图解 VS2010 & TeeChart8 快速入门

TeeChart 是 Steema Software 开发的图表图形组件，它提供了上百种 2D 和 3D 图形风格、多达几十种数学和统计功能，在图表图形开发领域占有举足轻重的作用。由于其强大之处，许多软件基于 TeeChart 开发。作者在网上找了大量的资料，但是很不幸的是，关于介绍 VS2010 中如何使用 TeeChart 的资料少之又少，而且比较分散。本文的目的是带领 TeeChart 的初学者快速的在 VS2010 中使用 TeeChart 组件构建起自己的图形图表应用。本文将以 TeeChart Pro 提供的 FastLine 作为示例，带领大家进入 TeeChart 的世界。

本文的作者也是 TeeChart 的初学者，在整理和归纳这些资料的时候，做了大量的试验，不过有些示例代码的实现可能有更好的方式来实现。在此请各位看官在发现有更好的实现方式、或者有一些扩展的话，敬请联系本人 QQ：41607842。我予以更正，谢谢！

1. 准备

   要使用 TeeChart Pro 组件，需要下载 TeeChart8.ocx 文件。文件 TeeChart8.ocx 可放置于系统中的任何位置。为了方便使用和管理，我按照如下文件夹结构存放我的工程文件和.ocx 文件：

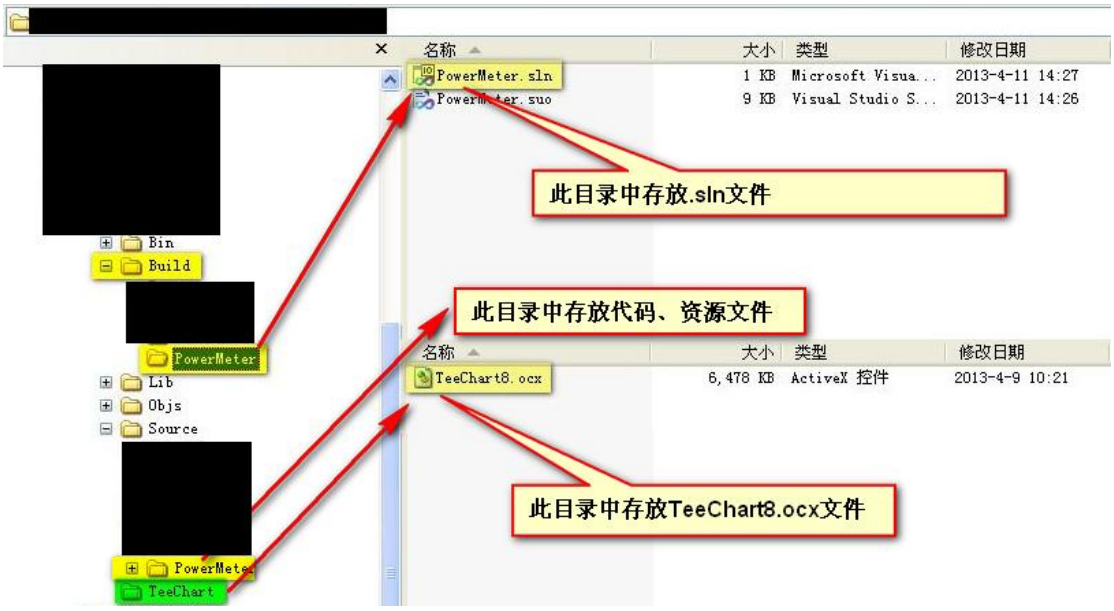

图 1

2. 在 VS2010 中新增一个 TeeChart8 Pro 的 FastLine 资源

   首先打开一个工程文件，然后在菜单栏，选择工具→选择工具箱，将 TeeChart8.ocx 添加进来，如下图：

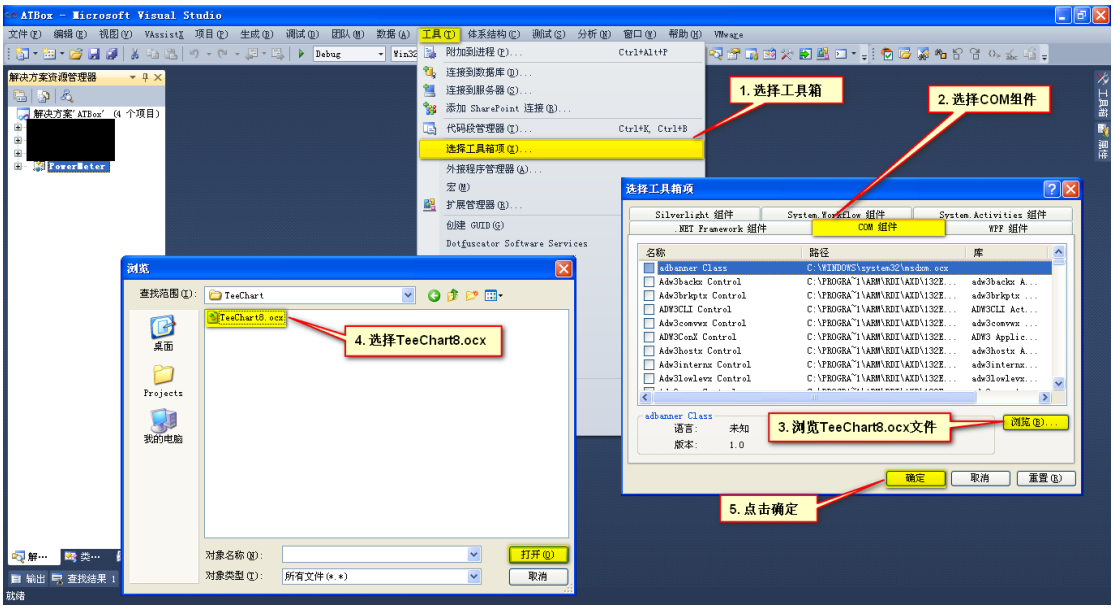

图 2

这样，我们就将 TeeChart 添加到我们的组件列表中来了。为了能够使用 FastLine，我们选择 TeeChart Pro Activex Control v8，如下图所示：

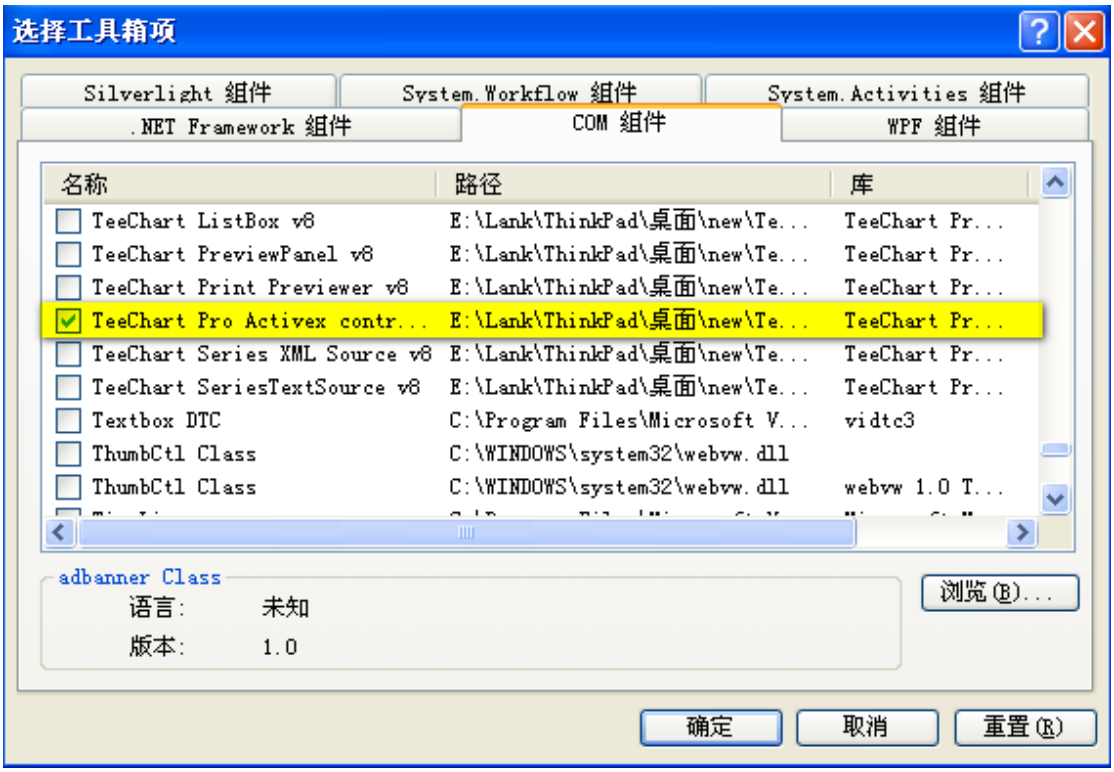

图 3

点击确定以后，打开工程的资源管理器，在右侧的工具箱中可以看到出现了一个新的工具 TeeChart pro Activex control v8，如下图所示：

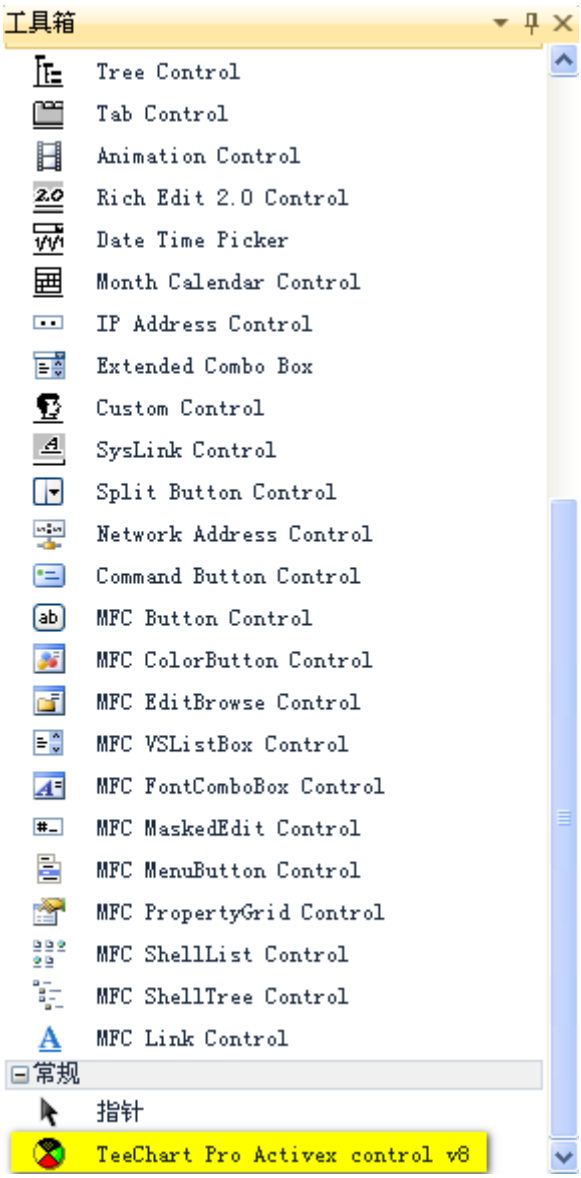

图 4

点击这个工具，在对话框中拖拉出一个控件出来：

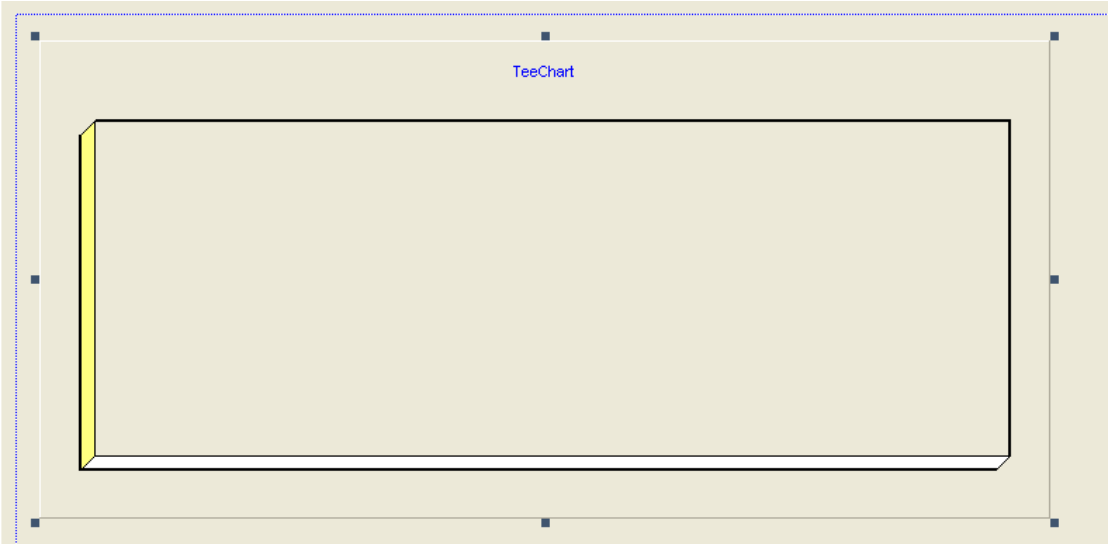

图 5

鼠标右击该控件，选择属性，然后可以调出该控件的属性，这里我们将该控件的 ID 改为：IDCTCHART：

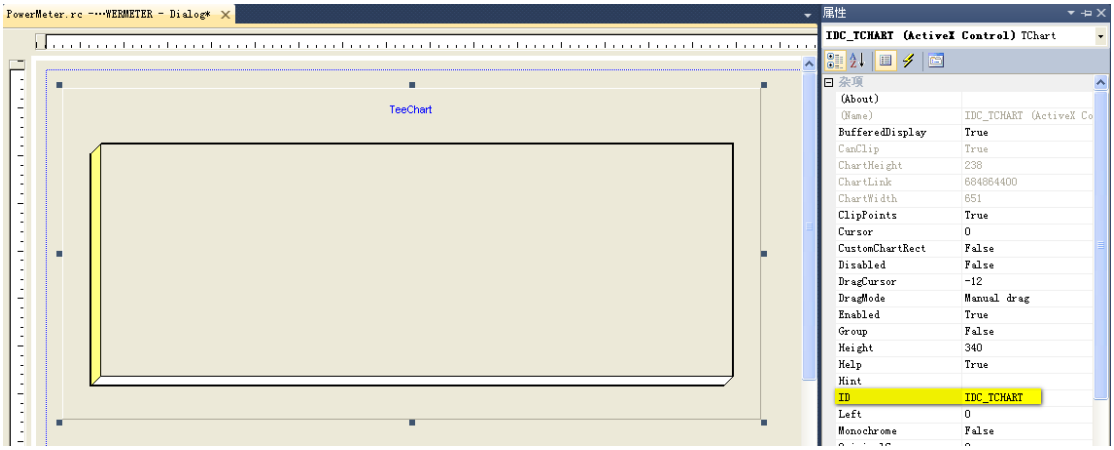

图 6

然后，点击**属性页**，可以对该控件进行进一步的设置。

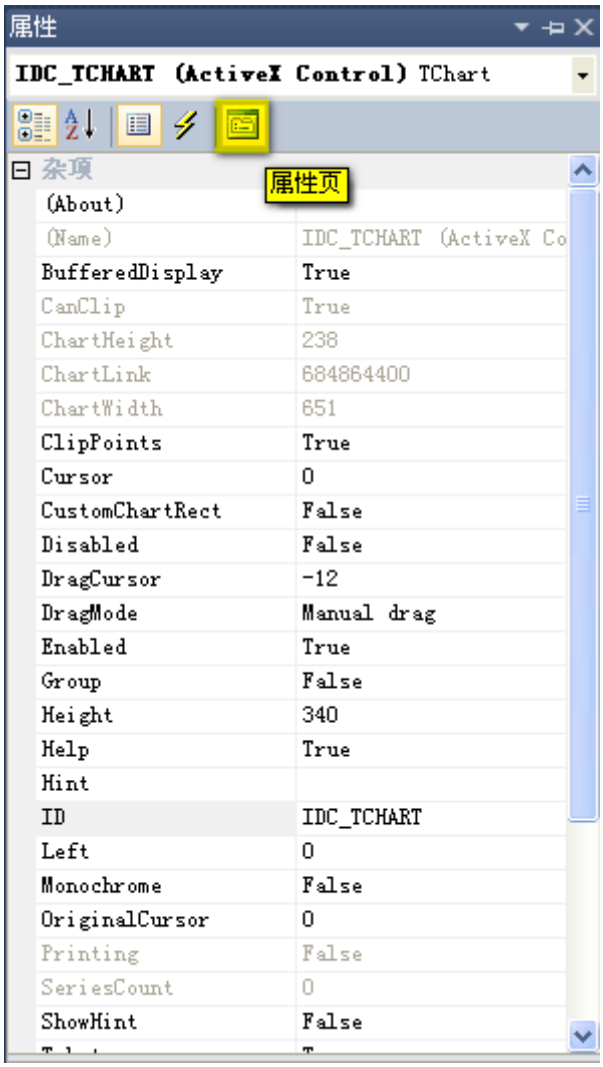
属性
IDC_TCHART (ActiveX Control) TChart
杂项
属性页
(About)
(Name) IDC_TCHART (ActiveX Co
BufferedDisplay True
CanClip True
ChartHeight 238
ChartLink 684864400
ChartWidth 651
ClipPoints True
Cursor 0
CustomChartRect False
Disabled False
DragCursor -12
DragMode Manual drag
Enabled True
Group False
Height 340
Help True
Hint
ID IDC_TCHART
Left 0
Monochrome False
OriginalCursor 0
Printing False
SeriesCount 0
ShowHint False

图 7

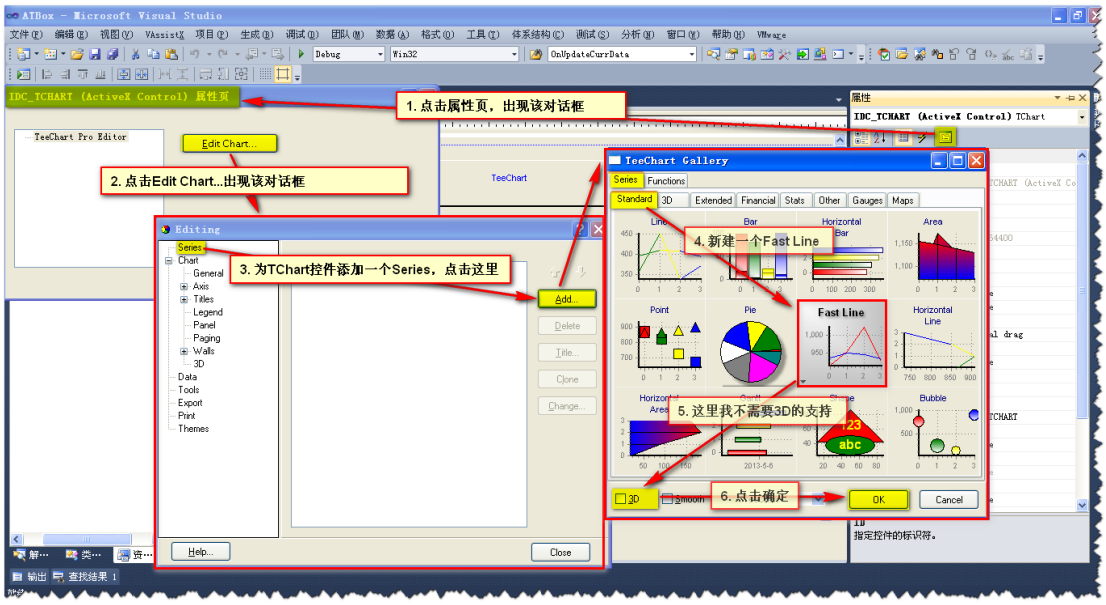
1. 点击属性页，出现该对话框
2. 点击Edit Chart...出现该对话框
3. 为TChart控件添加一个Series，点击这里
4. 新建一个Fast Line
5. 这里我不需要3D的支持
6. 点击确定
TeeChart Gallery
Editing
Edit Chart...
Add...
Fast Line
OK
Cancel
Close

图 8

经过上图的操作以后，我们就成功的添加了一个 2D 的 Fast Line。添加以后，属性页的效果如下图所示：

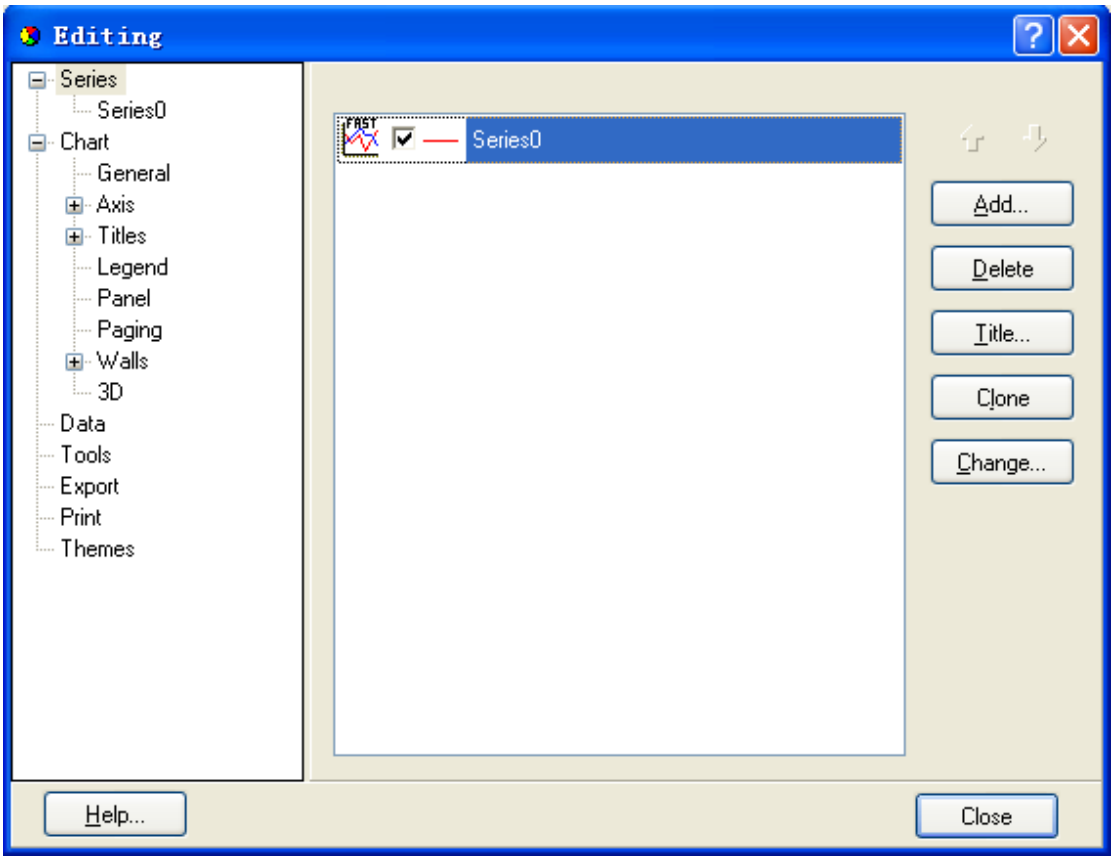

图 9

在图 9 中，可以进一步设置更详细的参数。

3. 将新增加的资源与变量建立起连接

为了能够正常的使用控件，需要新增加几个类。这些类继承自 TeeChart 提供的各种接口，方法如下：

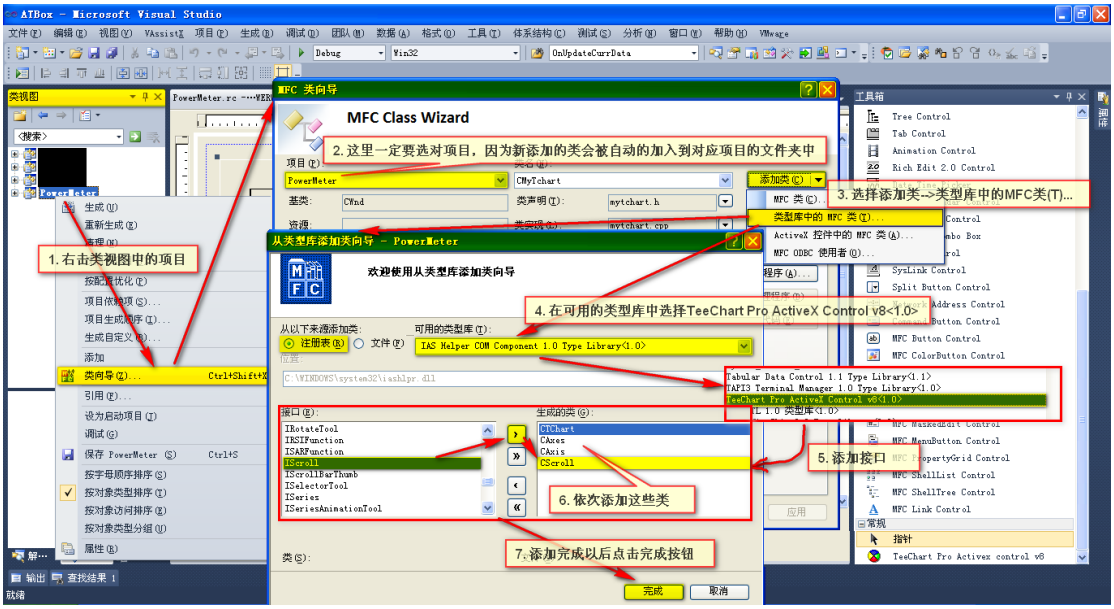

图 10

这些类的源文件存放在这里：

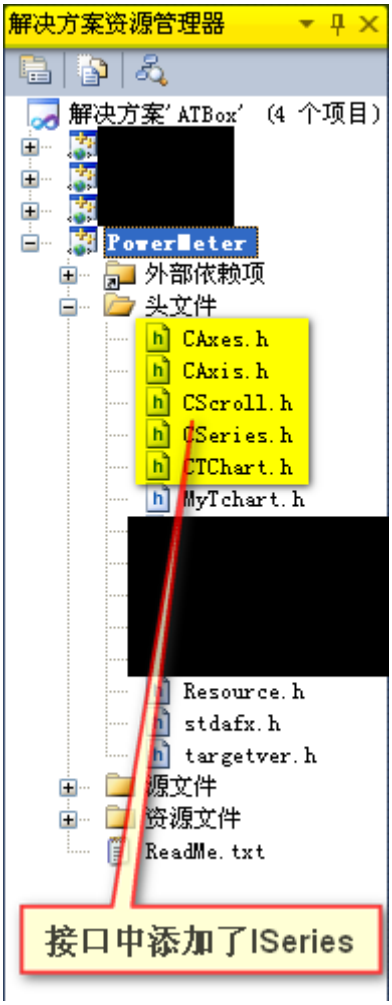

图 11

有了这些类以后（将来编程需要这些基本的类，如果需要更多的功能可以使用同样的方法增加新的接口），我们为控件创建变量：

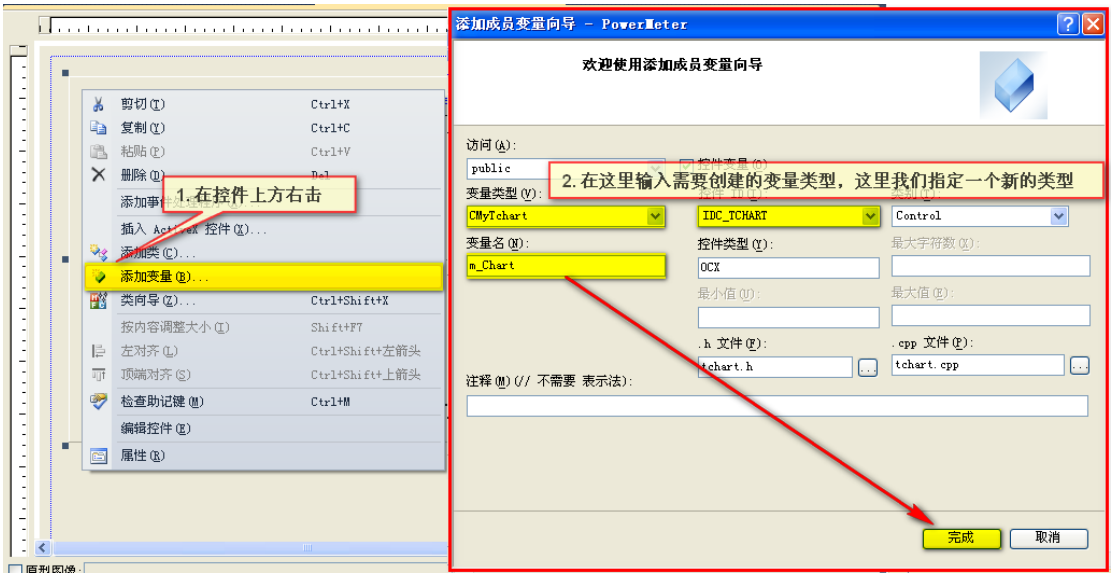

图 12

这样，工程中又多了两个文件：

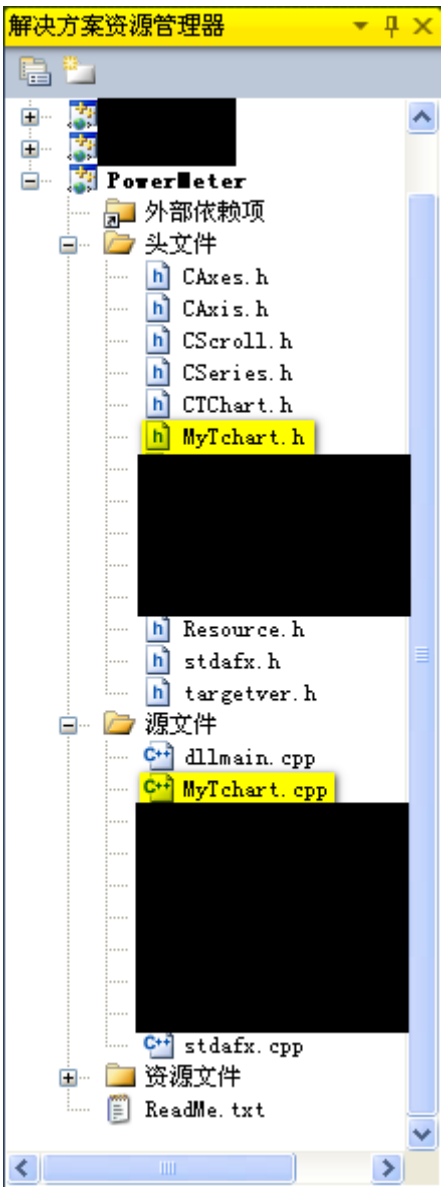

图 13

此时，我们可以在对话框的头文件中看到有如下定义：

```
private:
    // 在资源管理器中对该资源进行的任何编辑，需要重新拖拉其尺寸后，再保存、编译后才能生效
    CMyTchart    m_Chart;
```

图 14

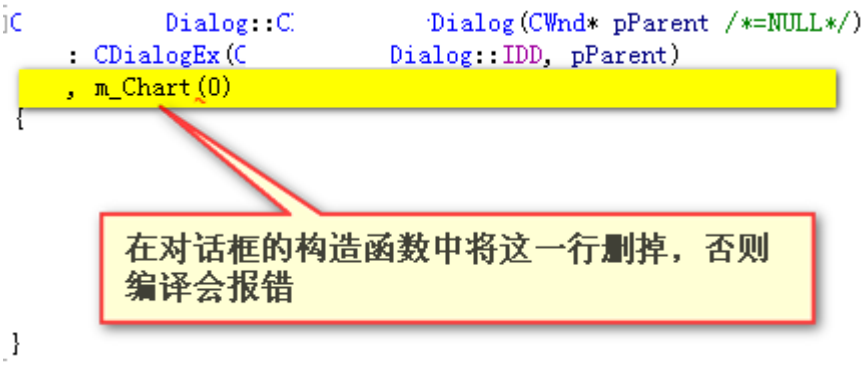

图 15

检查DoDataExchange函数是否已经添加了控件资源与变量的绑定：

```
void C Dialog::DoDataExchange(CDataExchange* pDX)
{
    CDialogEx::DoDataExchange(pDX);
    DDX_Control(pDX, IDC_TCHART, m_Chart);
}
```

图 16

至此，控件与变量的关联已经建立完毕。下面我们写一些测试代码，在 Fast Line 中显示数据。

4. 实时显示数据

为了能够实时显示数据，我们需要一个定时器，这在函数OnInitDialog中完成：

```
BOOL C Dialog::OnInitDialog()
{
    CDialogEx::OnInitDialog();

    // TODO:  在此添加额外的初始化
    SetTimer(CHARTTIMER, 100, NULL);

    return TRUE;  // return TRUE unless you set the focus to a control
    // 异常: OCX 属性页应返回 FALSE
}
```

图 17

然后，再定义几个成员变量：

```
double  m_Voltage[XAXISCOUNT];
double  m_Current[XAXISCOUNT];
double  m_Time[XAXISCOUNT];
```

图 18

在 OnTimer 函数中输入以下代码：

```
void C          Dialog::OnTimer(UINT_PTR nIDEvent)
{
    // TODO: 在此添加消息处理程序代码和/或调用默认值
    int nCount = m_Chart.get_SeriesCount();
    if (nCount != 0)
    {
        CSeries Chart = (CSeries)m_Chart.Series(0);



        int nSeriesCount = Chart.get_Count();
        if (0 == nSeriesCount)
        {
            DrawLine(m_Time, m_Voltage, XAXISCOUNT);
        }
        else
        {
            CString strLabel;
            strLabel.Format(_T("%d"), nSeriesCount - XAXISCOUNT);

            // Add a value to CSeries, the value of X-Axis increment 1 by default
            Chart.Add(rand() % 5, strLabel, 0);

            // X-Axis Scroll
            CAxes myAxes = (CAxes)m_Chart.get_Axis();
            CAxis myBottomAxis = (CAxis)myAxes.get_Bottom();
            myBottomAxis.Scroll(1, FALSE);
        }
    }

    CDialogEx::OnTimer(nIDEvent);
```

图 19

其中，函数 DrawLine 中的宏 XAXISCOUNT 是 X 轴的最大显示长度，函数 DrawLine 定义如下：

```
void C          Dialog::DrawLine(double* pX, double* pY, long nNum)
{
    COleSafeArray XValues;
    COleSafeArray YValues;

    long i;
    DWORD wLength = nNum;

    XValues.Create(VT_R8, 1, &wLength);
    YValues.Create(VT_R8, 1, &wLength);

    for (i = 0; i < nNum; ++i)
    {
        XValues.PutElement(&i, pX + i);
        YValues.PutElement(&i, pY + i);
    }

    CSeries Chart = (CSeries)m_Chart.Series(0);
    Chart.Clear();
    Chart.AddArray(nNum, YValues, XValues);
}
```

图 20

至此，我们完成了实时曲线的绘制，可以编译并运行程序查看效果了。

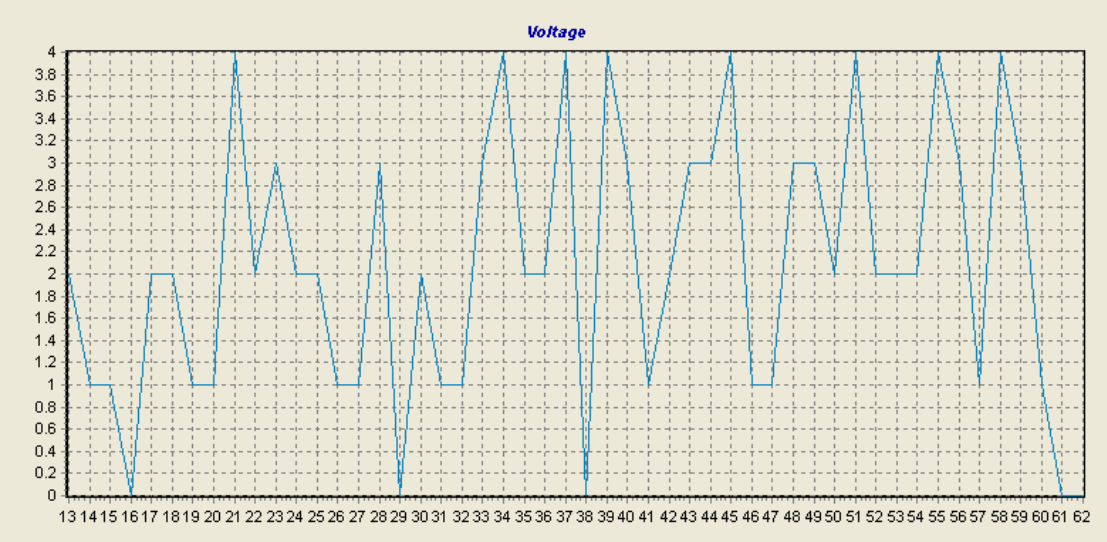
Voltage
4
3.8
3.6
3.4
3.2
3
2.8
2.6
2.4
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62

图 21