# Project Name: T3000\_CrossPlatform

This document describes how to use a COM component written in C# , in the main C++/MFC application. Specifically, the control basic editor, it is done in C# and can be called from within the T3000

GITHUB: <https://github.com/temcocontrols/T3000_CrossPlatform>

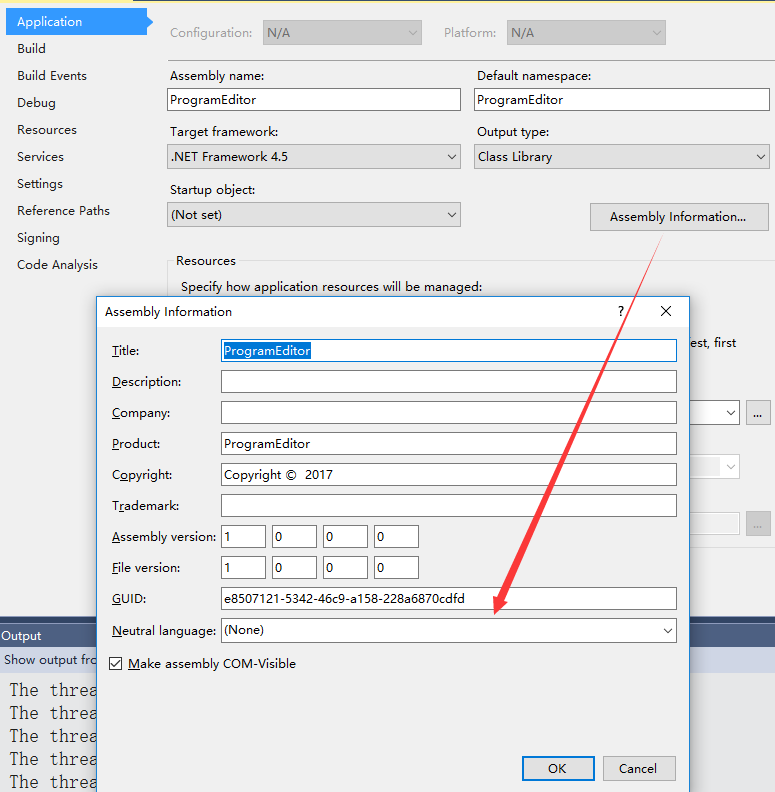
Here is a document about program editor

<https://github.com/temcocontrols/T3000_CrossPlatform/tree/master/T3000Documenter/Help>

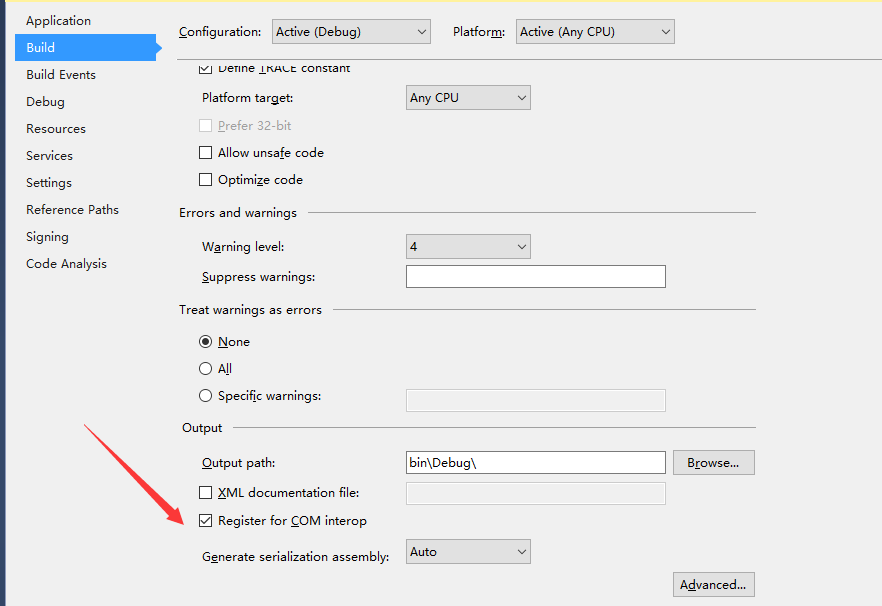
TBD: Move this to the \Documentation folder of the cross platform project.

How to Make C# Component use in MFC

1. Make the com visible

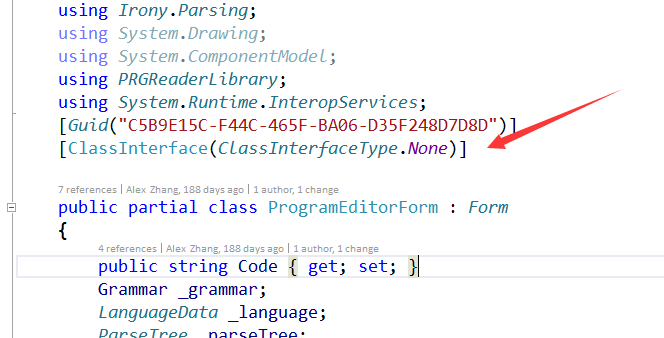


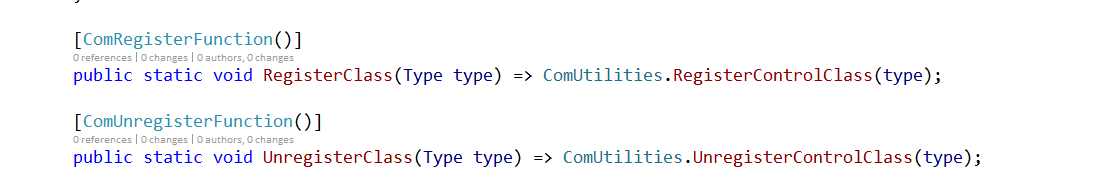
2.Register for COM interop



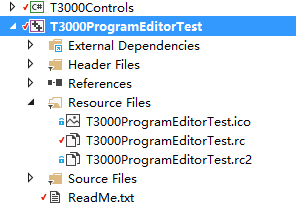
1. Put the [ComUtilities](#ComUtilities) class into the project .the code is in the background in the document .

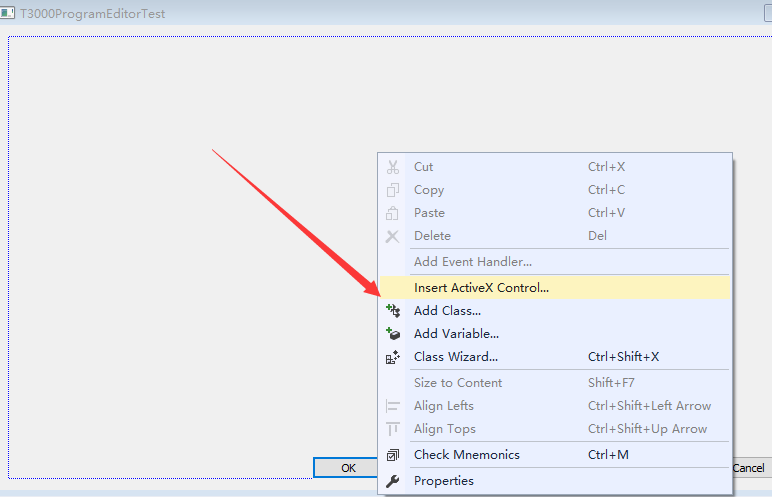
1. Select the component what you want to use in other language .



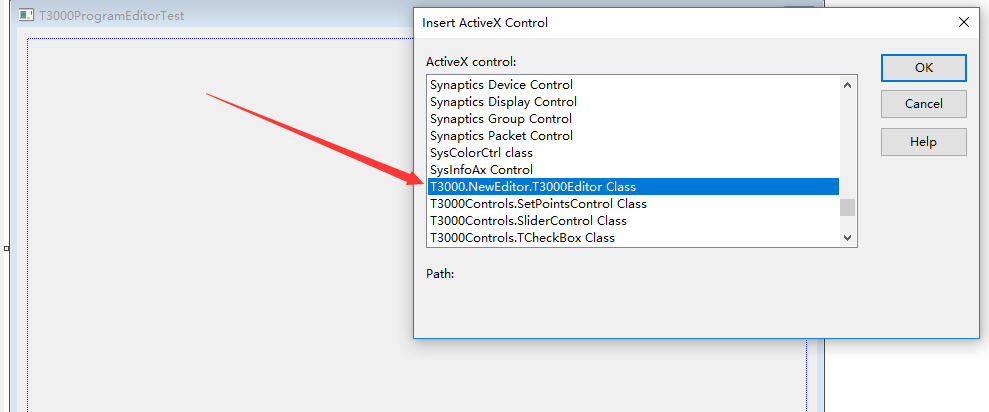


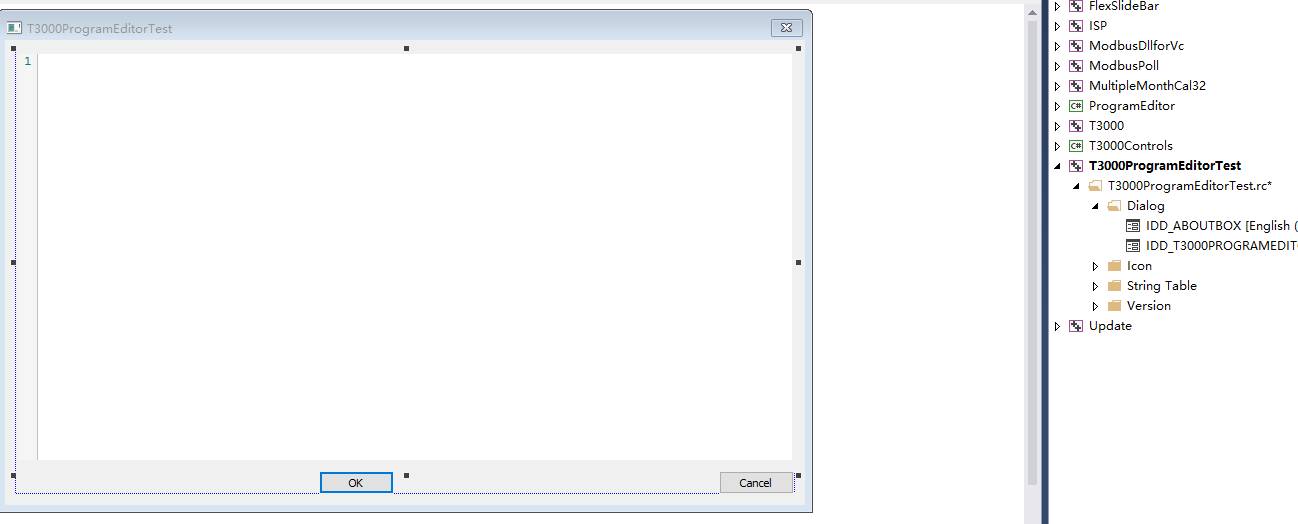
1. Make a test project



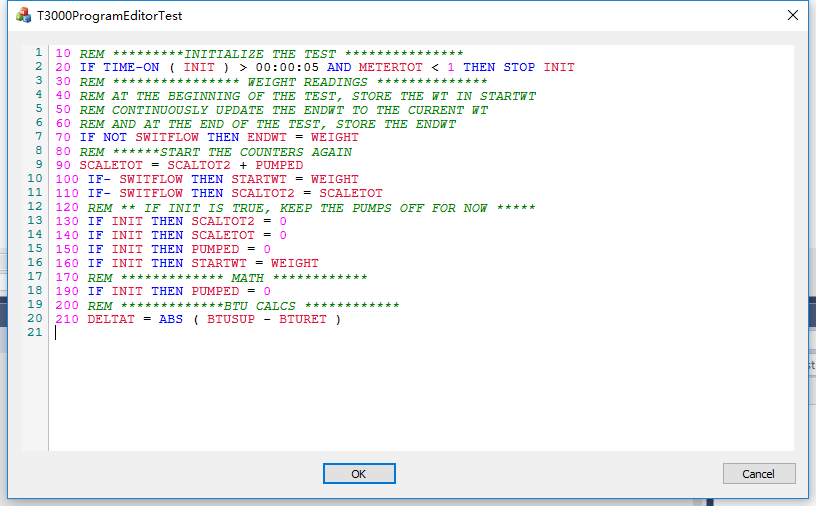


Select the component that we want to use .





F5 ,run the test project .



CODE – ComUtilities

namespace T3000.Forms

{

using *System*;

using *System*.*Reflection*;

using *Microsoft*.*Win32*;

using *System*.*Runtime*.*InteropServices*;

internal class NativeMethods

{

internal static class OLEMISC

{

internal const int OLEMISC\_RECOMPOSEONRESIZE = 0x1;

internal const int OLEMISC\_ONLYICONIC = 0x2;

internal const int OLEMISC\_INSERTNOTREPLACE = 0x4;

internal const int OLEMISC\_STATIC = 0x8;

internal const int OLEMISC\_CANTLINKINSIDE = 0x10;

internal const int OLEMISC\_CANLINKBYOLE1 = 0x20;

internal const int OLEMISC\_ISLINKOBJECT = 0x40;

internal const int OLEMISC\_INSIDEOUT = 0x80;

internal const int OLEMISC\_ACTIVATEWHENVISIBLE = 0x100;

internal const int OLEMISC\_RENDERINGISDEVICEINDEPENDENT = 0x200;

internal const int OLEMISC\_INVISIBLEATRUNTIME = 0x400;

internal const int OLEMISC\_ALWAYSRUN = 0x800;

internal const int OLEMISC\_ACTSLIKEBUTTON = 0x1000;

internal const int OLEMISC\_ACTSLIKELABEL = 0x2000;

internal const int OLEMISC\_NOUIACTIVATE = 0x4000;

internal const int OLEMISC\_ALIGNABLE = 0x8000;

internal const int OLEMISC\_SIMPLEFRAME = 0x10000;

internal const int OLEMISC\_SETCLIENTSITEFIRST = 0x20000;

internal const int OLEMISC\_IMEMODE = 0x40000;

internal const int OLEMISC\_IGNOREACTIVATEWHENVISIBLE = 0x80000;

internal const int OLEMISC\_WANTSTOMENUMERGE = 0x100000;

internal const int OLEMISC\_SUPPORTSMULTILEVELUNDO = 0x200000;

}

}

internal static class ComUtilities

{

/// <summary>

/// Register the class as a control and set it's CodeBase entry.

/// Key format: "HKEY\_CLASSES\_ROOT\CLSID\{guid}"

/// </summary>

/// <param name="type">The type of the control</param>

public static void RegisterControlClass(Type type, int bitmapId = 101)

{

var keyName = @"CLSID\" + type.*GUID*.*ToString*("B");

using (var key = *Registry*.*ClassesRoot*.*OpenSubKey*(keyName, true))

{

key.*SetValue*(null, type.*ToString*() + " Class");

key.*CreateSubKey*("Programmable").*Close*();

key.*CreateSubKey*("Control").*Close*();

key.*CreateSubKey*("Insertable").*Close*();

using (var subkey = key.*CreateSubKey*("MiscStatus"))

{

var miscStatus =

NativeMethods.OLEMISC.OLEMISC\_RECOMPOSEONRESIZE +

NativeMethods.OLEMISC.OLEMISC\_INSIDEOUT +

NativeMethods.OLEMISC.OLEMISC\_ACTIVATEWHENVISIBLE +

NativeMethods.OLEMISC.OLEMISC\_SETCLIENTSITEFIRST;

subkey.SetValue(null, miscStatus.*ToString*());

}

using (var subkey = key.*CreateSubKey*("TypeLib"))

{

var libid = *Marshal*.*GetTypeLibGuidForAssembly*(type.*Assembly*);

subkey.SetValue(null, libid.*ToString*("B"));

}

//key.DeleteSubKeyTree("InprocServer32");

using (var subkey = key.*CreateSubKey*("InprocServer32"))

{

//subkey.SetValue(null, Environment.SystemDirectory + @"\mscoree.dll");

subkey.SetValue("CodeBase", *Assembly*.*GetExecutingAssembly*().*CodeBase*);

//subkey.SetValue(null, Assembly.GetExecutingAssembly().Location);

//subkey.SetValue("ThreadingModel", "Apartment");

}

//using (var subkey = key.CreateSubKey("Server"))

//{

// subkey.SetValue(null, "mscorld.dll");

//}

//

//using (var subkey = key.CreateSubKey("ToolboxBitmap32"))

//{

//subkey.SetValue(null,

// Assembly.GetExecutingAssembly().Location + ", " +

// bitmapId.ToString());

//}

using (var subkey = key.*CreateSubKey*("Implemented Categories"))

{

subkey.*CreateSubKey*("{7DD95801-9882-11CF-9FA9-00AA006C42C4}");

subkey.*CreateSubKey*("{7DD95802-9882-11CF-9FA9-00AA006C42C4}");

subkey.*CreateSubKey*("{40FC6ED3-2438-11CF-A3DB-080036F12502}");

subkey.*CreateSubKey*("{40FC6ED4-2438-11CF-A3DB-080036F12502}");

subkey.*CreateSubKey*("{40FC6ED5-2438-11CF-A3DB-080036F12502}");

subkey.*CreateSubKey*("{4FED769C-D8DB-44ea-99EA-65135757C156}");

subkey.*CreateSubKey*("{0DE86A52-2BAA-11CF-A229-00AA003D7352}");

subkey.*CreateSubKey*("{0DE86A53-2BAA-11CF-A229-00AA003D7352}");

subkey.*CreateSubKey*("{0DE86A57-2BAA-11CF-A229-00AA003D7352}");

}

using (var subkey = key.*CreateSubKey*("Version"))

{

int majorVersion;

int minorVersion;

*Marshal*.*GetTypeLibVersionForAssembly*(

type.*Assembly*, out majorVersion, out minorVersion);

subkey.SetValue(null, $"{majorVersion}.{minorVersion}");

}

key.*CreateSubKey*("InstalledVersion").*Close*();

}

}

/// <summary>

/// Called to unregister the control.

/// Key format: "HKEY\_CLASSES\_ROOT\CLSID\{guid}"

/// </summary>

/// <param name="type">The type of the control</param>

public static void UnregisterControlClass(Type type)

{

*Registry*.*ClassesRoot*.*DeleteSubKeyTree*(@"CLSID\" + type.*GUID*.*ToString*("B"), false);

}

}

}