基于 WINCE 平台 C#编程要点之二

本文主要介绍在基于 Windows CE 平台的英创嵌入式主板下进行 C#（Microsoft Visual Stdio.Net 2005）应用程序开发时会常常用到的一些功能函数以及开发方法，这些方法适用于英创采用 WinCE 平台的所有型号嵌入式主板，包括 EM9000、EM9260、EM9160 等。

本文要点包括：

- 在英创 ARM9 嵌入式主板上实现 GPRS 拨号上网
- 使用 TcpClient 和 TcpListener 两个类实现网络数据传输

一、在英创 ARM9 嵌入式主板上实现 GPRS 拨号上网

在进行 GPRS 拨号之前，需要首先创建一个拨号连接，也就是需要添加一个 RASEntry 到 RAS 的电话薄中，并将其保存在注册表中。RASEntry 中需要设置作为调制解调器的设备属性（端口设置、拨号选项）以及指定电话号码和用户验证参数。在 EM9000 嵌入式主板上，直接支持 WinCE 桌面系统，用户可以通过界面中“控制面板”手动添加 GPRS 拨号连接。具体的设置方法请参见技术文档《EM9000 无线通讯网络设置》一文，这里就不再赘述。而 EM9260、EM9160 嵌入式主板没有支持 WinCE 的界面，所以英创公司为了方便用户的使用，英创专门设计了自动创建拨号连接的程序 RASEntry.exe，并捆绑在内核中，用户可以直接运行 windows>目录下的 RASEntry.exe，即可自动创建 GPRS 的拨号连接。该程序对于英创的所有的 ARM9 嵌入式主板均适用。注意 RASEntry.exe 只需运行一次即可。

对于 EM9000 标准板卡，GPRS 拨号属性中的端口配置为 COM3，波特率为 57600，8 位数据位，无校验，1 位停止位。EM9260、EM9160 中 GPRS 拨号属性中的端口配置为 COM2，波特率为 57600，8 位数据位，无校验，1 位停止位。

在英创提供的所有 ARM9 嵌入式主板开发套件中均可直接接上 MC39i 模块进行 GPRS 应用开发，这里主要介绍西门子公司的 MC39I 的上电过程。

MC39i 的电源管理是通过 1 路数字输入（GPRS_PWR）、2 路数字输出（GPRS_STB、GPRS_AUX）来实现。其中 GPRS_PWR 用于检测 MC39i 模块上电的状态，输入高表示无线模块已上电；GPRS_STB 主要用于对 MC39i 进行上电操作，通过 GPRS_STB 输出一个脉冲，使能 MC39i 上电；GPRS_AUX 主要用于对 MC39i 进行控制，输出一个高电平，

将关闭 MC39i 电源供电。

在程序设计中，为了确保每次 MC39i 模块上电操作正常，我们所提供的程序所采用的策略是先将模块关电，然后再进行正常的上电操作。

```
GPRS_PowerOff();          // 操作 GPRS_AUX 关闭 MC39i 模块电源
Thread.Sleep(400);                // 延时 400ms
GPRS_PowerOn();           // 操作 GPRS_STB 对 MC39i 模块进行上电操作
```

目前华为公司的 GTM900 无线模块，与 MC39i 在外形尺寸以及管脚都是一致的，因此在英创提供的 ARM9 嵌入式主板开发套件中也可以直接接上 GTM900 进行 GPRS 应用开发。GTM900 的电源管理也是利用的 GPRS_STB、GPRS_AUX，它和 MC39I 唯一的区别就在上电、关电的操作过程，为此英创公司对阵 GTM900 提供相应的 GPRS_PowerOff( )、GPRS_PowerOn( ) 函数，应用仍然按照相同调用顺序即可实现对 GTM900 的自动上电操作。

在使用 C#编程操作 GPRS 模块拨号之前，首先要明确：很多底层操作的函数（如 PPP 拨号函数），Visual Studio 2005.NET 的 API 库中并没有提供，这个时候，我们就要在 C#开发中调用 Win32 的函数来进行相应的操作。一大批 Win32 底层操作的函数都存在于 cordll.dll 动态链接库中。

调用 Win32 的申明：

```
using System.Runtime.InteropServices;
```

要使用的两个 Win32 函数申明如下：

```
#region Win32 API RASDial 函数声明
[DllImport("coredll.dll")]
public static extern uint RasDial(IntPtr dialExtensions, IntPtr phoneBookPath, IntPtr rasDialParam, uint NotifierType, IntPtr notifier, ref IntPtr pRasConn);
[DllImport("coredll.dll")]
public static extern uint RasHangUp(IntPtr pRasConn);
#endregion
```

拨号操作使用 RasDial(…)函数，挂断操作使用 RasHangUp(…)函数。尤其需要注意的是对 RasDial 函数里面使用的参数 rasDialParam 的赋值过程，即将拨号连接名称、用户名、密码等通过 rasDialParam 参数进行传递的过程，具体请参见英创 C#例程。

二、使用 TcpClient 和 TcpListener 两个类实现网络数据传输

客户在进行 TCP 编程的时候可以使用微软在 VS2005 中提供的两个类 TcpClient 和 TcpListener。

建立一个客户机连接：

```
TcpClient aClient = new TcpClient();
aClient.Connect("远程服务器IP地址", 服务器端口);
```

向服务器发送字符串数据：

```
byte[] OutBuffer;
string strSendMessage = "发送的内容";
OutBuffer = Encoding.Default.GetBytes(strSendMessage.ToCharArray());
aClient.GetStream().Write(OutBuffer, 0, OutBuffer.Length);
```

接受服务器发送的字符串数据，可以启动一个线程，在线程里设置如下循环：

```
for (; ; )
{
    if (aClient.GetStream().DataAvailable)
    {
        aClient.GetStream().Read(InBuffer, 0, InBuffer.Length);
        ReturnMessage += Encoding.Default.GetString(InBuffer, 0, InBuffer.Length);
        ……要做的其它处理……
    }
    if (threadStop == true)    //bool变量，以确定是否结束此线程
    {
        Thread.CurrentThread.Abort();
        Thread.CurrentThread.Join();
    }
}
```

设置一个服务器侦听：

```
IPAddress  LocalIP = Dns.GetHostEntry(Dns.GetHostName()).AddressList[0];
TcpListener aServer = new TcpListener(LocalIP, 侦听端口);
aServer.Start();
```

要确定远端是否有连接请求，可以设置一个 Timer 控件，在一定时间间隔内去检查远端是否有连接请求，如果有连接请求 aServer.Pending()则为 true。这时可以启动一个线程来接受客户端传来的数据，如：

```
int Bytes;
Socket CurSocket = aServer.AcceptSocket();
for(;;)
{
    if (CurSocket.Available > 0)
```

```
    {
        Bytes = CurSocket.Receive(InBuffer, InBuffer.Length, 0);
        ReturnMessage = ReturnMessage+Encoding.Default.GetString(InBuffer, 0, Bytes);
        ……要做的其它处理……
    }
}
```

向远程连接客户端发送字符串数据：

```
byte[] OutBuffer;
string strSendMessage = “发送的内容”;
OutBuffer = Encoding.Default.GetBytes(strSendMessage.ToCharArray());
CurSocket.Send(OutBuffer, OutBuffer.Length, 0);
```