回零功能模块制作

08

回零功能模块制作

“

编辑界面内控件的属性，见表”回零界面控件属性表”。

08

回零功能模块制作

“

编辑界面内控件的属性，见表”回零界面控件属性表”。

控件类型	ID	描述文字	组
Group Box	IDC_STATIC	选择轴号	
Group Box	IDC_STATIC	回零启动方向	
Group Box	IDC_STATIC	捕获触发沿	
Radio Button	IDC_H_Axis1	一轴	True
Radio Button	IDC_H_Axis2	二轴	
Radio Button	IDC_H_Axis3	三轴	
Radio Button	IDC_H_Axis4	四轴	
Radio Button	IDC_H_StartNegative	负方向	True
Radio Button	IDC_H_StartPositive	正方向	
Radio Button	IDC_H_fallingEdge	下降沿	True
Radio Button	IDC_H_risingEdge	上升沿	
Static Text	IDC_STATIC	搜索速度	
Static Text	IDC_STATIC	偏置速度	
Static Text	IDC_STATIC	加速度	
Static Text	IDC_STATIC	减速度	
Static Text	IDC_STATIC	零点偏置	
Static Text	IDC_STATIC	搜索距离	
Static Text	IDC_STATIC	反向步长	
Static Text	IDC_STATIC	回零状态	
Static Text	IDC_STATIC	当前阶段	
Static Text	IDC_STATIC	错误提示	
Static Text	IDC_STATIC	捕获位置	
Static Text	IDC_STATIC	pulse	
Static Text	IDC_STATIC	pulse/ms	
Static Text	IDC_H_state	停止运动	
Static Text	IDC_H_step	未启动回原点	
Static Text	IDC_H_error	未发生错误	
Static Text	IDC_H_capturePos	0	
Edit Control	IDC_H_searchVel		
Edit Control	IDC_H_localizationVel		
Edit Control	IDC_H_acc		
Edit Control	IDC_H_dec		
Edit Control	IDC_H_offset		
Edit Control	IDC_H_homeSearch		
Edit Control	IDC_H_escapeStep		
Button	IDC_H_AxisOn	启动伺服	
Button	IDC_H_AxisOff	关闭伺服	
Button	IDC_H_start	点位运动启动	
Button	IDC_H_stop	停止运动	
Button	IDC_H_clrsts	清除报警	
Button	IDC_H_zeroPos	位置清零	

08

回零功能模块制作

“

单击键盘的Ctrl+D，显示控件的Tab顺序，通过单击数字按钮重新整理Tab顺序，如图所示。

08

回零功能模块制作

2. 读取界面参数

参考制作单轴运动章节，从界面上获取“选择轴号”、“回零启动方向”和“捕获触发沿”的选项参数。

按下表添加控制变量及事件函数。

控件	名称	事件BN_CLICKED
IDC_H_Axis1	m_H_axis	OnBnClickedHAxis1
IDC_H_StartNegative	m_H_startDirection	OnBnClickedHStartnegative
IDC_H_fallingEdge	m_H_edge	OnBnClickedHfallingedge

08

回零功能模块制作

在Home_Page.cpp中添加代码

```
BEGIN_MESSAGE_MAP(Home_Page, CDialogEx)
        ON_BN_CLICKED(IDC_H_Axis1, &Home_Page::OnBnClickedHAxis1)
        ON_BN_CLICKED(IDC_H_Axis2, &Home_Page::OnBnClickedHAxis1)
        ON_BN_CLICKED(IDC_H_Axis3, &Home_Page::OnBnClickedHAxis1)
        ON_BN_CLICKED(IDC_H_Axis4, &Home_Page::OnBnClickedHAxis1)
        ON_BN_CLICKED(IDC_H_StartNegative, &Home_Page::OnBnClickedHStartnegative)
        ON_BN_CLICKED(IDC_H_StartPositive, &Home_Page::OnBnClickedHStartnegative)
        ON_BN_CLICKED(IDC_H_fallingEdge, &Home_Page::OnBnClickedHfallingedge)
        ON_BN_CLICKED(IDC_H_risingEdge, &Home_Page::OnBnClickedHfallingedge)
END_MESSAGE_MAP()
// Home_Page 消息处理程序
void Home_Page::OnBnClickedHAxis1()
{
    // TODO: 在此添加控件通知处理程序代码
    UpdateData(TRUE);
}
void Home_Page::OnBnClickedHStartnegative()
{
    // TODO: 在此添加控件通知处理程序代码
    UpdateData(TRUE);
}
void Home_Page::OnBnClickedHfallingedge()
{
    // TODO: 在此添加控件通知处理程序代码
    UpdateData(TRUE);
}
```

08

回零功能模块制作

3. 添加头文件

添加gts.h头文件引用，使用控制卡指令函数按表13-3添加控制变量及事件函数。

```
#include"gts.h"
```

4. 启动伺服

添加启动伺服按钮功能。

```
voidHome_Page::OnBnClickedHAxison()
{
    // TODO: 在此添加控件通知处理程序代码
    short sRtn;
    sRtn = GT_AxisOn(m_H_axis+1);
    CMotionControlDemoApp::commandhandler(_T("GT_AxisOn"), sRtn);
}
```

08

回零功能模块制作

“

5. 关闭伺服

添加关闭伺服按钮功能。

```
voidHome_Page::OnBnClickedHAxisoff()
{
    // TODO: 在此添加控件通知处理程序代码
    short sRtn;
    sRtn = GT_AxisOff(m_H_axis + 1);
    CMotionControlDemoApp::commandhandler(_T("GT_AxisOff"), sRtn);
}
```

08

回零功能模块制作

“

6. 实现回零

添加启动回零按钮功能，此处以限位+Home原点方式回零为例进行说明。

```
void Home_Page::OnBnClickedHstart()
{
	short sRtn;
	THomePrm tHomePrm;
	CString str;
	short axis = m_H_axis + 1;		//回零轴号
	//读取回零参数
	sRtn = GT_GetHomePrm(axis, &tHomePrm);
	CMotionControlDemoApp::commandhandler(_T("GT_GetHomePrm"), sRtn);
	//设置回零模式
	if (axis < 4)
	{
	  tHomePrm.mode = HOME_MODE_LIMIT_HOME;
	}
	else
	{
	  tHomePrm.mode = HOME_MODE_LIMIT;
	}
	//设置回零启动方向
	if (m_H_startDirection == 0)
	{
	  tHomePrm.moveDir = -1;
	}
```

08 回零功能模块制作

```
    else
    {
      tHomePrm.moveDir = 1;
    }
    tHomePrm.edge = m_H_edge;          //设置捕获触发沿
    //检测当前是否在限位或原点
    tHomePrm.pad2[0] = 1;
    tHomePrm.pad2[1] = 1;
    tHomePrm.pad2[2] = 1;
    //回零搜索速度
    GetDlgItem(IDC_H_searchVel)->GetWindowTextW(str);
    tHomePrm.velHigh = _wtof(str.GetBuffer());
    //回零偏置速度
    GetDlgItem(IDC_H_localizationVel)->GetWindowTextW(str);
    tHomePrm.velLow = _wtof(str.GetBuffer());
    //回零加速度
    GetDlgItem(IDC_H_acc)->GetWindowTextW(str);
    tHomePrm.acc = _wtof(str.GetBuffer());
    //回零减速度
    GetDlgItem(IDC_H_dec)->GetWindowTextW(str);
    tHomePrm.dec = _wtof(str.GetBuffer());
    tHomePrm.smoothTime = 10;          //平滑时间
```

```
    //平滑时间
    tHomePrm.smoothTime = 10;
    //零点偏置
    tHomePrm.homeOffset =
GetDlgItemInt(IDC_H_offset, NULL, 1);
    //设置Home搜索范围
    tHomePrm.searchHomeDistance =
GetDlgItemInt(IDC_H_homeSearch, NULL, 1);
    //设置采用“限位回原点”方式时，反方向离开限
位的脱离步长
    tHomePrm.escapeStep =
GetDlgItemInt(IDC_H_escapeStep, NULL, 1);

    //启动Smart Home回原点
    sRtn = GT_GoHome(axis, &tHomePrm);
    CMotionControlDemoApp::commandhandler(_T("GT
_GoHome"), sRtn);
}
```

08

回零功能模块制作

7. 停止回零

添加停止回零按钮功能。

```
void Home_Page::OnBnClickedHstop()
{
	// TODO: 在此添加控件通知处理程序代码
	short sRtn;
	sRtn = GT_Stop(15, 0);
	CMotionControlDemoApp::commandhandler(_T("GT_Stop"), sRtn);
}
```

8. 清除报警

添加清除报警按钮功能。

```
void Home_Page::OnBnClickedHclrsts()
{
	// TODO: 在此添加控件通知处理程序代码
	short sRtn;
	sRtn = GT_ClrSts(m_H_axis + 1, 1);
	CMotionControlDemoApp::commandhandler(_T("GT_ClrSts"), sRtn);
}
```

08

回零功能模块制作

9. 位置清零

在回零运动结束后，轴会运动到机械零点，但控制器中的计数器并没有清零，此时需要使计数器清零，以方便后续运动的位置计算。

添加位置清零按钮功能。

```
void Home_Page::OnBnClickedHzeropos()
{
    // TODO: 在此添加控件通知处理程序代码;   shortsRtn;
    sRtn = GT_ZeroPos(m_H_axis + 1, 1);
    CMotionControlDemoApp::commandhandler(_T("GT_ZeroPos"), sRtn);
}
```

08

回零功能模块制作

10. 监控回零状态

（1）定义消息ID、对话框句柄和消息处理函数,添加一个线程实时监控轴的回零状态。

在Home_Page.h中定义一个线程，添加在public下。

```
static UINT MotionStatusThread(LPVOIDpParam);//状态检测线程
```

添加自定义消息ID，命名为WM_MotionStatus_MESSAGE。

```
#pragmaonce
#define WM_HomeStatus_MESSAGEWM_USER+101 //回零状态消息
 // Home_Page 对话框
```

定义对话框句柄。

```
public:
        Home_Page(CWnd* pParent= nullptr); // 标准构造函数
        virtual~Home_Page();
        staticHWNDHome_Page::m_hWndHome;//定义对话框句柄
```

08

回零功能模块制作

然后定义消息处理函数，可添加在DECLARE_MESSAGE_MAP()上方。代码如下。

```
afx_msgLRESULTOnHomeStatusMessage (WPARAMwParam, LPARAMlParam); //控制卡状态监测消息处理
```

（2）引入对话框句柄

在Home_Page.cpp中，引入对话框句柄。

```
#include"gts.h"
HWNDHome_Page::m_hWndHome = 0;//对话框句柄
// Home_Page 对话框
```

08

回零功能模块制作

（3）读取回零状态

在Home_Page.cpp中，新建函数HomeStatusThread，编写回零状态读取工作。

```
UINT Home_Page::HomeStatusThread(LPVOID pParam)
{
	THomeStatus mHomeStatus[4];	//传递回零状态数组
	while (1)
	{
	  for (int i = 1; i < 5; i++)	//循环读取1-4轴回原点状态
	  {
	    GT_GetHomeStatus(i, &mHomeStatus[i - 1]);
	  }
	  //获取对话框句柄
	  CWnd* m_hWndHome = CWnd::FromHandle(Home_Page::m_hWndHome);
	  //传递状态至界面线程
	  ::PostMessage(m_hWndHome->GetSafeHwnd(), WM_HomeStatus_MESSAGE,
(WPARAM)mHomeStatus, NULL);
	  Sleep(300); //延时300毫秒防止界面卡死
	}
	return 0;
}
```

08

回零功能模块制作

（4）添加OnInitDalog函数

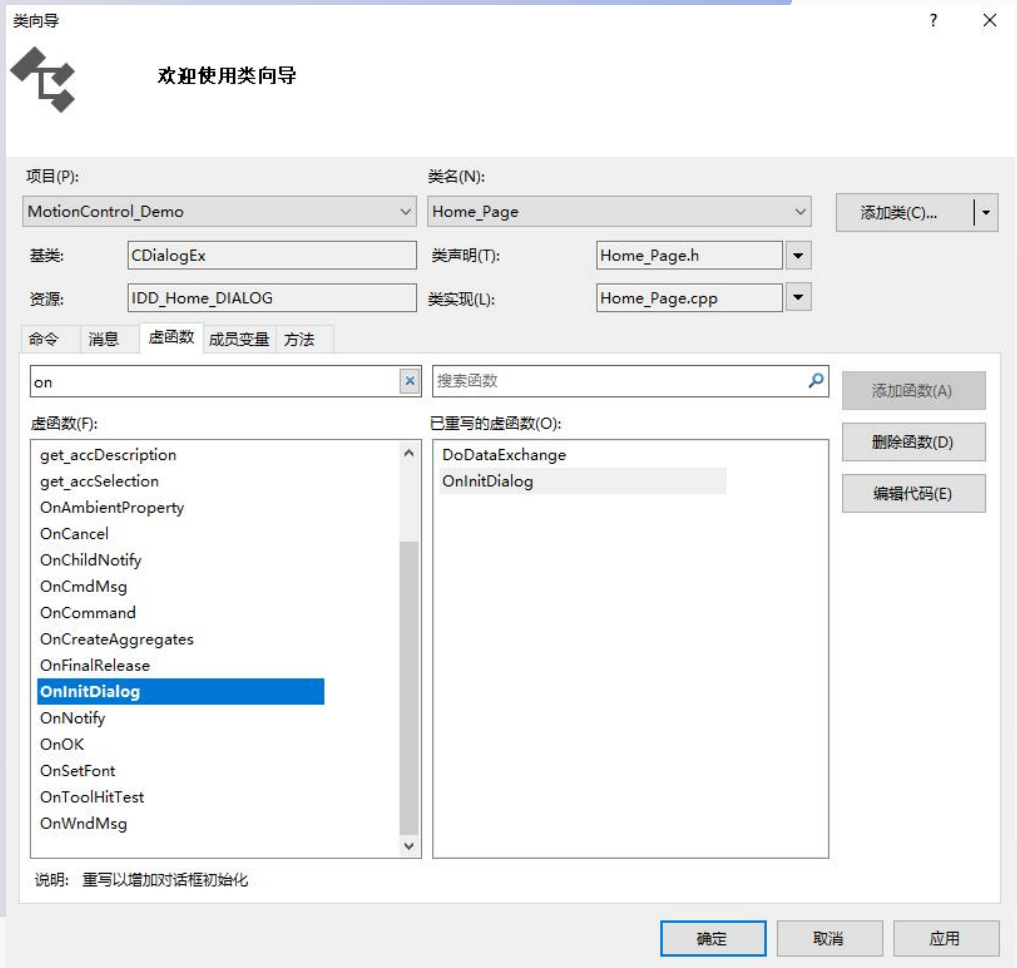
在资源视图中打开IDD_Home_DIALOG，单击鼠标右键，选择“类向导”，如图所示。

08

回零功能模块制作

“

在类向导中，选择虚函数，添加OnInitDalog函数，如图所示。



08

回零功能模块制作

“

（5）启动状态监测

在Home_Page.cpp中，找到OnInitDialog()函数，添加获取对话框句柄和启动状态监测线程。

```
BOOLHome_Page::OnInitDialog()
{
    CDialogEx::OnInitDialog();      // TODO: 在此添加额外的初始化
    Home_Page::m_hWndHome = this->m_hWnd;//获取对话框句柄
    AfxBeginThread((AFX_THREADPROC)HomeStatusThread, (VOID*)this, THREAD_PRIORITY_NORMAL, 0,
0, NULL);
    //启动状态监控线程
    returnTRUE; // return TRUE unless you set the focus to a control                    //
异常: OCX 属性页应返回FALSE
}
```

08

回零功能模块制作

（6）界面显示

新建函数OnHomeStatusMessage()作为消息处理函数，把监测线程中传递过来的数据显示在界面上。

```
//更新状态至界面
LRESULT Home_Page::OnHomeStatusMessage(WPARAM wParam, LPARAM lParam)
{
	THomeStatus* mAHomeStatus = (THomeStatus*)wParam;
	CString str;
	// 是否正在进行回原点，0—已停止运动，1-正在回原点
	if (mAHomeStatus[m_H_axis].run == 1)
	{
	  SetDlgItemText(IDC_H_state, _T("正在回原点"));
	}
	else
	{
	  SetDlgItemText(IDC_H_state, _T("已停止运动"));
	}
	// 回原点运动的阶段
	switch (mAHomeStatus[m_H_axis].stage)
	{
	  case HOME_STAGE_IDLE:
	    SetDlgItemText(IDC_H_step, _T("未启动回原点"));
	    break;
	  case HOME_STAGE_START:
	    SetDlgItemText(IDC_H_step, _T("启动回原点"));
	    break;
	  case HOME_STAGE_SEARCH_LIMIT:
	    SetDlgItemText(IDC_H_step, _T("正在寻找限位"));
	    break;
	  case HOME_STAGE_SEARCH_LIMIT_STOP:
	    SetDlgItemText(IDC_H_step, _T("触发限位停止"));
	    break;
	  case HOME_STAGE_SEARCH_LIMIT_ESCAPE:
	    SetDlgItemText(IDC_H_step, _T("反方向运动脱离限位"));
	    break;
	  case HOME_STAGE_SEARCH_LIMIT_RETURN:
	    SetDlgItemText(IDC_H_step, _T("重新回到限位"));
	    break;
	  case HOME_STAGE_SEARCH_LIMIT_RETURN_STOP:
	    SetDlgItemText(IDC_H_step, _T("重新回到限位停止"));
	    break;
	  case HOME_STAGE_SEARCH_HOME:
	    SetDlgItemText(IDC_H_step, _T("正在搜索Home"));
	    break;
	  case HOME_STAGE_SEARCH_HOME_RETURN:
	    SetDlgItemText(IDC_H_step, _T("搜索到Home后运动到捕获的Home
位置"));
	    break;
	  case HOME_STAGE_SEARCH_INDEX:
	    SetDlgItemText(IDC_H_step, _T("正在搜索Index"));
	    break;
	  case HOME_STAGE_GO_HOME:
	    SetDlgItemText(IDC_H_step, _T("正在执行回原点过程"));
	    break;
```

08

回零功能模块制作

```
        case HOME_STAGE_END:
            SetDlgItemText(IDC_H_step, _T("回原点结束"));
            break;
        }
        //回原点错误
        switch (mAHomeStatus[m_H_axis].error)
        {
            case HOME_ERROR_NONE:
                SetDlgItemText(IDC_H_error, _T("未发生错误"));
                break;
            case HOME_ERROR_NOT_TRAP_MODE:
                SetDlgItemText(IDC_H_error, _T("执行Smart Home回原点的轴不是处于点位运动模式"));
                break;
            case HOME_ERROR_DISABLE:
                SetDlgItemText(IDC_H_error, _T("执行Smart Home回原点的轴未使能"));
                break;
            case HOME_ERROR_ALARM:
                SetDlgItemText(IDC_H_error, _T("执行Smart Home回原点的轴驱动报警"));
                break;
            case HOME_ERROR_STOP:
                SetDlgItemText(IDC_H_error, _T("未完成回原点，轴停止运动"));
                break;
            case HOME_ERROR_STAGE:
                SetDlgItemText(IDC_H_error, _T("回原点阶段错误"));
                break;
            case HOME_ERROR_HOME_MODE:
                SetDlgItemText(IDC_H_error, _T("模式错误"));
                break;
            case HOME_ERROR_SET_CAPTURE_HOME:
                SetDlgItemText(IDC_H_error, _T("设置Home捕获模式失败"));
                break;
            case HOME_ERROR_NO_HOME:
                SetDlgItemText(IDC_H_error, _T("未找到Home"));
                break;
            case HOME_ERROR_SET_CAPTURE_INDEX:
                SetDlgItemText(IDC_H_error, _T("设置Index捕获模式失败"));
                break;
            case HOME_ERROR_NO_INDEX:
                SetDlgItemText(IDC_H_error, _T("未找到Index"));
                break;
        }
        // 捕获到Home或Index时刻的编码器位置
        str.Format(_T("%ld"), mAHomeStatus[m_H_axis].capturePos);
        SetDlgItemText(IDC_H_capturePos, str);
        return LRESULT();
}
```

08

回零功能模块制作

“

（7）消息映射

把消息ID和处理函数关联起来，这就是消息映射，同样是在主类中操作，找到MESSAGE_MAP，在BEGIN_MESSAGE_MAP(Home_Page, CDialogEx)内写入关联代码。

```
ON_MESSAGE(WM_HomeStatus_MESSAGE, &Home_Page::OnHomeStatusMessage)//回零状态监测，消息ID关联处理函数
```

谢　谢　观　看