基于 WINCE 平台 C#编程要点之一

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：

- 文件的删除和复制
- 如何获取存取设备的空间大小
- 如何重启系统
- 如何隐藏/显示 Windows 任务栏

一、文件的删除和复制

一般来说，C#在 WinCE 平台上进行文件的删除和复制有两种方法，下面我们将对这两种方法进行介绍。在介绍之前，对一种情况要特殊说明一下：关于文件的复制，大家都知道最简便的方法是将源文件直接复制到目标文件，如果目标文件事先存在，则直接覆盖；但在 WinCE 的平台上，由于采用 TFAT 文件系统，当进行文件覆盖时，要求系统剩余空间至少大于所需更新的文件大小，否则文件拷贝时，系统将报错。所以我们建议，如果目标文件事先存在，先删除目标文件，然后再将源文件复制过去。

两种方法的申明均是：

```
using System.IO;
```

1) 方法一：静态 File 方法（最简单的方法）

文件删除： `File.Delete(string path);`

文件复制： `File.Copy(string sourceFileName, string destFileName);`

2) 方法二：FileInfo 方法（最灵活的方法）

使用这种方法，每个具体的文件要定义一个 FileInfo 类，然后通过操作具体的类来进行文件删除、复制或其它操作。下面以一个具体范例来说明：

```
string sourceFileName, destFileName;
sourceFileName = @"\USB Storage\mysourcefile.bin";     //源文件名及路径
```

```
destFileName = @"\NandFlash\mydestfile.bin";              //目标文件名及路径
FileInfo sourceFile = new System.IO.FileInfo(sourceFileName);
FileInfo destFile = new System.IO.FileInfo(destFileName);
try
{
    if (destFile.Exists) destFile. Delete();      //如果目标文件已经存在，则删除
    sourceFile.CopyTo(destFileName,true);   //将源文件复制到目标文件
}
catch(Exception ex)
{
    MessageBox.Show(ex.Message);
}
```

二、获取存储设备的大小信息

很多底层操作的函数，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;
```

在 WinCE 下已经没有了驱动器名的概念，文件存储设备都是在 WinCE 的根目录下中以目录的形式出现，可以采用如下方法并调用相应函数来获取存储设备的总的大小和空闲空间的信息：

```
[DllImport("coredll.dll")]
private static extern bool GetDiskFreeSpaceEx(string directoryName, ref long freeBytesAvailable, ref long totalBytes, ref long totalFreeBytes);
```

调用例子如下：

```
long freeBytes = 0, totalBytes = 0, totalFreeBytes = 0;
GetDiskFreeSpaceEx("\\Nor Flash", ref freeBytes, ref totalBytes, ref totalFreeBytes);
```

string strtotalBytes = "Nor Flash 磁盘空间大小为："+totalBytes.ToString()+"Bytes";

三、重启系统函数

很多底层操作的函数，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;

调用“cordll.dll”里的 Win32 函数 SetCleanRebootFlag(…)和 KernelIoControl(…)可以实现 WINCE 系统重新启动，定义的代码如下：

```
[DllImport("Coredll.dll")]
extern static int KernelIoControl(int dwIoControlCode, IntPtr lpInBuf, int nInBufSize, IntPtr lpOutBuf, int nOutBufSize, ref int lpBytesReturned);
[DllImport("Coredll.dll")]
extern static void SetCleanRebootFlag();
```

可以写成一个系统重启的函数：

```
public void HardReset()
{
  int IOCTL_HAL_REBOOT = 0x101003C;
  int bytesReturned = 0;
  SetCleanRebootFlag();
  KernelIoControl(IOCTL_HAL_REBOOT,IntPtr.Zero,0, IntPtr.Zero, 0, ref bytesReturned);
}
```

然后在程序里需要重启的地方直接调用 HardReset()这个函数即可。

四、隐藏/显示 Windows 任务栏

很多底层操作的函数，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;
```

很多客户的应用程序需要独占屏幕，而不需要下方的 Windows 任务栏。因此显示/隐藏 Windows 任务栏是一个很实用的功能。这个时候就需要调用 coredll.dll 里的 FindWindow(…)和 ShowWindow(…)函数来实现任务栏的显示和隐藏（Windows 的任务栏实际上也是一个特殊的 Windows 窗口）。

```
[DllImport("coredll.dll", EntryPoint = "FindWindow")]
public static extern int FindWindow( string lpWindowName, string lpClassName );
[DllImport("coredll.dll", EntryPoint = "ShowWindow")]
public static extern int ShowWindow( int hwnd, int nCmdShow );
public const int SW_SHOW = 5;     //显示窗口常量
public const int SW_HIDE = 0;        //隐藏窗口常量
```

下面是一个隐藏/显示 Windows 任务栏的小例子：

```
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            int Hwnd = FindWindow("HHTaskBar", null);
            if (Hwnd != 0)
            {
                ShowWindow(Hwnd, SW_HIDE);     //隐藏任务栏
                button2.Enabled = true;
                button1.Enabled = false;
            }
        }
```

```
private void button2_Click(object sender, EventArgs e)
{
    int Hwnd = FindWindow("HHTaskBar", null);
    if (Hwnd != 0)
    {
        ShowWindow(Hwnd, SW_SHOW);    //显示任务栏
        button1.Enabled = true;
        button2.Enabled = false;
    }
}
```