int GetInputData( int ComIdx )

功能描述：从接收数据缓冲区中取出数据

输入参数：

ComIdx：串行端口号设置,定义同 InitUART。

返回参数：

整数：返回接收数据缓冲区中所取整数

-1：接收数据缓冲区为空

## 5.2 RS232 收发演示程序

232DEMO.PRJ 工程文件由 232DEMO.CPP 及 RS232.CPP 构成。在 232DEMO.CPP 中包含主函数 main( )，main( )首先根据命令参数（端口号）对相应串口进行初始化，开中断，实现数据接收显示及将接收数据向外发送的功能。程序中，端口号通过命令行参数的形式输入。如：232DEMO COM2。如果是在调试状态下运行，为：td –rp 232DEMO COM2。

由 232DEMO.prj 编译连接生成的 232DEMO.exe 作为服务器端，它首先响应客户端的接收请求，将客户端发送的数据在显示屏上显示出来，同时又将接收到的数据发送到客户端。

建议客户端执行串口调试助手程序或者 windows 提供的超级终端程序。

## 5.3 RS485 数据收发

RS485 的数据收发的低层驱动函数和 RS232 是完全一致的，所以相关的函数介绍请参见 RS232 的说明。RS485 和 RS232 的区别在于 RS485 是半双工的，所以 RS485 收发的数据都是以具有一定特征的数据帧为单位的，为了方便测试，在我们提供的程序 RS485demo.prj 和 Clnt.prj 是以字符“！”作为一个数据帧的结束标志。

# 6、LCD 汉字显示程序

ETR232i 的一类主要应用是作为智能终端的核心平台，智能终端总是带有一种显示单元。ETR232i 作为一种高效、低成本的产品解决方案，专门针对小型的 LCD 模块（分辨率通常在 122×32 至 320×240）设计了一套通用的汉字及图形显示接口函数。这些 API 函数均以源码形式提供，用户不仅可以直接使用，还可根据自身需求进行必要的修改。

关于 LCD 汉字驱动及相应的测试程序在光盘中<LCD>目录下。

## 6.1 汉字显示接口函数定义

ETR232i 的 LCD 汉字显示 API 共包括 7 个函数，以及一个从 CPP 文件中提取汉字字模的工具程序 source.exe（源码在 source.cpp 中）。这些函数的详细定义如下：

(1) int LCD_Init( char* CHN_FntFile, char* ASCII_FntFile )

功能描述：对 LCD 进行图形方式初始化，并加载汉字和西文字符字库。

输入参数：

char* CHN_FntFile：汉字字库文件名，我们提供的汉字字库文件 mlib.chr

char* ASCII_FntFile：ASCII 码文件名，我们提供的 ASCII 码库文件 ascii.chr

(2) int LCD_Clearup( )

功能描述：在图形方式下对 LCD 进行清屏处理。

输入参数：无

(3) int LCD_SetMode( int Mode )

功能描述：设置 LCD 的显示模式。

输入参数：

int Mode：0 表示 copy 方式显示模式

1 表示 xor 方式显示模式

(4) int LCD_PutPixel( int x0, int y0, int color );

功能描述：在图形方式下进行描点。

输入参数：

int x0：点的列坐标，范围 0~127（或 0-319），不可大于或等于 128（或 320）

int y0：点的行坐标，范围 0~63（或 0-239），不可大于或等于 64（或 240）

int color：颜色设置，0（不显示）或者 1（显示）。

(5) int LCD_DrawLine( int x0, int y0, int x1, int y1, int color );

功能描述：在图形方式下进行画直线

输入参数：

x0：直线开始列坐标，范围 0~127（或 0-319），不可大于或等于 128（或 320）

y0：直线开始行坐标，范围 0~63（或 0-239），不可大于或等于 64（或 240）

x1：直线结束列坐标，范围 0~127（或 0-319），不可大于或等于 128（或 320）

y1：直线结束行坐标，范围 0~63（或 0-239），不可大于或等于 64（或 240）

color：颜色设置，0 或者 1。

(6) int LCD_FillBar( int x0, int y0, int x1, int y1, int color );

功能描述：在图形方式下画 Bar 条。

输入参数：

x0：Bar 开始列坐标，范围 0~127（或 0-319），不可大于或等于 128（或 320）

y0：Bar 开始行坐标，范围 0~63（或 0-239），不可大于或等于 64（或 240）

x1：Bar 结束列坐标，范围 0~127（或 0-319），不可大于或等于 128（或 320）

y1：Bar 结束行坐标，范围 0~63（或 0-239），不可大于或等于 64（或 240）

color：颜色设置，0 或者 1。

(7) int LCD_WriteString( int x0, int y0, char* pStr, int color, int len=0 );

功能描述：在图形方式下写字符串（中文或西文字符）。

输入参数：

x0：字符输入列坐标，范围 0~127（或 0-319），不可大于或等于 128（或 320）

y0：字符输入行坐标，范围 0~63（或 0-239），不可大于或等于 64（或 240）

＊pStr：需显示的字符串

color：字符颜色设置， 0 或 1

len：字符显示长度，缺省设为 0，表示按 str 实际长度显示；否则按 length

定义的长度显示。

## 6.2 汉字显示程序的有关事项

1. 在 BC 集成环境（IDE）下建立包含由英创公司提供的 LCD 低层驱动模块（如 HD61202.cpp 、SED1335.cpp 等）、Dotlib.cpp、Lcd_api.cpp 及其他应用模块的工程文件，在需要的场合调用相关的图形操作函数或汉字显示函数。最后编译连接形成可执行文件（.EXE）。

2. 汉字字模 mlib.chr 可以通过我们提供的 source.exe 搜索生成，直接编译 source.cpp 文件即可生成 source.exe 运行文件。

source 文件在运行时，需带上参数，参数为需搜索的 cpp 文件名。

如： >source tst1.cpp 搜索 tst1.cpp 中的汉字，生成新的汉字库 mlib.chr，同时生成新的头文件 source.h。此时需再重新编译工程文件，最后连接形成与用户应用相关的可执行文件。

如果从多个 CPP 文件提取汉字，则执行 source *.cpp。注意不要把 source.cpp 放在应用程序同一个目录，以免把 source.cpp 中的汉字也搜索进去。

3. mlib.chr 最多只能包含 512 个汉字。需要全汉字的应用，则需要把整个汉字字模文件加载到 ETR232i 的高端存储器中，有关这方面的问题，请与英创公司技术支持部门联系。

4. 显示中西文必须使用工具软件 tdrf 把 mlib.chr 和 ascii.chr 先拷贝入 ETR232i 的运行盘中（B:\ 或 C:\ ），然后程序才能正常运行。

5. 建议客户的应用程序，不用汉字来写注释，以保证 mlib.chr 包含的都是有效汉字字模。

## 6.3 图形方式下汉字和西文显示示例

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <math.h>
#include "lcd_api.h"
```

```
void main()
{
    int       x, y;

    clrscr();                                          // 清屏
    printf( "\nDOT LCD Demo\n" );
    // goto graghic mode
        LCD_Init( "mlib.chr", "ascii.chr" );     //   打开 LCD 图形显示模式，并加载字模文件
    LCD_SetMode( 1 );                                  // set to XOR mode
    // show chinese char at center position
    char BufStr[20];
    strcpy( BufStr, "**  欢迎测试  **" );
    x = (128-strlen(BufStr)*8)/2;
    y = 8;
    LCD_WriteString( x, y, BufStr, 1 );
                                          //图形方式下显示西文或中文字符
    strcpy( BufStr, "嵌入式网络模块" );
    x = (128-strlen(BufStr)*8)/2;
    y = 24;
    LCD_WriteString( x, y, BufStr, 0 );
                                          //图形方式下显示西文或中文字符
    // draw lines
    LCD_DrawLine( 0, 42, 127, 42, 1 );
    LCD_DrawLine( 0, 63, 127, 63, 1 );
                                          //图形方式画直线
    // draw curve
    for( x=0; x<128; x++ )
        {
        y = 53 - 10.0*sin( 2.0*M_PI*x/24.0 );;
        LCD_PutPixel( x, y, 1 );
```

```
        }
                                    //图形方式下画曲线
    for( x=0; x<128; x++ )
        {
        y = 53 - 10.0*sin( 2.0*M_PI*x/24.0 );;
        LCD_PutPixel( x, y, 1 );
        }
// 图形方式下再画与上相同的曲线，如果是在异或（xor）模式下，该操作相当于清曲线。
    LCD_FillBar( 0, 42, 127, 63, 0 );
    LCD_FillBar( 0, 42, 127, 63, 1 );
    LCD_FillBar( 0, 42, 127, 63, 0 );          //图形方式下画 Bar 条。
}
```

# 附录 1 BC31 集成开发环境的基本配置

为了编写正确的应用程序代码，需对 BC 集成开发环境的相关参数做出相应的设置。本附录将对与应用程序开发相关的主要配置参数与以说明，具体的设置方法请参见“BC 使用简介”中的相关章节。

（1） 编译路径的设置

建议在 AUTOEXEC.BAT 中把“BC\BIN”加入系统的路径列表中，这样用户可在任意工作目录中启动 BC。若 BC 放置在 C:\BC，则在 Autoexec.bat 可加入：

PATH = C:\BC\BIN;%PATH%

把 BC 软件所在目录设入 BC 集成开发环境的目录选项中，如 BC 安装在 C:\BC，则 Include 目录应设为 C:\BC\INCLUDE；而 Library 目录应设为：C:\BC\LIB。

（2） 模式的设置

由于我们的 TCP/IP 库采用的是 Large 模式，因此用户在包含 TCP/IP 库的工程文件中，需要将编译模式设置为 Large 模式。如果用户采用了我们提供的 RTOS 库文件，还需要将编译模式设置为 Huge 模式。

（3） 代码生成选项设置

代码生成选项中有 3 项内容用户需仔细确认设置，它们是：

浮点设置：由于 386EX 不带协处理器，因此该项应设置为仿真“Emulation”。

指令设置：应设置为“80186”，不能选择“80286”或“80386”。

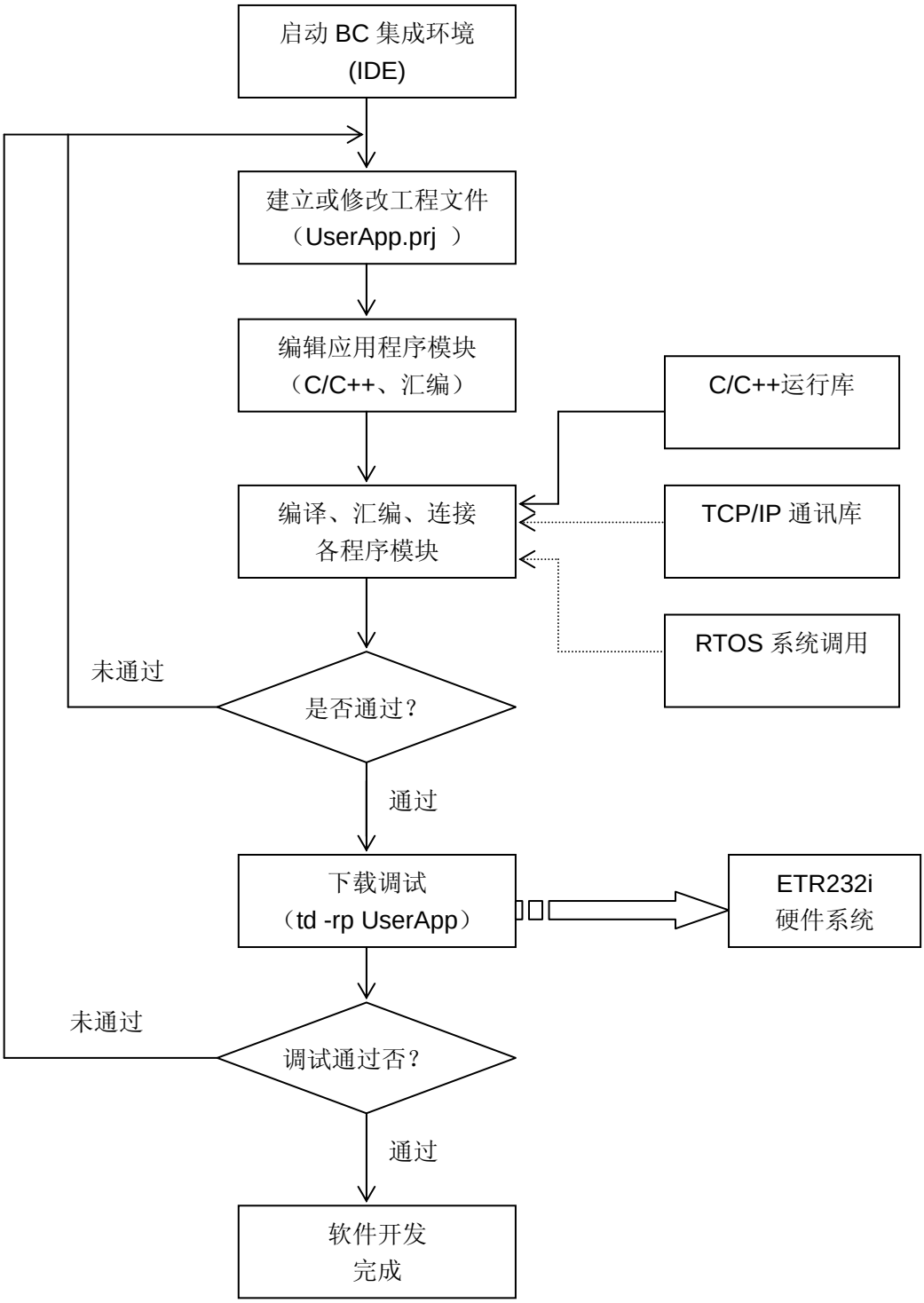
调试设置：建议用户设置带调试信息的编译，这样可在 TD 中进行源码调试。

（4） 运行库的设置

由于 ETR232i 已不支持通常的 VGA 显示，所以无需连接图形库“Graphics library”。标准的 Run-time 库应设置为静态“Static”，注意其他库的选项都应设置为无“None”。

# 附录 2　基于 ETR232i 的应用软件开发流程

如下图所示，ETR232i 的软件开发可完全在 PC 上按以下步骤进行：




第 1 步　在 PC 环境下启动 BC/C++或 TC/C++的集成开发环境 IDE。如：

C:\Myapp>BC

第 2 步 建立工程（Project）文件，在 Prj 工程文件中主要是定义系统的各软件模块。Prj 工程文件，可根据设计需求随时进行修改。

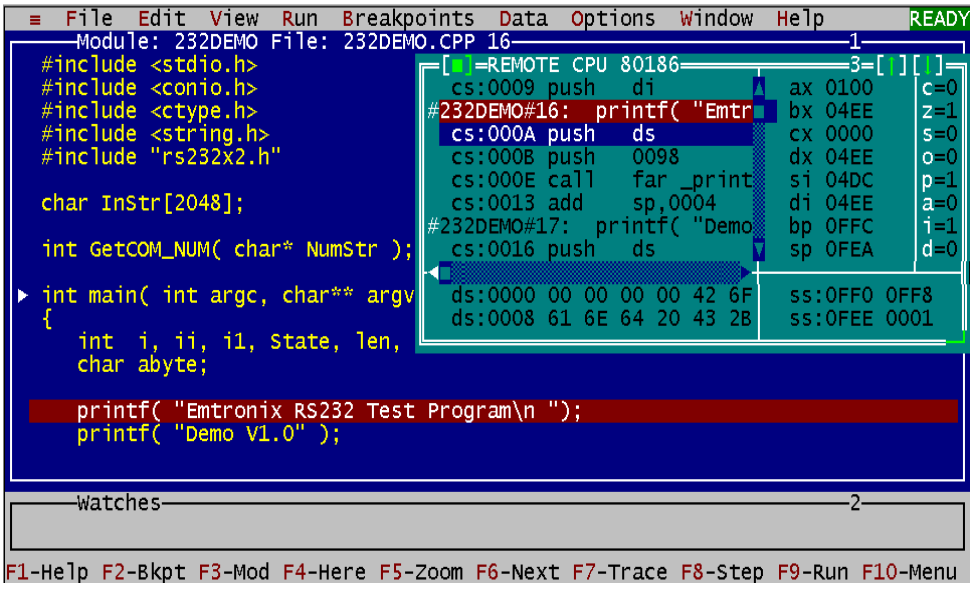
第 3 步 编写各软件模块，即常规的程序设计。在 BC 的集成环境下，用户即可采用 C/C++ 这样的高级语言设计程序，也可用 x86 的汇编来设计关键模块，如硬件驱动程序、中断服务程序等。

第 4 步 对编写好的程序进行编译（Compiler）、汇编（Assembler）、连接（Linker），若程序有错，集成环境将提示错误信息，用户可根据错误信息返回第 2 步进行修改，直至生成可执行文件（如 userapp.exe）。

第 5 步 运行调试程序，对第 4 步生成的用户应用程序进行调试，典型命令如下：

C:\Myapp>td –rp userapp

TD 具有自动下载应用程序至目标系统的功能。TD 是一个功能强大的源级程序调试工具，基本界面如下：




若调试发现错误，可方便退出调试，返回第 2 步进行修改，直至整个程序完整调试。

第 6 步 调试完成即意味着程序开发的结束，用户可把 ETR232i 设置成直接运行模式，即可让系统独立运行。

# 附录 3 TDRF 及 TD 调试工具使用说明

TDRF 是一个简易的基于 RS232 的远程文件访问工具软件。基本使用方法如下：

TDRF [选项] 命令 [参数]

其中的选项有：

－rp<#> 设置 HOST 的调试串行端口号，rp1=COM1；rp2=COM2。

－rs<#> 设置串行波特率，rs1=9600bps；rs2=19kbps；rs3=38kbps；rs4=115kbps。

缺省选项为：－rp1 －rs4，即 COM1，115k 波特率。

“命令”为类似 DOS 命令的英文单词或单个缩写字母，与 DOS 命令类似，跟在 dir、copy、del、copyfrom 命令后的参数可采用“*”表示任意文件名或扩展名。TDRF 的主要命令有：

| 命令 | 缩写 | 参数数目 | 描述 |
|---|---|---|---|
| Copy | T | 1 或 2 | 拷贝 HOST 文件至 ETR232i |
| Copyfrom | F | 1 或 2 | 拷贝 ETR232i 中的文件至 HOST |
| Del，Erase | E | 1 | 删除 ETR232i A 盘或 C 盘中的文件 |
| Dir | D | 0 或 1 | 显示 ETR232i 中的文件目录 |
| Ren | R | 2 | 重命名 ETR232i 中的文件 |
| Md | M | 1 | 在 ETR232i 中建立新目录 |
| Rd | K | 1 | 删除 ETR232i A 盘或 C 盘中已存在的目录 |
| Cd | C | 0 或 1 | 改变 ETR232i 的当前目录 |

TD 即 Turbo Debugger 是一个功能强大的源程序调试工具，在基于 ETR232i 的应用开发中，可采用 TD 作为基本的交叉调试工具。具体方法为：把 ETR232i 的 RS232 调试端口与用于软件开发的 PC 的一个 RS232 端口（COM1 或 COM2）相连，一般在编写应用程序的当前目录启动 TD，如：

C:\MyApp>TD –rp# Userapp

其中-rp#用于指定 PC 的调试端口，-rp1=COM1，也可简写成-rp；-rp2=COM2。

运行 TD 时，用户编写的应用程序（如 Userapp.exe）将首先被下载至 ETR232i 的当前目录，然后应用程序自动启动，进入调试状态，相应的用户 PC 的屏幕上会显示应用程序的 main 模块的源代码，如附录 5 中的图所示。这时就可进行各种程序调试了。