



# 基于C++语言的GTS控制卡

---

## 初级入门手册 (点位运动示例)

# 目录

一、如何识别控制卡端子板型号 .....	1
二、安装控制卡和驱动 .....	4
三、驱动安装完以后，开始编程。 .....	7
1、首先打开 VS2008.....	7
2、然后点击新建项目 .....	8
3、将产品配套光盘 .....	9
4、使用 MCT2008 进行点动 .....	10
5、生成 MCT2008 配置文件 .....	11
6、导入 MCT2008 文件 .....	12
7、调用 lib 文件 .....	13
8、添加头文件 .....	14
9、添加第一个控件 .....	14
10、更改控件属性 .....	15
11、为控件添加变量 .....	16
12、设置全局变量 .....	17
13、添加消息处理函数 .....	17
14、编辑第一个按钮代码 .....	18
15、编辑其它按钮代码 .....	19
16、调试运行 .....	20
17、启动运动 .....	20
18、停止运动及关闭程序 .....	20

## 一、如何识别控制卡端子板型号

方法1、包装盒,如图1



图 1

方法2、控制卡和端子板标签，如图2，如图3



图 2



图 3

注：GT2-800-ACC2-VB-G-A 具体型号说明请参考用户手册

## 二、安装控制卡和驱动

1、打开主机后盖，将卡插到PCI插槽上，然后右击我的电脑，点击属性，通过查看系统类型查看电脑的操作系统是32为还是64位；打开设备管理器，会看到PCI设备上有个感叹号，然后右击，弹出更新驱动程序软件，选择对应的驱动程序如图4，如图5，如图6，如图7，如图8，如图9所示



图4

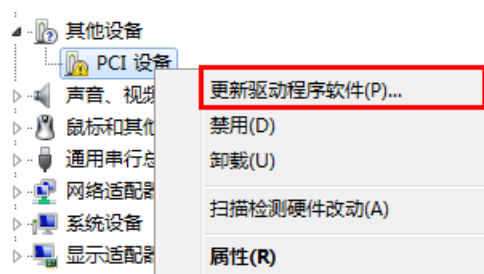


图5



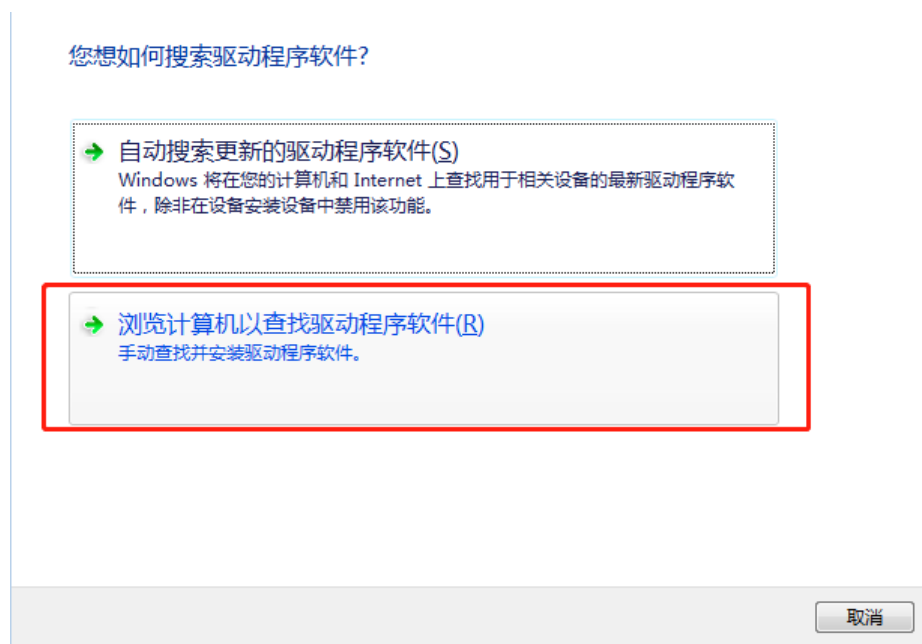


图 6

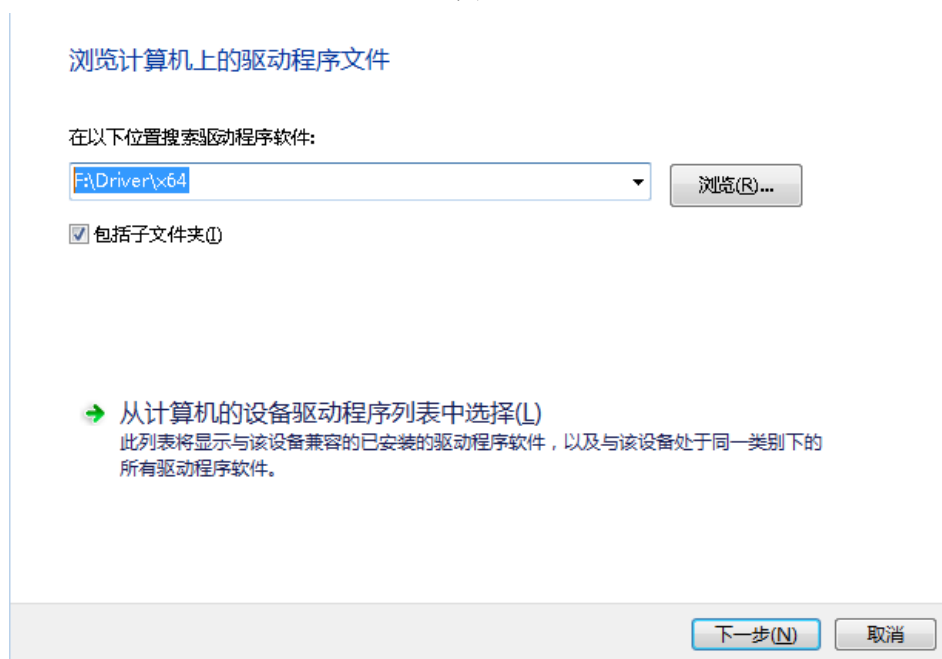


图 7



图 8

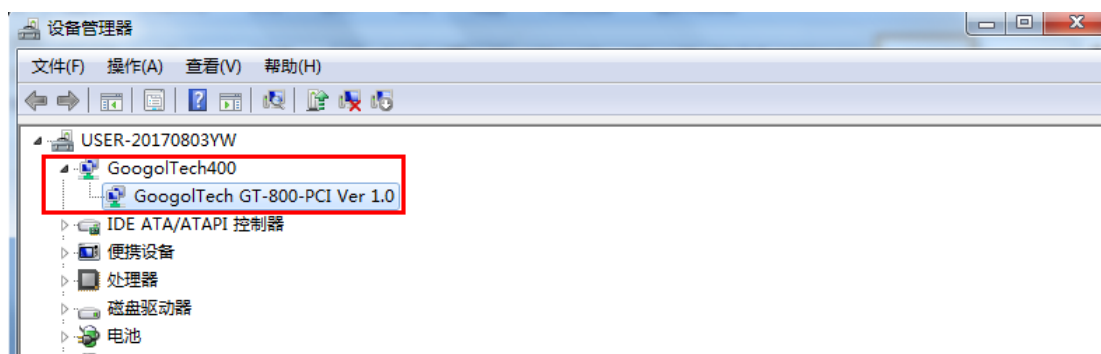


图9

注：*GTS400* 和 *GTS800* 控制卡安装完驱动以后，都会显示 *GoogolTech GT-800-PCI Ver 1.0*



三、驱动安装完以后，开始编程。

1、首先打开 VS2008，如图 10 所示。

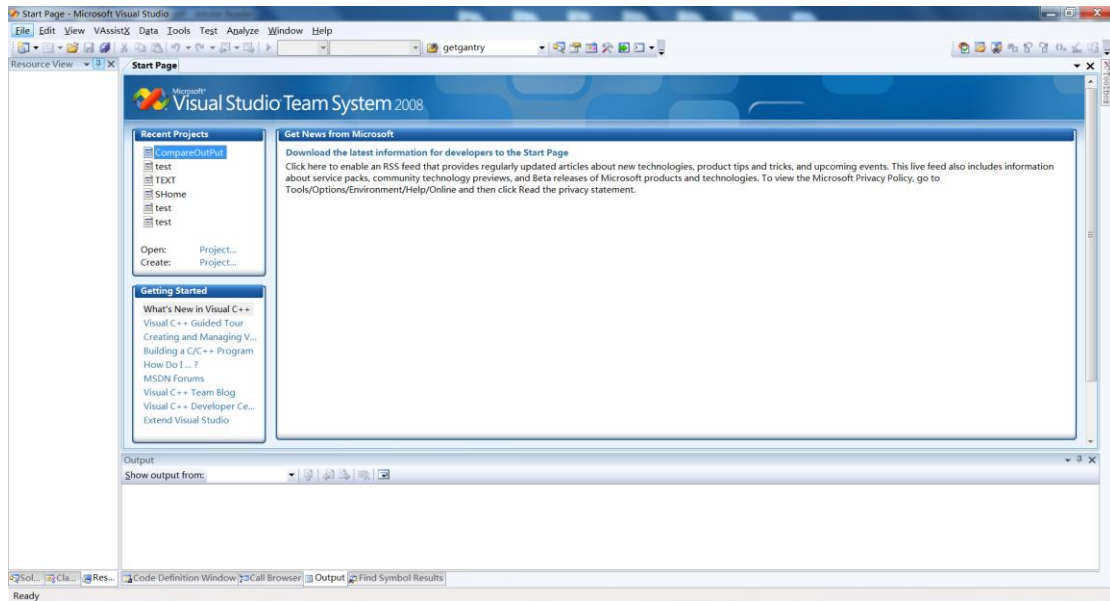


图 10

2、然后点击新建项目，选择 Visual C++，然后选择创建 MFC Application(MFC 应用程序)，将名称改为点位运动，点击 Next，然后选择 Dialog base(基于对话框)，最后点击 Finish 如图 11，如图 12。

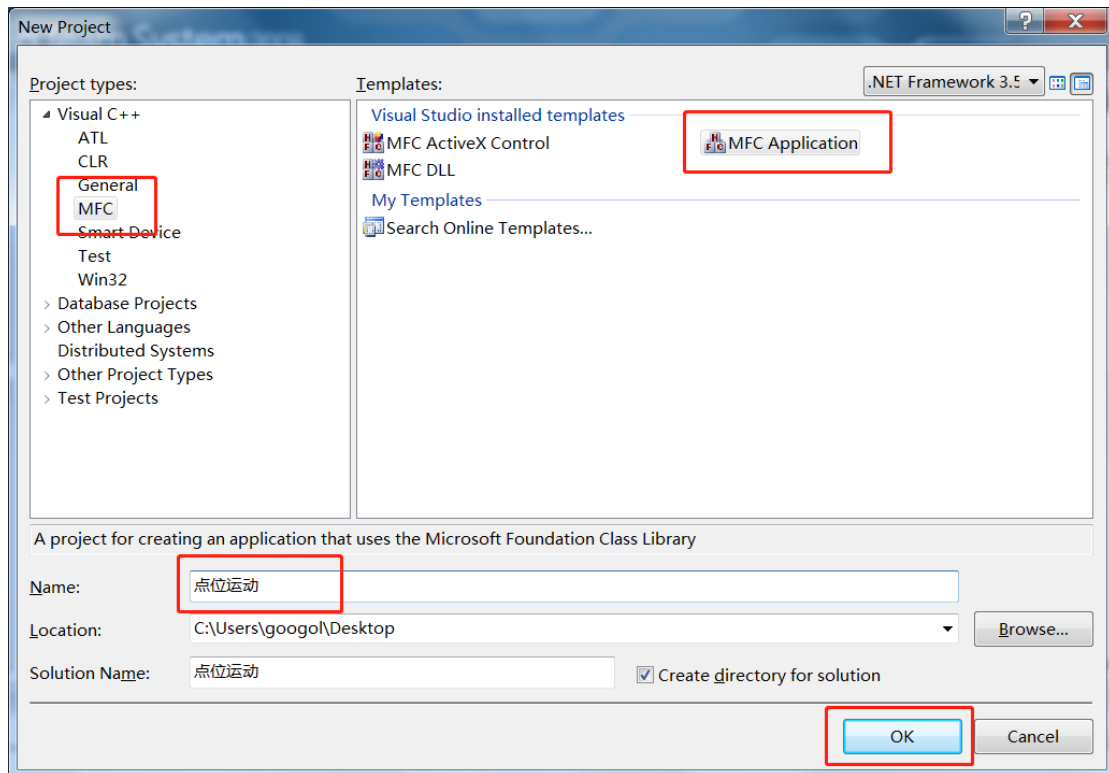


图 11

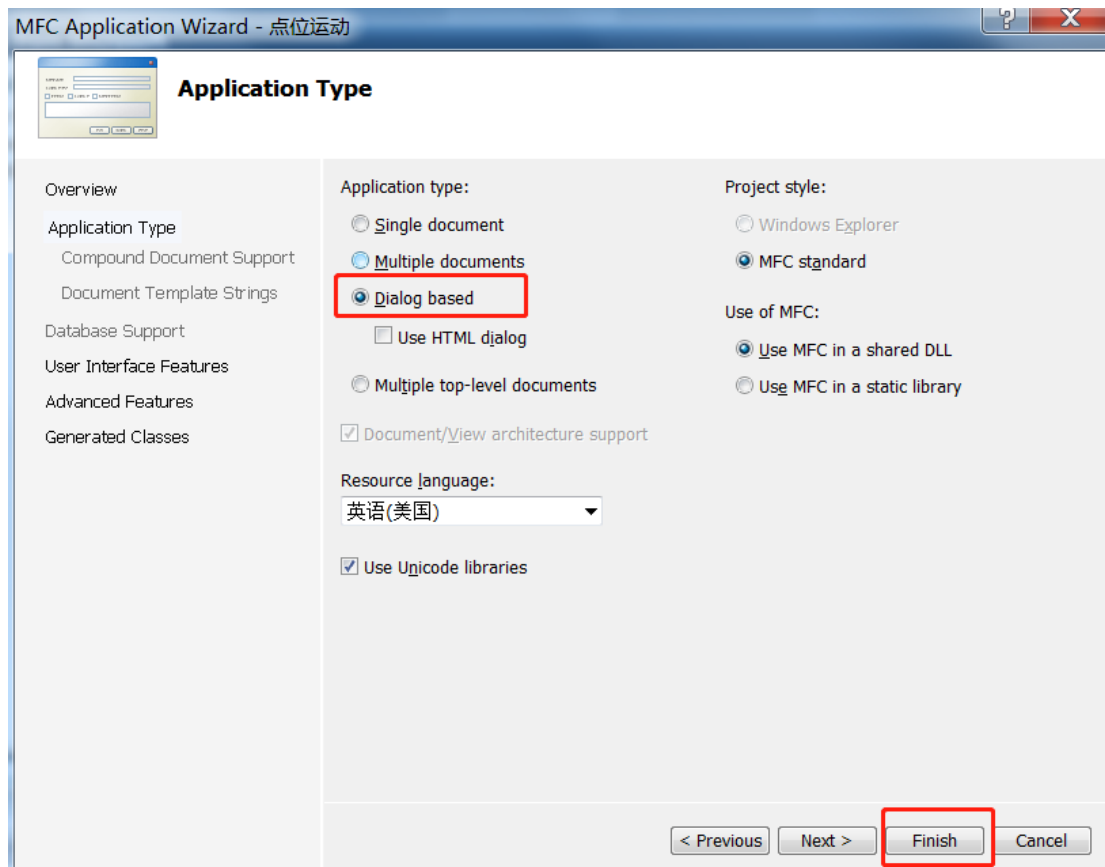


图 12

3、将产品配套光盘Windows\dll文件夹中的动态链接库、头文件和lib文件复制到工程文件夹中，如图13

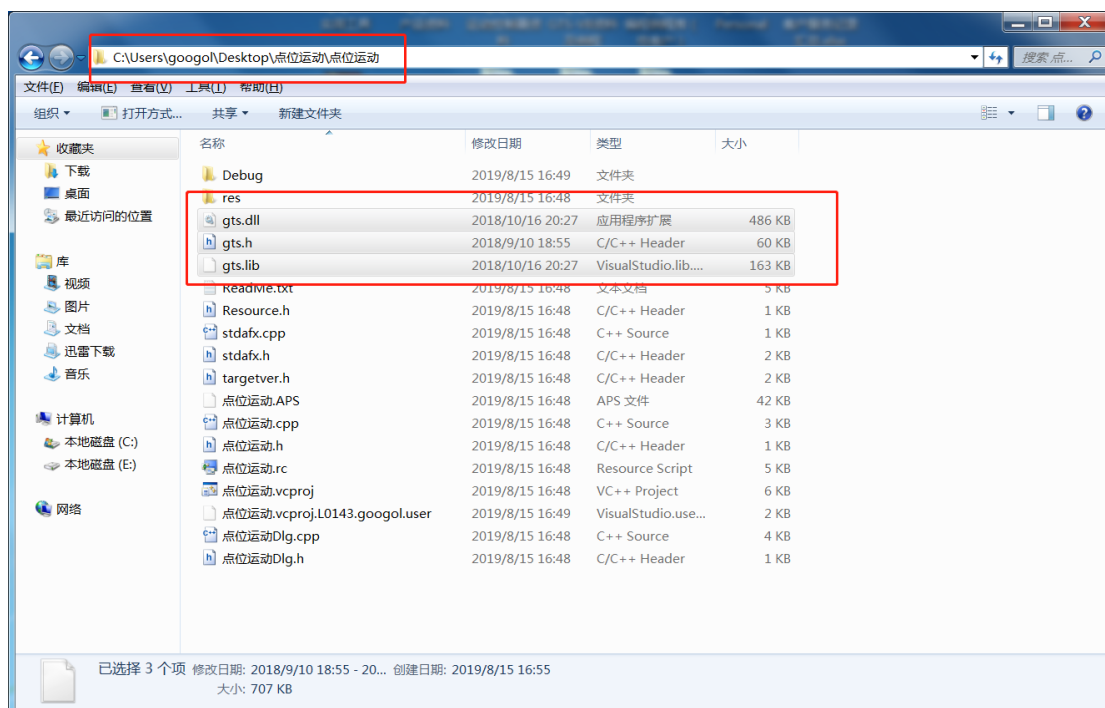


图 13

4、打开mct2008，选择工具→控制器配置，正、负限位选择none，然后点击控制→写入控制器，点击视图→轴状态，确认驱动、限位都无报警后，点击视图→Jog进行点动如图14，15，16



图 14



图 15



图 16

5、点击文件→写入到文件，将文件名改成 GTS800，保存到桌面，如图 17

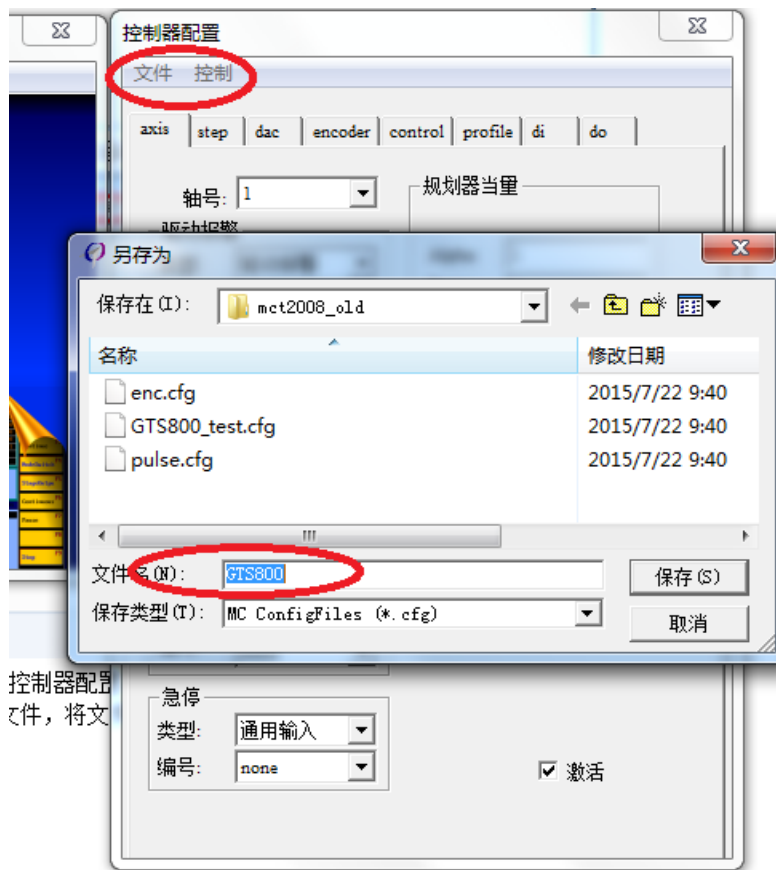


图 17

6、把桌面生成的 GTS800 文件复制到

(C:\Users\googol\Desktop\点位运动\点位运动) 文件下，

如图 18

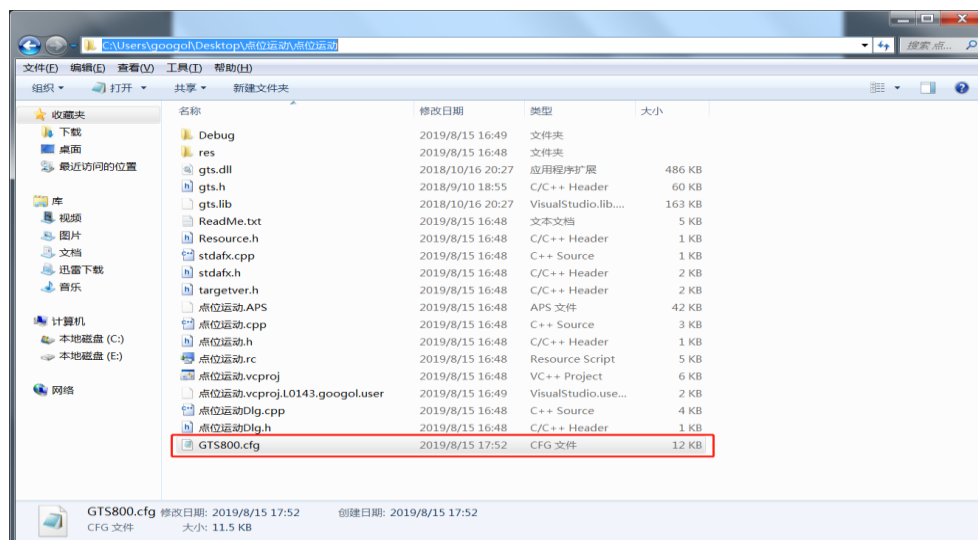


图 18

7、打开刚才创建的 vs2008 程序，在左侧的解决方案管理器中右击点位运动→属性→配置属性→链接器→输入→附加依赖项栏中输入 lib 文件名，例如 gts.lib，如图 19，如图 20

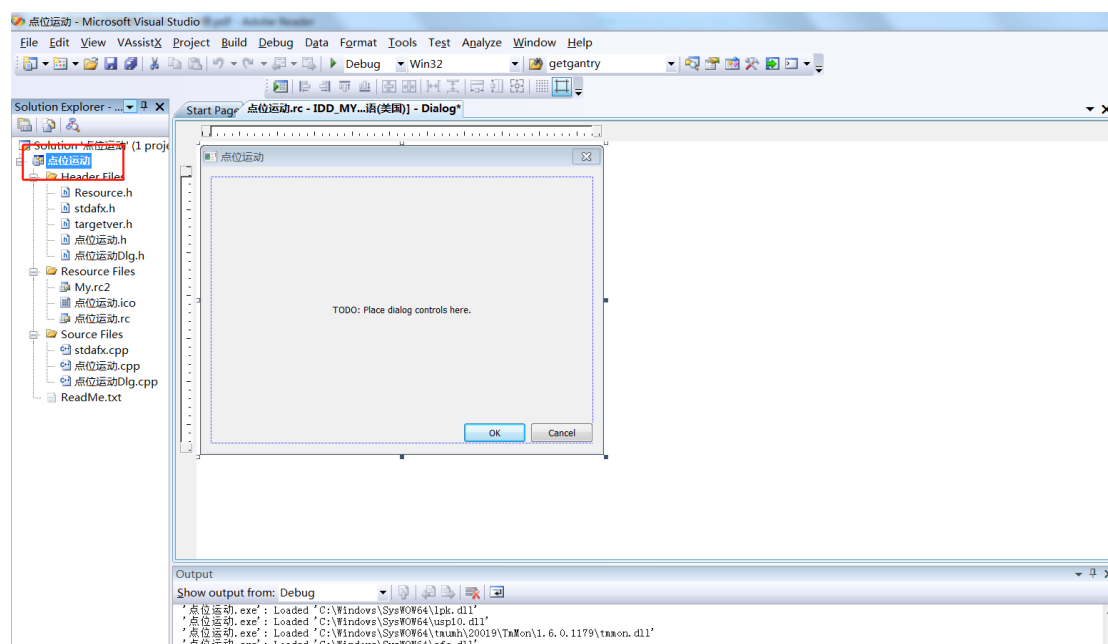


图 19

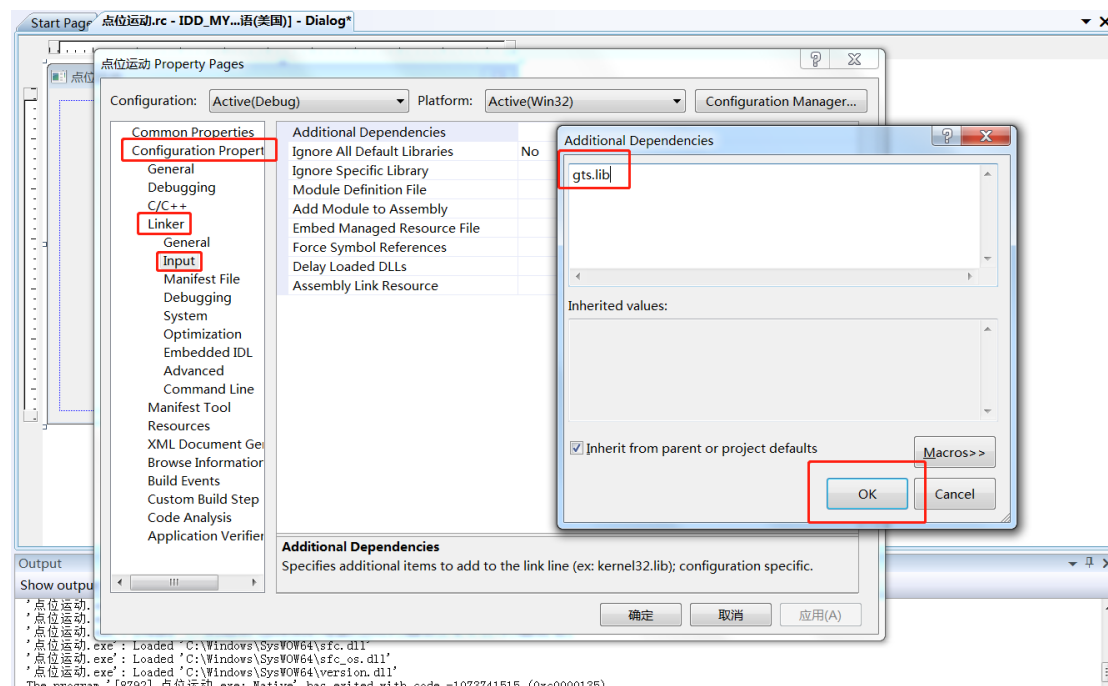


图 20



8、在应用程序文件中加入函数库头文件的声明，例如：

`#include "gts.h"`；至此，用户就可以在Visual C++中调用函数库中的任何函数，开始编写应用程序。对于步骤7，还有一种比较简便的方法，那就是在应用程序文件里面添加包含链接文件的声明，例如：`#pragma comment`  
(lib,"gts.lib")，如图21

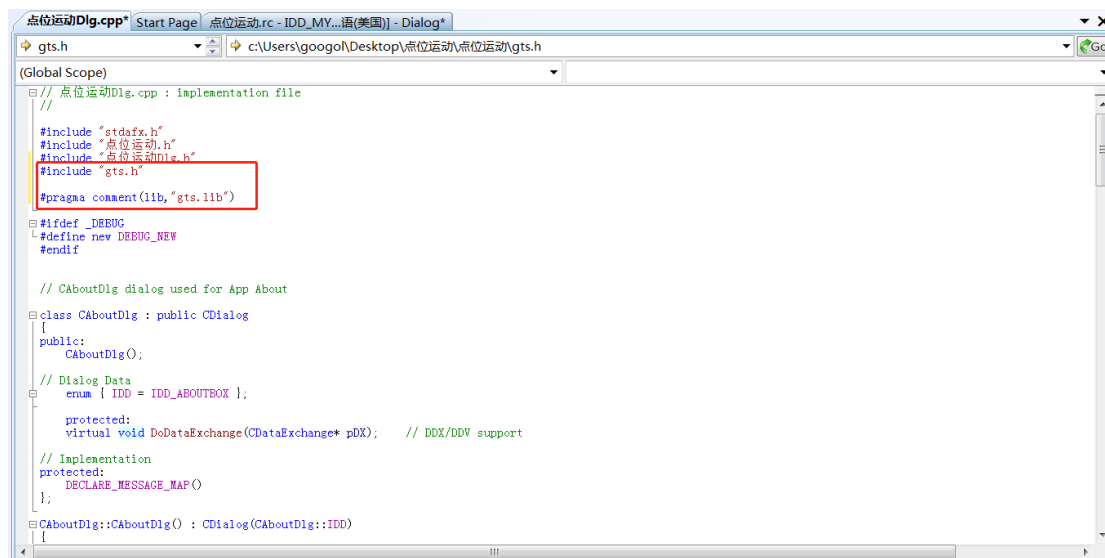


图 21

9、下一步开始编程，先将对话框上多余的控件删除，在右侧工具箱中选中 **button** 控件，然后添加到点位运动窗体中，如图 22

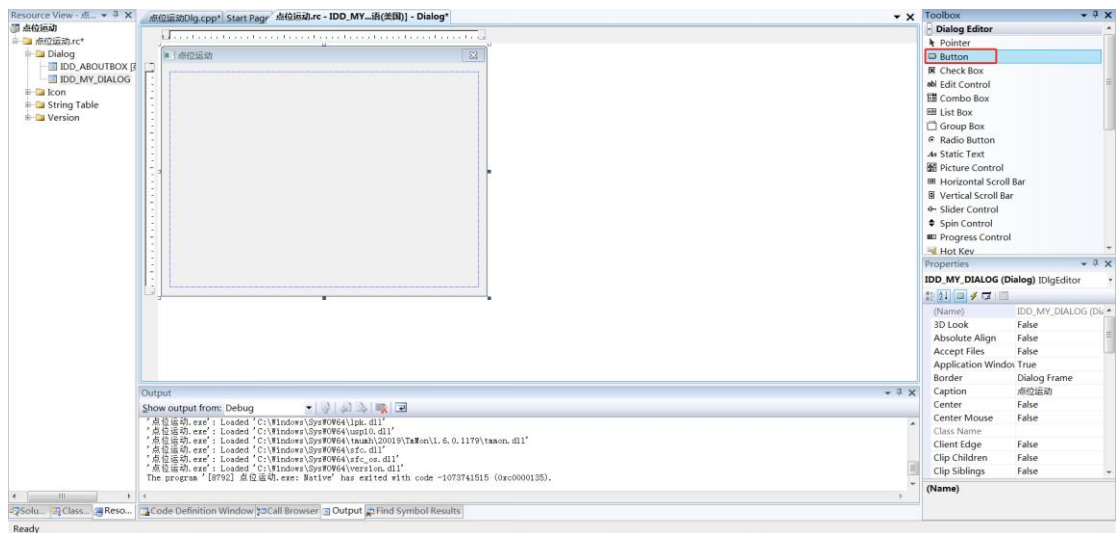


图 22

10、然后选中 **button1**，在右下角的属性窗口中，选择 **text** 属性，将其改为“初始化”，如图 23

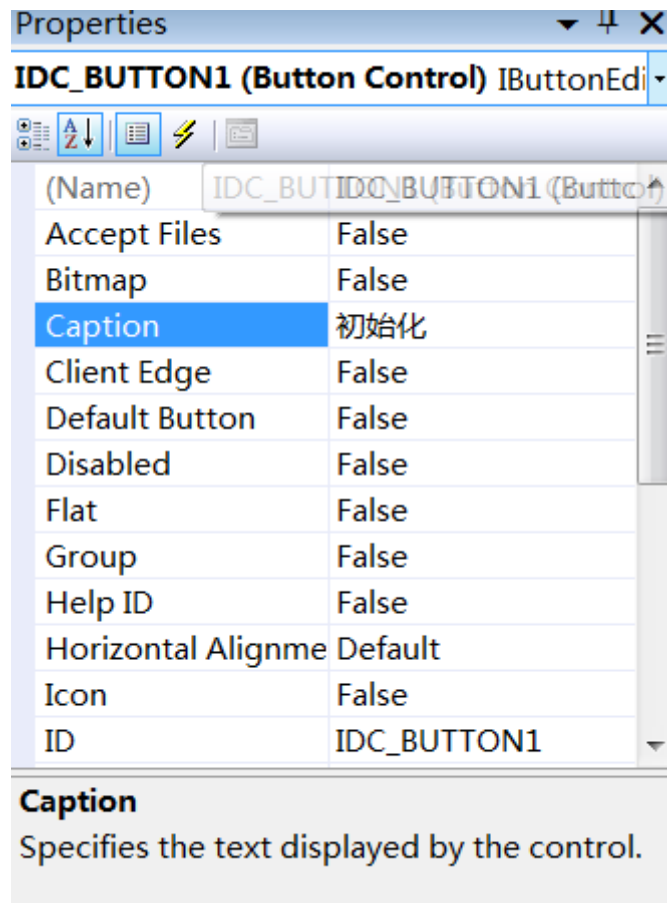


图 23

11、然后再在工具箱中选择Static Text, Edit Control控件放到窗体中，并将其控件Caption属性分别改成图中所示名称，如图24



图24

接着，为点位距离和点位速度的编辑框IDC\_EDIT1和IDC\_EDIT2添加变量。

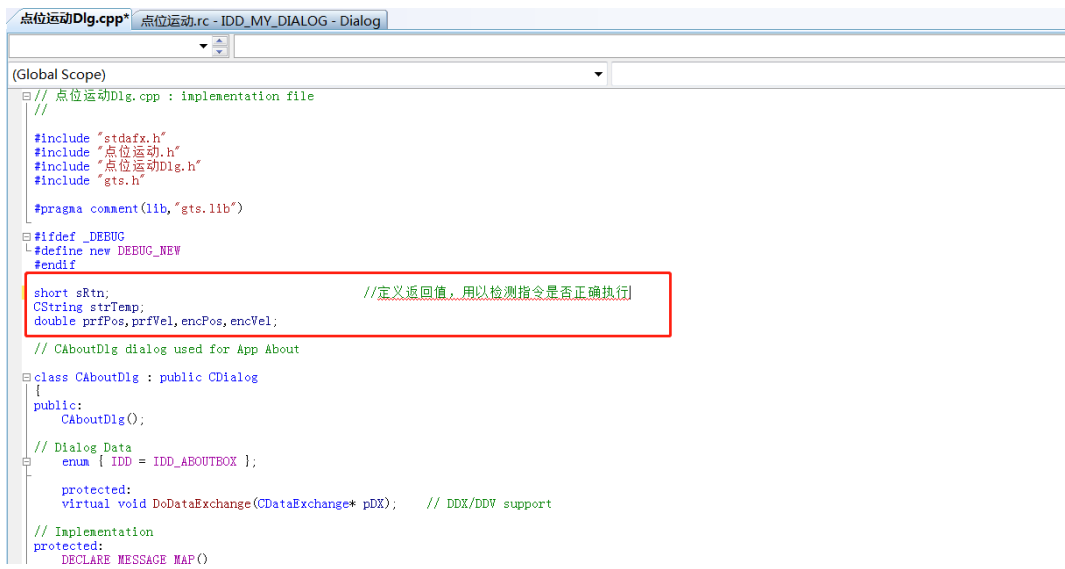
(1) 在编辑框上点右键，在右键菜单中选择“Add Variable”。弹出添加成员变量的向导对话框。

(2) 我们想为其添加值变量而不是控件变量，所以对话框中“Category”下的组合框中选择 Value。

(3) “Variable type”下的组合框此时默认选中的是“CString”，CString 是字符串类，显然不能进行数据交换。我们可以选择 double、float、int 等。这里我们选择 double，即编辑框关联一个 double 类型的变量。

(4) 在“Variable name”中写入自定义的变量名。我为其取名 m\_editPos 和 m\_editVel。

12、下一步开始编程，首先，声明全局变量，如图25



```
// 点位运动Dlg.cpp : implementation file
//
#include "stdafx.h"
#include "点位运动.h"
#include "点位运动Dlg.h"
#include "gts.h"

#pragma comment(lib, "gts.lib")

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

short sRtn; //定义返回值, 用以检测指令是否正确执行
CString strTemp;
double prfPos, prfVel, encPos, encVel;

// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
    CAboutDlg();

    // Dialog Data
    enum { IDD = IDD_ABOUTBOX };

protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support

    // Implementation
protected:
    DECLARE_MESSAGE_MAP()
}
```

图25

13、在窗体的属性视图添加消息处理函数，这里我们双击红框，加入定时器，以便获取规划位置、规划速度等变量的实时变化，如图26

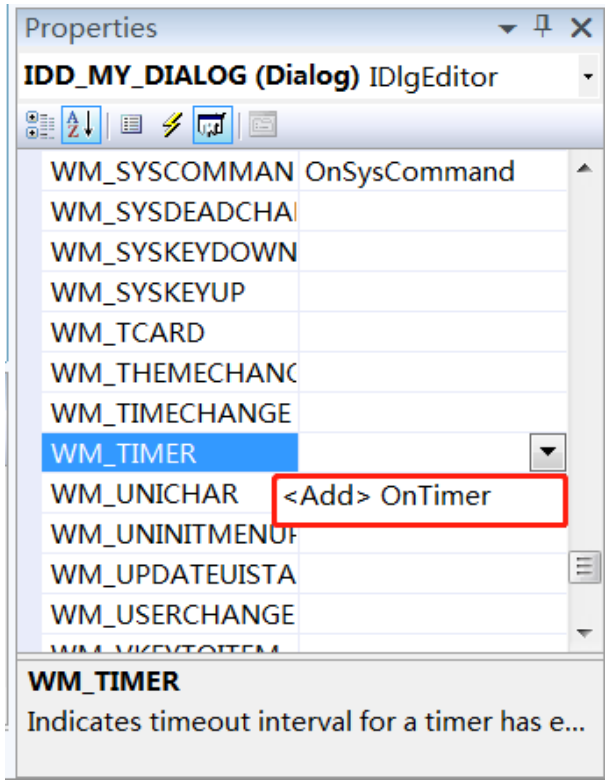


图26

接着会自动进入定时器代码编辑页面，

在void C点位运动Dlg:: OnTimer(UINT\_PTR nIDEvent)

```
{  
    // TODO: Add your control notification handler code here
```

}中输入以下代码，如图27所示

```
void C点位运动Dlg::OnTimer(UINT_PTR nIDEvent)  
{  
    GT_GetPrfPos(1,&prfPos,1,NULL);           // 读1轴的规划位置  
    strTemp.Format(_T("%.3f"),prfPos);  
    SetDlgItemText(IDC_STATIC1,strTemp);        //IDC_STATIC1为规划位置右侧静态文本的ID  
  
    GT_GetPrfVel(1,&prfVel,1,NULL);           // 读1轴的规划速度  
    strTemp.Format(_T("%.3f"),prfPos);  
    SetDlgItemText(IDC_STATIC2,strTemp);  
  
    GT_GetEncPos(1,&encPos,1,NULL);           // 读1轴的实际位置  
    strTemp.Format(_T("%.3f"),prfPos);  
    SetDlgItemText(IDC_STATIC3,strTemp);  
  
    GT_GetEncVel(1,&encVel,1,NULL);           // 读1轴的实际速度  
    strTemp.Format(_T("%.3f"),prfPos);  
    SetDlgItemText(IDC_STATIC4,strTemp);  
  
    CDialog::OnTimer(nIDEvent);  
}
```

图27

14、然后双击初始化按钮，会自动进入按钮的代码编辑页面，  
如图28

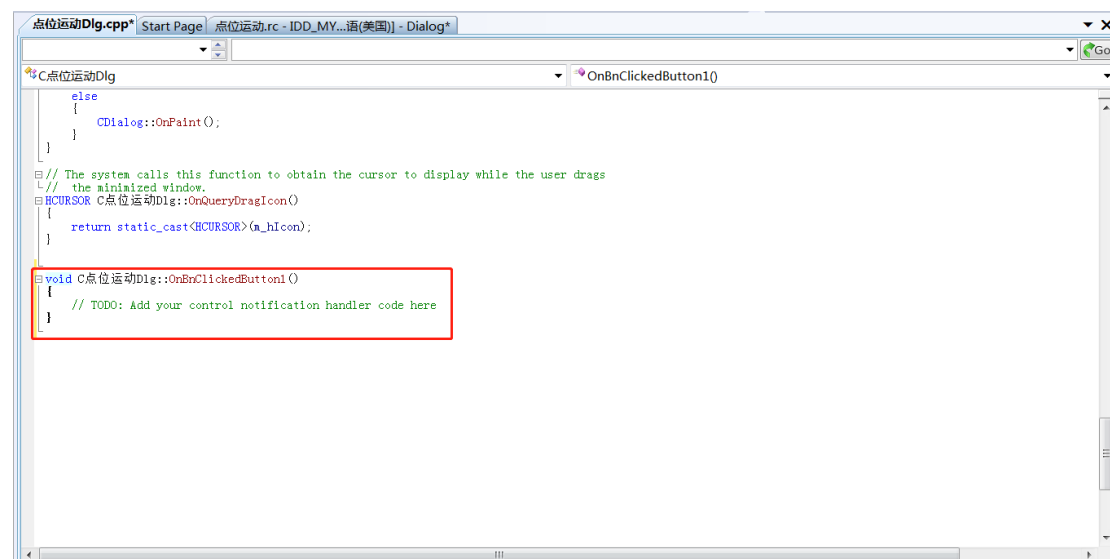


图28

在void C点位运动Dlg::OnBnClickedButton1()

```
{  
    // TODO: Add your control notification handler code here
```

}中输入以下代码，如图27所示

```
void C点位运动Dlg::OnBnClickedButton1()  
{  
    // TODO: Add your control notification handler code here  
  
    sRtn = GT_Open();           // 打开运动控制器  
    sRtn = GT_Reset();          // 复位运动控制器  
    sRtn = GT_LoadConfig("GTS800.cfg"); // 下载控制器配置文件  
    sRtn = GT_ClrSts(1,1);      // 清除1轴的轴状态  
    SetTimer(1,10,NULL);        // 设置定时器  
}
```

图27

15、然后依次双击所有按钮，按照步骤 14 的方法分别在每个按钮的代码编辑页面添加代码，如图 28，29

```
sRtn = GT_Open();           // 打开运动控制器  
sRtn = GT_Reset();          // 复位运动控制器  
sRtn = GT_LoadConfig("GTS800.cfg"); // 下载控制器配置文件  
sRtn = GT_ClrSts(1,1);      // 清除1轴的轴状态  
SetTimer(1,10,NULL);        // 设置定时器  
}  
  
void C点位运动Dlg::OnBnClickedButton2()  
{  
    sRtn = GT_ClrSts(1,1);    // 清除1轴的轴状态  
}  
  
void C点位运动Dlg::OnBnClickedButton3()  
{  
    sRtn = GT_AxisOn(1);      // 使能运动轴  
}  
  
void C点位运动Dlg::OnBnClickedButton4()  
{  
    sRtn = GT_ZeroPos(1,1);   // 位置清零  
}  
  
void C点位运动Dlg::OnBnClickedButton5()  
{  
    UpdateData(TRUE);         // 将各控件中的数据保存到相应的变量  
    sRtn = GT_SetPrfPos(1,0);  // 将AXIS轴设为点位模式  
    sRtn = GT_PrflTrap(1);     // 读取点位运动参数  
    TTrapPrm trap;  
    sRtn = GT_GetTrapPrm(1, &trap);  
    trap.acc = 0.1;  
    trap.dec = 0.1;  
    trap.smoothTime = 1;       // 设置点位运动参数  
    sRtn = GT_SetTrapPrm(1, &trap); // 设置AXIS轴的目标位置  
    sRtn = GT_SetPos(1, m_editPos); // 设置AXIS轴的目标速度  
    sRtn = GT_SetVel(1, m_editVel); // 启动AXIS轴的运动  
    sRtn = GT_Update(1<<0);  
}  
  
void C点位运动Dlg::OnBnClickedButton6()  
{  
    sRtn = GT_Stop(1,1);      // 停止运动  
}
```

图28

16、检查代码没有错误后，开始调试运行按F5，或者点击如图所示按钮，如图29

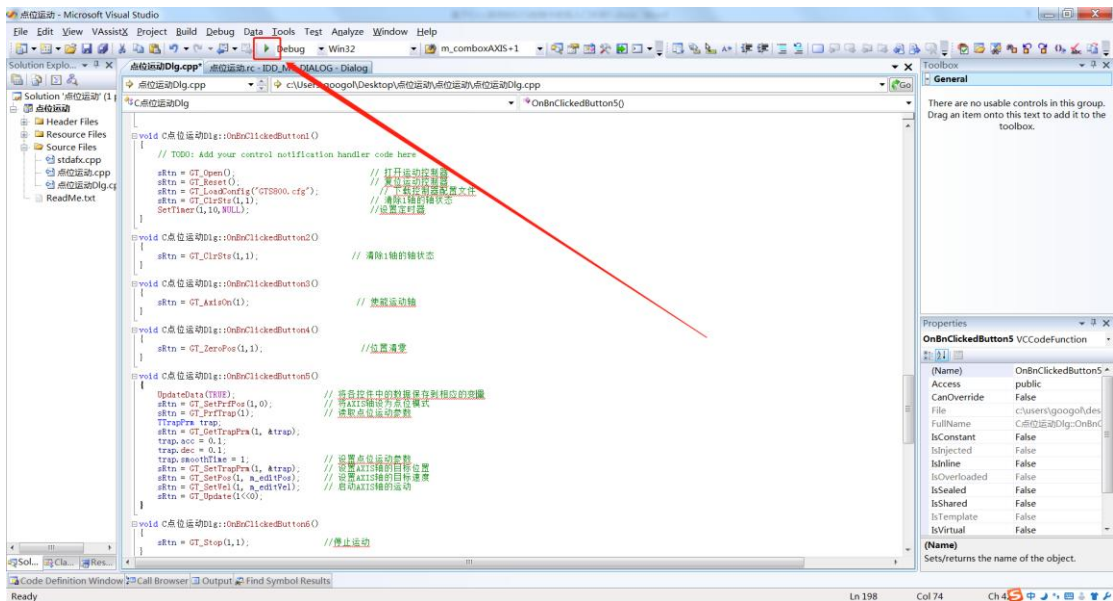


图29

17、输入点位距离和速度，然后点击开始运动按钮，如图30



图30

18、运动过程中，规划位置和实际位置会实时变化，点击停止按钮可停止点位运动。点击×号可以关闭整个调试程序。