正文：

在用LABVIEW进行数据采集分析过程中，一般会用到数据库。在数据量很大的场合，可考虑使用SQL SERVER。下面就介绍LABVIEW与SQL SERVER的连接方法。

连接思路：先在SQL SERVER建立自己的数据库，如MyDB。然后在操作系统建立一数据源（ODBC），此数据源的驱动选择SQL SERVER，数据源与SQL SERVER中的MyDB关联。最后在LABVIEW中使用ActiveX连接到数据源。本文假设用户已经建立一数据源LABVIEWconSQL，且数据源关联到数据库MyDB，包含表tempDemo，表格式样式如图1：



图1

下面介绍详细步骤：

1．连接connection。在后面板放置Invoke Node，右键选择属性类Select Calss/ActiveX/ADOB.\_Connection,然后选择其Messord为Open。创建Automation Open节点，再创建一constant，其Class也选择ADOB.\_Connection。如图2：

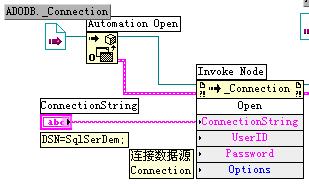


图2

2．在连接成功之后，创建并执行命令command。首先配置Command属性，在后面板放置Porperty Node，关联Class为ADOB.\_Command。然后选择ActiveConnection。全过程如图3：

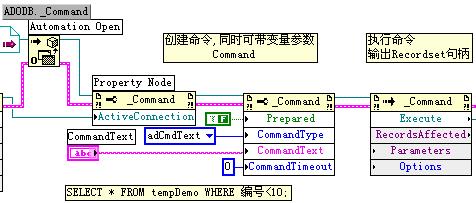


图3

3．最后对表的记录进行操作recordset。因程序较简单，此处不再详细说明，具体示例如图4：

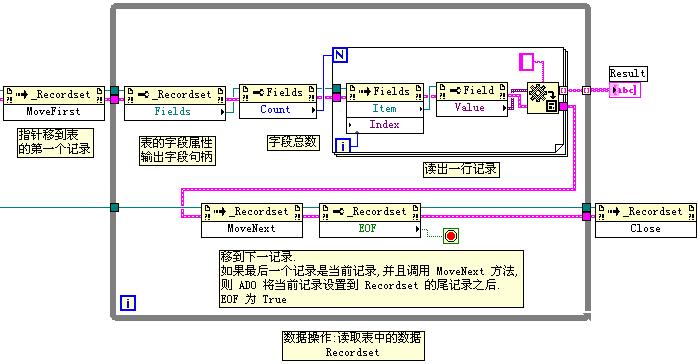
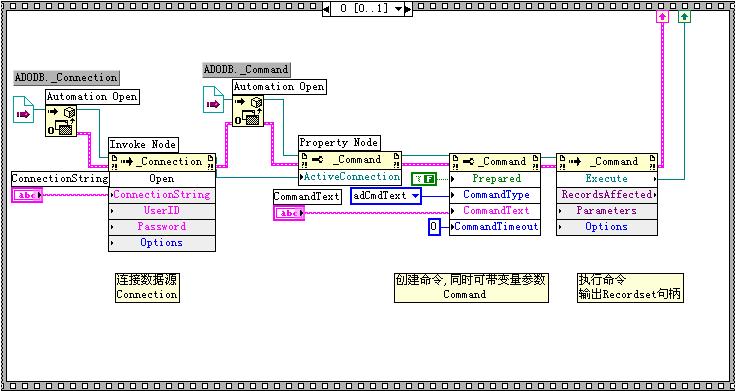


图4

所有程序如图5：



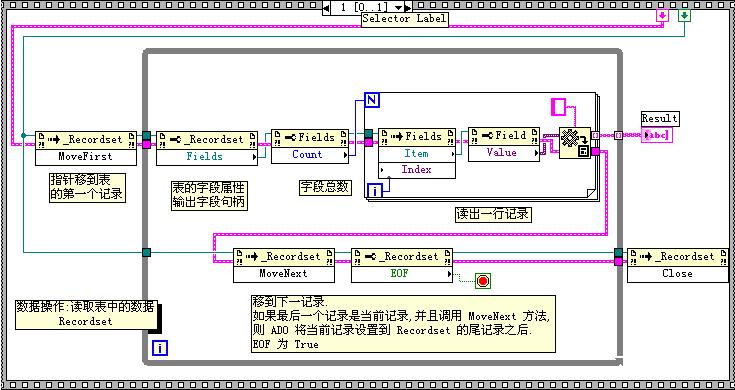


图5

程序运行后得到表的前9条记录，如图6所示：

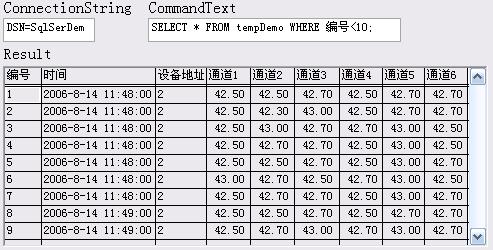


图6

总结：

本文所述的连接方法是使用LABVIEW最底层的ActiveX控件执行ADO API。看懂本文需理解LABVIEW的ActiveX运行机制及有关ADO的编程模型。

附基本的 ADO 编程模型：

· 连接到数据源 (Connection)，并可选择开始一个事务。

· 可选择创建对象来表示 SQL 命令 (Command)。

· 可选择在 SQL 命令中指定列、表和值作为变量参数 (Parameter)。

· 执行命令 (Command、Connection 或 Recordset)。

· 如果命令按行返回，则将行存储在缓存中 (Recordset)。

· 可选择创建缓存视图，以便能对数据进行排序、筛选和定位 (Recordset)。

· 通过添加、删除或更改行和列编辑数据 (Recordset)。

· 在适当情况下，使用缓存中的更改内容来更新数据源 (Recordset)。

· 如果使用了事务，则可以接受或拒绝在完成事务期间所作的更改。结束事务 (Connection)。

参考文献：

1． 程序员参考Microsoft Corporation. Microsoft ActiveX Data Objects (ADO)

2． National Instruments Corporation. LabVIEW Help

正文：

在用LABVIEW进行数据采集分析过程中，一般会用到数据库。在数据量很大的场合，可考虑使用SQL SERVER。下面就介绍LABVIEW与SQL SERVER的连接方法。

连接思路：先在SQL SERVER建立自己的数据库，如MyDB。然后在操作系统建立一数据源（ODBC），此数据源的驱动选择SQL SERVER，数据源与SQL SERVER中的MyDB关联。最后在LABVIEW中使用ActiveX连接到数据源。本文假设用户已经建立一数据源LABVIEWconSQL，且数据源关联到数据库MyDB，包含表tempDemo，表格式样式如图1：



图1

下面介绍详细步骤：

1．连接connection。在后面板放置Invoke Node，右键选择属性类Select Calss/ActiveX/ADOB.\_Connection,然后选择其Messord为Open。创建Automation Open节点，再创建一constant，其Class也选择ADOB.\_Connection。如图2：

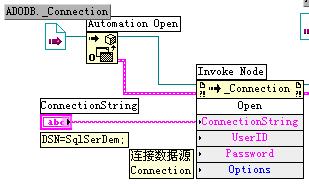


图2

2．在连接成功之后，创建并执行命令command。首先配置Command属性，在后面板放置Porperty Node，关联Class为ADOB.\_Command。然后选择ActiveConnection。全过程如图3：

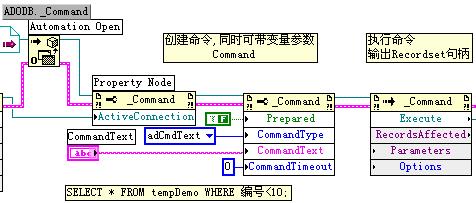


图3

3．最后对表的记录进行操作recordset。因程序较简单，此处不再详细说明，具体示例如图4：

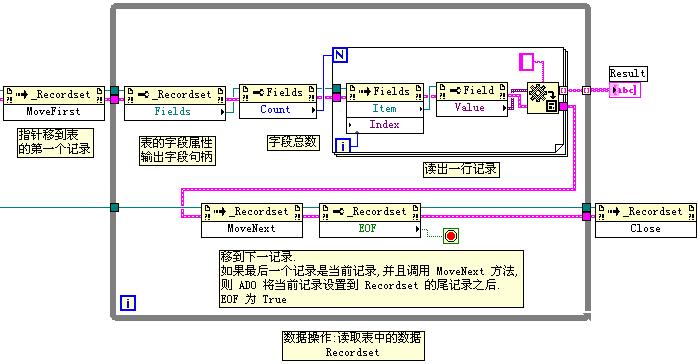
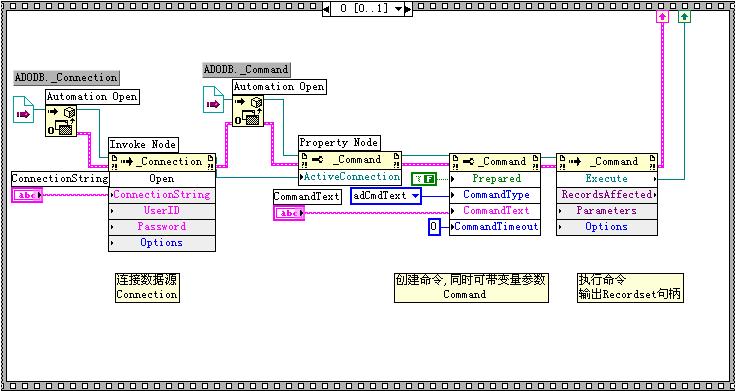


图4

所有程序如图5：



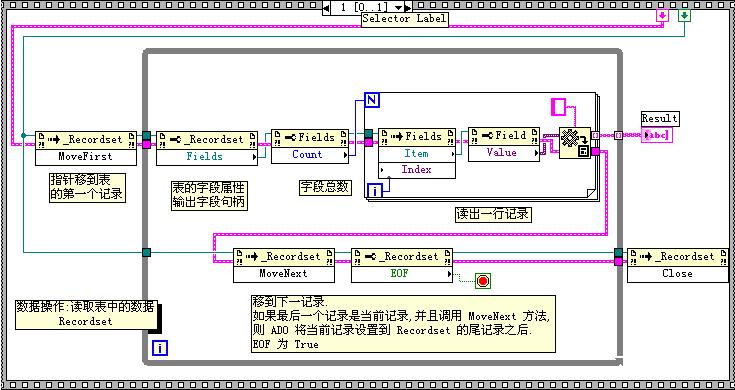


图5

程序运行后得到表的前9条记录，如图6所示：

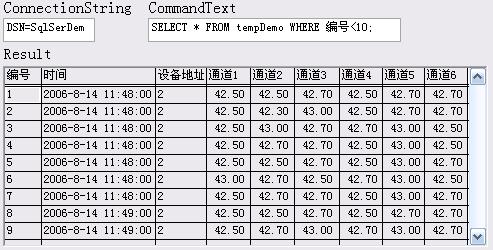


图6

总结：

本文所述的连接方法是使用LABVIEW最底层的ActiveX控件执行ADO API。看懂本文需理解LABVIEW的ActiveX运行机制及有关ADO的编程模型。

附基本的 ADO 编程模型：

· 连接到数据源 (Connection)，并可选择开始一个事务。

· 可选择创建对象来表示 SQL 命令 (Command)。

· 可选择在 SQL 命令中指定列、表和值作为变量参数 (Parameter)。

· 执行命令 (Command、Connection 或 Recordset)。

· 如果命令按行返回，则将行存储在缓存中 (Recordset)。

· 可选择创建缓存视图，以便能对数据进行排序、筛选和定位 (Recordset)。

· 通过添加、删除或更改行和列编辑数据 (Recordset)。

· 在适当情况下，使用缓存中的更改内容来更新数据源 (Recordset)。

· 如果使用了事务，则可以接受或拒绝在完成事务期间所作的更改。结束事务 (Connection

ODBC连接  
& E2 t2 n5 \! [/ o适合数据库类型 连接方式9 x4 [9 g- t4 t# \\* Q: Z  
access  "Driver={microsoft access driver(\*.mdb)};dbq=\*.mdb;uid=admin;pwd=pass;"  
9 e# k/ y4 L1 h3 x' n0 \_dBase  "Driver={microsoft dbase driver(\*.dbf)};driverid=277;dbq=------------;"  
, P& ]# P% }5 y- ]3 Moracle  "Driver={microsoft odbc for oracle};server=oraclesever.world;uid=admin;pwd=pass;"3 d  ]3 A" [4 ?# ~& g- ~  
MSSQL server  "Driver={**sql** server};server=servername;database=dbname;uid=sa;pwd=pass;"  
7 s9 h\* D0 E$ K, b+ }. kProvider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=%DBNAME%;Data Source=%SERVER%  
# l9 f1 i2 c9 f" Y, G7 b2 I- L% rMS text  "Driver={microsoft text driver(\*.txt; \*.csv)};dbq=-----;extensions=asc,csv,tab,txt;Persist SecurityInfo=false;"  
5 v. H6 R: {/ D3 o7 a3 |Visual Foxpro  "Driver={microsoft Visual Foxpro driver};sourcetype=DBC;sourceDB=\*.dbc;Exclusive=No;"  
6 J8 |" ^4 K! W8 S4 kMySQL  "Driver={my**sql**};database=yourdatabase;uid=username;pwd=yourpassword;option=16386;"4 c1 Y% }! ^, C- u/ R  
SQLite  "Driver={SQLite3 ODBC Driver};Database=D:\SQLite\\*.db"2 j; d8 T; B5 e8 D+ \& i' ?- x  
PostgreSQL  "Driver={PostgreSQL ANSI};server=127.0.0.1;uid=admin;pwd=pass;database=databaseName"" f" z5 m( q! ]5 t8 P! b4 \_/ K  
OLEDB连接  S% Y' ~: h2 j7 g& C$ k  
适合的数据库类型 连接方式  
1 u9 j) M" p4 Z. M  a- R1 @- ~7 u1 {$ Eaccess  "Provider=microsoft.jet.oledb.4.0;data source=your\_database\_path;user id=admin;password=pass;"  
7 J/ K% D8 J! y" D1 Y/ Joracle  "Provider=OraOLEDB.Oracle;data source=dbname;user id=admin;password=pass;"  
; @$ I+ D4 ^7 SMS SQL Server  "Provider=SQLOLEDB;data source=machinename;initial catalog=dbname;userid=sa;password=pass;"  
. g7 A! A; D( y- |\* WMS text  "Provider=microsof.jet.oledb.4.0;data source=your\_path;Extended Properties'text;FMT=Delimited'"   
9 `3 y8 N! t3 h补充: 如果你用的是SQL2008专业版, 在数据库名字后面加上\SQLEXPRESS试试.

在LabVIEW中设置菜单

一、普通菜单  
如果VI的窗口外观设置中选择了“Show menu bar”，VI在运行时会在前面板窗口中显示菜单条，默认的菜单是LabVIEW的标准菜单，为了适应使用的需要，我们可以定制个性化的运行菜单，并在程序中对用户的菜单作出响应。定制菜单可以在编辑程序时通过对话框进行设置，也可以在程序运行过程中通过编程实现。

使用菜单命令“Edit —Run —Time Menu”弹出菜单编辑器如图所示。

菜单编辑器打开时工具条的Item Type 下拉列表中显示的是Default选项，表示当前菜单是LabVIEW的标准菜单，编辑器中的其它各项功能不可使用。下拉列表中的Minimal选项，表示使用LabVIEW标准菜单中的常用选项。如果需要定制VI运行菜单则要选择Custom选项。   
菜单编辑器工具条的6个按钮用来在菜单项列表中编辑菜单项，它们是   
Insert 在选定的菜单项后插入新的菜单项    
Delete  删除选定的菜单项   
Left 使选定的菜单项成为上一级菜单项    
Right 使选定的菜单项成为上一级菜单项的子菜单   
Up  向上移动选定的菜单项，如果包含子菜单，也一起移动   
Down  向下移动选定的菜单项，如果包含子菜单，也一起移动

菜单预览区可以看到运行时的菜单条。  
菜单项列表显示菜单的层次结构，用来对菜单进行编辑。  
菜单项属性编辑区逐个设置菜单项的属性，包括：  
Item Type 定义菜单的类型    
     User Item 用户定制菜单项，需要编辑，并需要编程响应   
Seperator 在菜单项中插入分隔符   
Application Item 从LabVIEW标准菜单中选择菜单项加入到运行菜单中 ，无需编辑和编程响应

Item Name  菜单名，出现在菜单条上的字符串    
Item Tag 菜单标识，程序通过菜单标识引用菜单选项    
Enabled 指定该菜单项是否有效    
Checked 菜单项是否有选中标志    
Shortcut 设置访问菜单项的快捷键

运行菜单设置完成后要将其保存在与Vi相同的位置（同一个文件路径或同一个LLB），名称后缀为..rtm。

二、右键菜单  
在其它开发环境中很容易实现右键菜单功能，在LabVIEW中相对就要复杂一些，只能通过调用DLL来实现！

|  |  |  |  |
| --- | --- | --- | --- |
| **Re:树形列表和右键弹出菜单**   |  | | --- | | subpanel退出  如下图运行的程序主面板，如何在其中退出？？ 作llb时最好设置一个主VI弹出执行  tools>>Edit VI Library  选择\*.llb 进入编辑llb对话框，选中相应的\*.vi把Top Level Check box选中即可。  我是用**菜单**控制的啊！你看到**菜单**了么？  请点击**菜单**里的数据采集---->建立标定文件  就可以看到在subpanel里面弹出的窗口，问题怎么关闭它呢  我点击**菜单**中的退出按钮来退出整个程序好像不行，大侠看看有什么方法可以啊？"这个是可以的，不过你的那个常量“false constant”应改为“true constant”。 那个subpanel 的“退出”好像也可以，因为点击那个“退出”之后，再点击“新标定系数”或者“取标定系数”、“保存标定系数”都无效了，而不点击那个“退出”则可以，说明那个“退出”点击之后还是起作用了，问题是subpanel还看得见，有点像没有实现退出的味道。再说，那个subpanel总是得有东西在哪里才好，譬如还有其他的subpanel vi，当点击**菜单**里其他选项时就调出来了，因此我觉得“没有实现退出的味道”没有关系。  这样理解不知道合不合理。  此外，使用subpanel，在还没有调用子vi时那个subpanel里什么都没有，有点不美观的感觉，可不可以先像其他控件一样有默认的东东在里面呢 谢谢你的答复！你说的也可以实现“退出”，尽管subpanel里面的东西还有。但是总感觉不想ＶＣ或Ｃ＋＋ｂｕｉｌｄｅｒ里面的子窗口那样好，关掉子窗口就剩下主窗口，主窗口里面很干净。看来ＬＶ这方面还需要改进阿！  你说的“在还没有调用子vi时那个subpanel里什么都没有，有点不美观的感觉，可不可以先像其他控件一样有默认的东东在里面呢？”，可以右键点击ｓｕｂｐａｎｅｌ取消显示ｆｒａｍｅ就可以了。  其实我觉得用弹出窗口的方式一般是对话框形式的，就是说不能同时操作主界面的**菜单**之类的东西。而ｓｕｂｐａｎｅｌ则不然。”还是得留一块地方（空白的，供subpanel用，别的控件不能占用）吧？这样的话，是不是用弹出窗口的方式要好？ “，这个要根据你的具体情况了，如果主面板很多空件的话不宜用ｓｕｂｐａｎｅｌ了。  有没有可能在弹出窗口以后（还没有关闭窗口时），还可以对主窗口实现操作（包括主窗口的**菜单**）？可以的。只是要记得在设置vi properties 时的时候，在windows appearance里面　customize选项中的，windows behavior里选择default不要选modal（这个不允许在打开对话框的时候再去操作主界面的东东)。  我是想退出subvi啊！停留在主程序里。如果直接在主vi中调用subvi，也就是让subvi以对话框的形式弹出，这样点击subvi的退出是没有问题的！但问题是如果用了subpanel之后，好像退出按钮失去作用了！改怎么办？  另外：我点击**菜单**中的退出按钮来退出整个程序好像不行，大侠看看有什么方法可以啊？  现在还没有解决!! 我想一个关键是 :子VI 的参数能否传递到 主VI中,按常规的方法是不行的,就像函数形参不能传递到主函数中去一样!  我下图点了子VI的退出之后,subpanel里的东西仍旧停留在那里 , 按理论可以用subpanel的remove 节点移出subpanel的子vi 但是怎么一按退出按钮就执行remove 事件呢! 退出->remove 怎么组合在一起是个难题!!  用全局变量试试吧！这个demo可以吗？问题的关键是两个VI直接的通信，用vi reference应该也可以，但可能比较麻烦！用全局变量比较简单！  我总的目的想做一个界面比较清晰的程序，但是可以根据任务的需要随时在进行数据采集的同时做其他工作，比如监控一下波形，做一下有关计算之类，其实就是有点类似VC做的多文档应用程序。数据的传送我现在想用shife register还可以做到。第一种办法是弹出式**菜单**，但在lavbiew里面两个弹出式**菜单**不能同时显示，无论是重叠还是平铺，只能看见一个(default\model\floating三种我都试了)；第二种办法就是你说的，我也还在看，但是这样的话不太能满足我的要求，因为随时改变subplane就要在前面板设计时摆好多面板，不如放在子VI中。这是我的一点想法，希望继续交流。  有人能告诉我这个例子中的toolbar是怎么作出来的? 用的是ActiveX控件 MS toolbar  是在function palette>comm>activex吗?我没有找到?方便的话能说明一下这个怎么用吗?我现在做的是有好多个界面组成的程序,这方面知道的太少,希望能得到大家的帮助 选择control--->containers--->ActiveX containers放置在front panel，点击右键选择insert ActiveX Object，在对话框中选择本机已注册的控件！同时会在block panel中得到其refnum。    程序为  [[http://www.vihome.com.cn/bbs/images/affix.gif](http://www.vihome.com.cn/bbs/attachments/bbsxp/2006-12/20061226185128.rar)20065283193961.rar](http://www.vihome.com.cn/bbs/attachments/bbsxp/2006-12/20061226185128.rar) | | | |
|  | | |
|  | | | [点评](http://www.vihome.com.cn/bbs/forum.php?mod=misc&action=comment&tid=3212&pid=168&extra=&page=1)[回复](http://www.vihome.com.cn/bbs/forum.php?mod=post&action=reply&fid=165&tid=3212&reppost=168&extra=&page=1) [引用](http://www.vihome.com.cn/bbs/forum.php?mod=post&action=reply&fid=165&tid=3212&repquote=168&extra=&page=1)  [评分](javascript:;) [举报](javascript:;) [返回顶部](javascript:;) |
|  | | |  |
| [**虚拟小强**](http://www.vihome.com.cn/bbs/home.php?mod=space&uid=1)   * [发短消息](http://www.vihome.com.cn/bbs/home.php?mod=spacecp&ac=pm&op=showmsg&handlekey=showmsg_1&touid=1&pmid=0&daterange=2&pid=169) * [加为好友](http://www.vihome.com.cn/bbs/home.php?mod=spacecp&ac=friend&op=add&uid=1&handlekey=addfriendhk_1)   [**虚拟小强**](http://www.vihome.com.cn/bbs/home.php?mod=space&uid=1) 当前离线  阅读权限  200  UID  1  [QQ](http://wpa.qq.com/msgrd?V=1&Uin=1005452066&Site=测量与测试技术&Menu=yes)[查看个人网站](http://www.vihome.com.cn/)[查看详细资料](http://www.vihome.com.cn/bbs/home.php?mod=space&uid=1&do=profile)  [http://www.vihome.com.cn/center/avatar.php?uid=1&size=middle](http://www.vihome.com.cn/bbs/home.php?mod=space&uid=1)  [管理员](http://www.vihome.com.cn/bbs/home.php?mod=spacecp&ac=usergroup&gid=1)  站长  Rank: 9Rank: 9Rank: 9   TA的每日心情   |  |  | | --- | --- | | http://www.vihome.com.cn/bbs/source/plugin/dsu_paulsign/img/kx.gif | 开心 2011-4-29 13:41 |   签到天数: 17 天  [LV.4]偶尔看看III  最后登录  2011-5-7  积分  1215119  帖子  [3539](http://www.vihome.com.cn/bbs/home.php?mod=space&uid=1&do=thread&type=reply&view=me&from=space)  Medal No.1Medal No.2Medal No.3Medal No.4 | [3楼](http://www.vihome.com.cn/bbs/forum.php?mod=redirect&goto=findpost&ptid=3212&pid=169&fromuid=92632)  http://www.vihome.com.cn/bbs/static/image/common/online_admin.gif发表于 2006-12-27 02:54 |[只看该作者](http://www.vihome.com.cn/bbs/forum.php?mod=viewthread&tid=3212&page=1&authorid=1)  [专业测量仪器 嵌入式开发板 数据采集卡电子商城](http://www.360mcu.com/)  **Re:树形列表和右键弹出菜单**   |  | | --- | | 固定前面板上的元件  group & lock 选项  方格背景怎样设置，方格大小能变吗？透明的图形指示器怎样设。  tools/options/alignment grid 可以有很多选择  在option中找一下，如果你要对背景更换颜色，只需要用Tools Pallete中的Set color就可以了  能否让字符串数组中的某一个元素闪光（blinking),可否通过改变字符串数组属性实现 在我们的应用当中，每一次循环，某一列的各个cell的blinking属性都有可能改变，因此每循环一次就要修改１次．采用您修改blinking属性的方法后，循环１次的时间是３０ms左右．  [[http://www.vihome.com.cn/bbs/images/affix.gif](http://www.vihome.com.cn/bbs/attachments/bbsxp/2006-12/20061226185432.zip)20065282212761.zip](http://www.vihome.com.cn/bbs/attachments/bbsxp/2006-12/20061226185432.zip)  最近在看LABVIEW里面的例程,在Analyzing and Processing Signals->Curve Fitting and Interpolation->Financial Forecasting.vi里面的Sales Forecast图表的横坐标,我看是直接用string indicator写的,而我们通常用string indicator写的字符串是有背景的,它是怎么做到背景是透明的呢? 在Classic Controls里面有Simple String，可以使用画笔工具Set Color设置前景色和背景色为透明！ See the color panel's top right corner, there are "T" remark. click it | | |
|

以下代码可以取得系统特征码（网卡MAC、硬盘序列号、CPU ID、BIOS编号）

BYTE szSystemInfo[4096]; // 在程序执行完毕后，此处存储取得的系统特征码

UINT uSystemInfoLen = 0; // 在程序执行完毕后，此处存储取得的系统特征码的长度

// 网卡 MAC 地址，注意: MAC 地址是可以在注册表中修改的

{

UINT uErrorCode = 0;

IP\_ADAPTER\_INFO iai;

ULONG uSize = 0;

DWORD dwResult = GetAdaptersInfo( &iai, &uSize );

if( dwResult == ERROR\_BUFFER\_OVERFLOW )

{

IP\_ADAPTER\_INFO\* piai = ( IP\_ADAPTER\_INFO\* )HeapAlloc( GetProcessHeap( ), 0, uSize );

if( piai != NULL )

{

dwResult = GetAdaptersInfo( piai, &uSize );

if( ERROR\_SUCCESS == dwResult )

{

IP\_ADAPTER\_INFO\* piai2 = piai;

while( piai2 != NULL && ( uSystemInfoLen + piai2->AddressLength ) < 4096U )

{

CopyMemory( szSystemInfo + uSystemInfoLen, piai2->Address, piai2->AddressLength );

uSystemInfoLen += piai2->AddressLength;

piai2 = piai2->Next;

}

}

else

{

uErrorCode = 0xF0000000U + dwResult;

}

VERIFY( HeapFree( GetProcessHeap( ), 0, piai ) );

}

else

{

return FALSE;

}

}

else

{

uErrorCode = 0xE0000000U + dwResult;

}

if( uErrorCode != 0U )

{

return FALSE;

}

}

// 硬盘序列号，注意：有的硬盘没有序列号

{

OSVERSIONINFO ovi = { 0 };

ovi.dwOSVersionInfoSize = sizeof( OSVERSIONINFO );

GetVersionEx( &ovi );

if( ovi.dwPlatformId != VER\_PLATFORM\_WIN32\_NT )

{

// Only Windows 2000, Windows XP, Windows Server 2003...

return FALSE;

}

else

{

if( !WinNTHDSerialNumAsPhysicalRead( szSystemInfo, &uSystemInfoLen, 1024 ) )

{

WinNTHDSerialNumAsScsiRead( szSystemInfo, &uSystemInfoLen, 1024 );

}

}

}

// CPU ID

{

BOOL bException = FALSE;

BYTE szCpu[16] = { 0 };

UINT uCpuID = 0U;

\_\_try

{

\_asm

{

mov eax, 0

cpuid

mov dword ptr szCpu[0], ebx

mov dword ptr szCpu[4], edx

mov dword ptr szCpu[8], ecx

mov eax, 1

cpuid

mov uCpuID, edx

}

}

\_\_except( EXCEPTION\_EXECUTE\_HANDLER )

{

bException = TRUE;

}

if( !bException )

{

CopyMemory( szSystemInfo + uSystemInfoLen, &uCpuID, sizeof( UINT ) );

uSystemInfoLen += sizeof( UINT );

uCpuID = strlen( ( char\* )szCpu );

CopyMemory( szSystemInfo + uSystemInfoLen, szCpu, uCpuID );

uSystemInfoLen += uCpuID;

}

}

// BIOS 编号，支持 AMI, AWARD, PHOENIX

{

SIZE\_T ssize;

LARGE\_INTEGER so;

so.LowPart=0x000f0000;

so.HighPart=0x00000000;

ssize=0xffff;

wchar\_t strPH[30]=L\\device\\physicalmemory;

DWORD ba=0;

UNICODE\_STRING struniph;

struniph.Buffer=strPH;

struniph.Length=0x2c;

struniph.MaximumLength =0x2e;

OBJECT\_ATTRIBUTES obj\_ar;

obj\_ar.Attributes =64;

obj\_ar.Length =24;

obj\_ar.ObjectName=&struniph;

obj\_ar.RootDirectory=0;

obj\_ar.SecurityDescriptor=0;

obj\_ar.SecurityQualityOfService =0;

HMODULE hinstLib = LoadLibrary("ntdll.dll");

ZWOS ZWopenS=(ZWOS)GetProcAddress(hinstLib,"ZwOpenSection");

ZWMV ZWmapV=(ZWMV)GetProcAddress(hinstLib,"ZwMapViewOfSection");

ZWUMV ZWunmapV=(ZWUMV)GetProcAddress(hinstLib,"ZwUnmapViewOfSection");

//调用函数，对物理内存进行映射

HANDLE hSection;

if( 0 == ZWopenS(&hSection,4,&obj\_ar) &&

0 == ZWmapV(

( HANDLE )hSection, //打开Section时得到的句柄

( HANDLE )0xFFFFFFFF, //将要映射进程的句柄，

&ba, //映射的基址

0,

0xFFFF, //分配的大小

&so, //物理内存的地址

&ssize, //指向读取内存块大小的指针

1, //子进程的可继承性设定

0, //分配类型

2 //保护类型

) )

//执行后会在当前进程的空间开辟一段64k的空间，并把f000:0000到f000:ffff处的内容映射到这里

//映射的基址由ba返回,如果映射不再有用,应该用ZwUnmapViewOfSection断开映射

{

BYTE\* pBiosSerial = ( BYTE\* )ba;

UINT uBiosSerialLen = FindAwardBios( &pBiosSerial );

if( uBiosSerialLen == 0U )

{

uBiosSerialLen = FindAmiBios( &pBiosSerial );

if( uBiosSerialLen == 0U )

{

uBiosSerialLen = FindPhoenixBios( &pBiosSerial );

}

}

if( uBiosSerialLen != 0U )

{

CopyMemory( szSystemInfo + uSystemInfoLen, pBiosSerial, uBiosSerialLen );

uSystemInfoLen += uBiosSerialLen;

}

ZWunmapV( ( HANDLE )0xFFFFFFFF, ( void\* )ba );

}

}

// 完毕， 系统特征码已取得。

以下是其中用到的某些结构及函数的定义

#define FILE\_DEVICE\_SCSI 0x0000001b

#define IOCTL\_SCSI\_MINIPORT\_IDENTIFY ( ( FILE\_DEVICE\_SCSI << 16 ) + 0x0501 )

#define IOCTL\_SCSI\_MINIPORT 0x0004D008 // see NTDDSCSI.H for definition

#define IDENTIFY\_BUFFER\_SIZE 512

#define SENDIDLENGTH ( sizeof( SENDCMDOUTPARAMS ) + IDENTIFY\_BUFFER\_SIZE )

#define IDE\_ATAPI\_IDENTIFY 0xA1 // Returns ID sector for ATAPI.

#define IDE\_ATA\_IDENTIFY 0xEC // Returns ID sector for ATA.

#define DFP\_RECEIVE\_DRIVE\_DATA 0x0007c088

typedef struct \_IDSECTOR

{

USHORT wGenConfig;

USHORT wNumCyls;

USHORT wReserved;

USHORT wNumHeads;

USHORT wBytesPerTrack;

USHORT wBytesPerSector;

USHORT wSectorsPerTrack;

USHORT wVendorUnique[3];

CHAR sSerialNumber[20];

USHORT wBufferType;

USHORT wBufferSize;

USHORT wECCSize;

CHAR sFirmwareRev[8];

CHAR sModelNumber[40];

USHORT wMoreVendorUnique;

USHORT wDoubleWordIO;

USHORT wCapabilities;

USHORT wReserved1;

USHORT wPIOTiming;

USHORT wDMATiming;

USHORT wBS;

USHORT wNumCurrentCyls;

USHORT wNumCurrentHeads;

USHORT wNumCurrentSectorsPerTrack;

ULONG ulCurrentSectorCapacity;

USHORT wMultSectorStuff;

ULONG ulTotalAddressableSectors;

USHORT wSingleWordDMA;

USHORT wMultiWordDMA;

BYTE bReserved[128];

} IDSECTOR, \*PIDSECTOR;

typedef struct \_DRIVERSTATUS

{

BYTE bDriverError; // Error code from driver, or 0 if no error.

BYTE bIDEStatus; // Contents of IDE Error register.

// Only valid when bDriverError is SMART\_IDE\_ERROR.

BYTE bReserved[2]; // Reserved for future expansion.

DWORD dwReserved[2]; // Reserved for future expansion.

} DRIVERSTATUS, \*PDRIVERSTATUS, \*LPDRIVERSTATUS;

typedef struct \_SENDCMDOUTPARAMS

{

DWORD cBufferSize; // Size of bBuffer in bytes

DRIVERSTATUS DriverStatus; // Driver status structure.

BYTE bBuffer[1]; // Buffer of arbitrary length in which to store the data read from the // drive.

} SENDCMDOUTPARAMS, \*PSENDCMDOUTPARAMS, \*LPSENDCMDOUTPARAMS;

typedef struct \_SRB\_IO\_CONTROL

{

ULONG HeaderLength;

UCHAR Signature[8];

ULONG Timeout;

ULONG ControlCode;

ULONG ReturnCode;

ULONG Length;

} SRB\_IO\_CONTROL, \*PSRB\_IO\_CONTROL;

typedef struct \_IDEREGS

{

BYTE bFeaturesReg; // Used for specifying SMART "commands".

BYTE bSectorCountReg; // IDE sector count register

BYTE bSectorNumberReg; // IDE sector number register

BYTE bCylLowReg; // IDE low order cylinder value

BYTE bCylHighReg; // IDE high order cylinder value

BYTE bDriveHeadReg; // IDE drive/head register

BYTE bCommandReg; // Actual IDE command.

BYTE bReserved; // reserved for future use. Must be zero.

} IDEREGS, \*PIDEREGS, \*LPIDEREGS;

typedef struct \_SENDCMDINPARAMS

{

DWORD cBufferSize; // Buffer size in bytes

IDEREGS irDriveRegs; // Structure with drive register values.

BYTE bDriveNumber; // Physical drive number to send

// command to (0,1,2,3).

BYTE bReserved[3]; // Reserved for future expansion.

DWORD dwReserved[4]; // For future use.

BYTE bBuffer[1]; // Input buffer.

} SENDCMDINPARAMS, \*PSENDCMDINPARAMS, \*LPSENDCMDINPARAMS;

typedef struct \_GETVERSIONOUTPARAMS

{

BYTE bVersion; // Binary driver version.

BYTE bRevision; // Binary driver revision.

BYTE bReserved; // Not used.

BYTE bIDEDeviceMap; // Bit map of IDE devices.

DWORD fCapabilities; // Bit mask of driver capabilities.

DWORD dwReserved[4]; // For future use.

} GETVERSIONOUTPARAMS, \*PGETVERSIONOUTPARAMS, \*LPGETVERSIONOUTPARAMS;

//////////////////////////////////////////////////////////////////////

//结构定义

typedef struct \_UNICODE\_STRING

{

USHORT Length;//长度

USHORT MaximumLength;//最大长度

PWSTR Buffer;//缓存指针

} UNICODE\_STRING,\*PUNICODE\_STRING;

typedef struct \_OBJECT\_ATTRIBUTES

{

ULONG Length;//长度 18h

HANDLE RootDirectory;// 00000000

PUNICODE\_STRING ObjectName;//指向对象名的指针

ULONG Attributes;//对象属性00000040h

PVOID SecurityDescriptor; // Points to type SECURITY\_DESCRIPTOR，0

PVOID SecurityQualityOfService; // Points to type SECURITY\_QUALITY\_OF\_SERVICE，0

} OBJECT\_ATTRIBUTES;

typedef OBJECT\_ATTRIBUTES \*POBJECT\_ATTRIBUTES;

//函数指针变量类型

typedef DWORD (\_\_stdcall \*ZWOS )( PHANDLE,ACCESS\_MASK,POBJECT\_ATTRIBUTES);

typedef DWORD (\_\_stdcall \*ZWMV )( HANDLE,HANDLE,PVOID,ULONG,ULONG,PLARGE\_INTEGER,PSIZE\_T,DWORD,ULONG,ULONG);

typedef DWORD (\_\_stdcall \*ZWUMV )( HANDLE,PVOID);

BOOL WinNTHDSerialNumAsScsiRead( BYTE\* dwSerial, UINT\* puSerialLen, UINT uMaxSerialLen )

{

BOOL bInfoLoaded = FALSE;

for( int iController = 0; iController < 2; ++ iController )

{

HANDLE hScsiDriveIOCTL = 0;

char szDriveName[256];

// Try to get a handle to PhysicalDrive IOCTL, report failure

// and exit if can't.

sprintf( szDriveName, "\\\\.\\Scsi%d:", iController );

// Windows NT, Windows 2000, any rights should do

hScsiDriveIOCTL = CreateFile( szDriveName,

GENERIC\_READ | GENERIC\_WRITE,

FILE\_SHARE\_READ | FILE\_SHARE\_WRITE, NULL,

OPEN\_EXISTING, 0, NULL);

// if (hScsiDriveIOCTL == INVALID\_HANDLE\_VALUE)

// printf ("Unable to open SCSI controller %d, error code: 0x%lX\n",

// controller, GetLastError ());

if( hScsiDriveIOCTL != INVALID\_HANDLE\_VALUE )

{

int iDrive = 0;

for( iDrive = 0; iDrive < 2; ++ iDrive )

{

char szBuffer[sizeof( SRB\_IO\_CONTROL ) + SENDIDLENGTH] = { 0 };

SRB\_IO\_CONTROL\* p = ( SRB\_IO\_CONTROL\* )szBuffer;

SENDCMDINPARAMS\* pin = ( SENDCMDINPARAMS\* )( szBuffer + sizeof( SRB\_IO\_CONTROL ) );

DWORD dwResult;

p->HeaderLength = sizeof( SRB\_IO\_CONTROL );

p->Timeout = 10000;

p->Length = SENDIDLENGTH;

p->ControlCode = IOCTL\_SCSI\_MINIPORT\_IDENTIFY;

strncpy( ( char\* )p->Signature, "SCSIDISK", 8 );

pin->irDriveRegs.bCommandReg = IDE\_ATA\_IDENTIFY;

pin->bDriveNumber = iDrive;

if( DeviceIoControl( hScsiDriveIOCTL, IOCTL\_SCSI\_MINIPORT,

szBuffer,

sizeof( SRB\_IO\_CONTROL ) + sizeof( SENDCMDINPARAMS ) - 1,

szBuffer,

sizeof( SRB\_IO\_CONTROL ) + SENDIDLENGTH,

&dwResult, NULL ) )

{

SENDCMDOUTPARAMS\* pOut = ( SENDCMDOUTPARAMS\* )( szBuffer + sizeof( SRB\_IO\_CONTROL ) );

IDSECTOR\* pId = ( IDSECTOR\* )( pOut->bBuffer );

if( pId->sModelNumber[0] )

{

if( \* puSerialLen + 20U <= uMaxSerialLen )

{

// 序列号

CopyMemory( dwSerial + \* puSerialLen, ( ( USHORT\* )pId ) + 10, 20 );

// Cut off the trailing blanks

for( UINT i = 20; i != 0U && ' ' == dwSerial[\* puSerialLen + i - 1]; -- i )

{}

\* puSerialLen += i;

// 型号

CopyMemory( dwSerial + \* puSerialLen, ( ( USHORT\* )pId ) + 27, 40 );

// Cut off the trailing blanks

for( i = 40; i != 0U && ' ' == dwSerial[\* puSerialLen + i - 1]; -- i )

{}

\* puSerialLen += i;

bInfoLoaded = TRUE;

}

else

{

::CloseHandle( hScsiDriveIOCTL );

return bInfoLoaded;

}

}

}

}

::CloseHandle( hScsiDriveIOCTL );

}

}

return bInfoLoaded;

}

BOOL DoIdentify( HANDLE hPhysicalDriveIOCTL, PSENDCMDINPARAMS pSCIP,

PSENDCMDOUTPARAMS pSCOP, BYTE bIDCmd, BYTE bDriveNum,

PDWORD lpcbBytesReturned )

{

// Set up data structures for IDENTIFY command.

pSCIP->cBufferSize = IDENTIFY\_BUFFER\_SIZE;

pSCIP->irDriveRegs.bFeaturesReg = 0;

pSCIP->irDriveRegs.bSectorCountReg = 1;

pSCIP->irDriveRegs.bSectorNumberReg = 1;

pSCIP->irDriveRegs.bCylLowReg = 0;

pSCIP->irDriveRegs.bCylHighReg = 0;

// calc the drive number.

pSCIP->irDriveRegs.bDriveHeadReg = 0xA0 | ( ( bDriveNum & 1 ) << 4 );

// The command can either be IDE identify or ATAPI identify.

pSCIP->irDriveRegs.bCommandReg = bIDCmd;

pSCIP->bDriveNumber = bDriveNum;

pSCIP->cBufferSize = IDENTIFY\_BUFFER\_SIZE;

return DeviceIoControl( hPhysicalDriveIOCTL, DFP\_RECEIVE\_DRIVE\_DATA,

( LPVOID ) pSCIP,

sizeof( SENDCMDINPARAMS ) - 1,

( LPVOID ) pSCOP,

sizeof( SENDCMDOUTPARAMS ) + IDENTIFY\_BUFFER\_SIZE - 1,

lpcbBytesReturned, NULL );

}

BOOL WinNTHDSerialNumAsPhysicalRead( BYTE\* dwSerial, UINT\* puSerialLen, UINT uMaxSerialLen )

{

#define DFP\_GET\_VERSION 0x00074080

BOOL bInfoLoaded = FALSE;

for( UINT uDrive = 0; uDrive < 4; ++ uDrive )

{

HANDLE hPhysicalDriveIOCTL = 0;

// Try to get a handle to PhysicalDrive IOCTL, report failure

// and exit if can't.

char szDriveName [256];

sprintf( szDriveName, "\\\\.\\PhysicalDrive%d", uDrive );

// Windows NT, Windows 2000, must have admin rights

hPhysicalDriveIOCTL = CreateFile( szDriveName,

GENERIC\_READ | GENERIC\_WRITE,

FILE\_SHARE\_READ | FILE\_SHARE\_WRITE, NULL,

OPEN\_EXISTING, 0, NULL);

if( hPhysicalDriveIOCTL != INVALID\_HANDLE\_VALUE )

{

GETVERSIONOUTPARAMS VersionParams = { 0 };

DWORD cbBytesReturned = 0;

// Get the version, etc of PhysicalDrive IOCTL

if( DeviceIoControl( hPhysicalDriveIOCTL, DFP\_GET\_VERSION,

NULL,

0,

&VersionParams,

sizeof( GETVERSIONOUTPARAMS ),

&cbBytesReturned, NULL ) )

{

// If there is a IDE device at number "i" issue commands

// to the device

if( VersionParams.bIDEDeviceMap != 0 )

{

BYTE bIDCmd = 0; // IDE or ATAPI IDENTIFY cmd

SENDCMDINPARAMS scip = { 0 };

// Now, get the ID sector for all IDE devices in the system.

// If the device is ATAPI use the IDE\_ATAPI\_IDENTIFY command,

// otherwise use the IDE\_ATA\_IDENTIFY command

bIDCmd = ( VersionParams.bIDEDeviceMap >> uDrive & 0x10 ) ? IDE\_ATAPI\_IDENTIFY : IDE\_ATA\_IDENTIFY;

BYTE IdOutCmd[sizeof( SENDCMDOUTPARAMS ) + IDENTIFY\_BUFFER\_SIZE - 1] = { 0 };

if( DoIdentify( hPhysicalDriveIOCTL,

&scip,

( PSENDCMDOUTPARAMS )&IdOutCmd,

( BYTE )bIDCmd,

( BYTE )uDrive,

&cbBytesReturned ) )

{

if( \* puSerialLen + 20U <= uMaxSerialLen )

{

CopyMemory( dwSerial + \* puSerialLen, ( ( USHORT\* )( ( ( PSENDCMDOUTPARAMS )IdOutCmd )->bBuffer ) ) + 10, 20 ); // 序列号

// Cut off the trailing blanks

for( UINT i = 20; i != 0U && ' ' == dwSerial[\* puSerialLen + i - 1]; -- i ) {}

\* puSerialLen += i;

CopyMemory( dwSerial + \* puSerialLen, ( ( USHORT\* )( ( ( PSENDCMDOUTPARAMS )IdOutCmd )->bBuffer ) ) + 27, 40 ); // 型号

// Cut off the trailing blanks

for( i = 40; i != 0U && ' ' == dwSerial[\* puSerialLen + i - 1]; -- i ) {}

\* puSerialLen += i;

bInfoLoaded = TRUE;

}

else

{

::CloseHandle( hPhysicalDriveIOCTL );

return bInfoLoaded;

}

}

}

}

CloseHandle( hPhysicalDriveIOCTL );

}

}

return bInfoLoaded;

}

UINT FindAwardBios( BYTE\*\* ppBiosAddr )

{

BYTE\* pBiosAddr = \* ppBiosAddr + 0xEC71;

BYTE szBiosData[128];

CopyMemory( szBiosData, pBiosAddr, 127 );

szBiosData[127] = 0;

int iLen = lstrlen( ( char\* )szBiosData );

if( iLen > 0 && iLen < 128 )

{

//AWard: 07/08/2002-i845G-ITE8712-JF69VD0CC-00

//Phoenix-Award: 03/12/2002-sis645-p4s333

if( szBiosData[2] == '/' && szBiosData[5] == '/' )

{

BYTE\* p = szBiosData;

while( \* p )

{

if( \* p < ' ' || \* p >= 127 )

{

break;

}

++ p;

}

if( \* p == 0 )

{

\* ppBiosAddr = pBiosAddr;

return ( UINT )iLen;

}

}

}

return 0;

}

UINT FindAmiBios( BYTE\*\* ppBiosAddr )

{

BYTE\* pBiosAddr = \* ppBiosAddr + 0xF478;

BYTE szBiosData[128];

CopyMemory( szBiosData, pBiosAddr, 127 );

szBiosData[127] = 0;

int iLen = lstrlen( ( char\* )szBiosData );

if( iLen > 0 && iLen < 128 )

{

// Example: "AMI: 51-2300-000000-00101111-030199-"

if( szBiosData[2] == '-' && szBiosData[7] == '-' )

{

BYTE\* p = szBiosData;

while( \* p )

{

if( \* p < ' ' || \* p >= 127 )

{

break;

}

++ p;

}

if( \* p == 0 )

{

\* ppBiosAddr = pBiosAddr;

return ( UINT )iLen;

}

}

}

return 0;

}

UINT FindPhoenixBios( BYTE\*\* ppBiosAddr )

{

UINT uOffset[3] = { 0x6577, 0x7196, 0x7550 };

for( UINT i = 0; i < 3; ++ i )

{

BYTE\* pBiosAddr = \* ppBiosAddr + uOffset[i];

BYTE szBiosData[128];

CopyMemory( szBiosData, pBiosAddr, 127 );

szBiosData[127] = 0;

int iLen = lstrlen( ( char\* )szBiosData );

if( iLen > 0 && iLen < 128 )

{

// Example: Phoenix "NITELT0.86B.0044.P11.9910111055"

if( szBiosData[7] == '.' && szBiosData[11] == '.' )

{

BYTE\* p = szBiosData;

while( \* p )

{

if( \* p < ' ' || \* p >= 127 )

{

break;

}

++ p;

}

if( \* p == 0 )

{

\* ppBiosAddr = pBiosAddr;

return ( UINT )iLen;

}

}

}

}

return 0;

}

本文来自CSDN博客，转载请标明出处：http://blog.csdn.net/skyremember/archive/2008/09/22/2959069.aspx