

博瑞电流采样自控软件 V1.0

```
#include "stdafx.h"
#include "WiseCut.h"
#include "ZAxisView.h"
#include "math.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] __FILE__;
#endif
extern double xpos, ypos, zpos, apos;
IMPLEMENT_DYNCREATE(CZAxisView, CView)
CZAxisView::CZAxisView()
{
}
CZAxisView::~CZAxisView()
{
}
BEGIN_MESSAGE_MAP(CZAxisView, CView)
    {{AFX_MSG_MAP(CZAxisView)
        ON_WM_PAINT()
        ON_WM_ERASEBKGD()
    }}AFX_MSG_MAP
END_MESSAGE_MAP()
CZAxisView drawing
void CZAxisView::OnDraw(CDC* pDC)
{
    CDocument* pDoc  GetDocument();
    TODO: add draw code here
}
CZAxisView diagnostics
#ifdef _DEBUG
void CZAxisView::AssertValid() const
{
    CView::AssertValid();
}
void CZAxisView::Dump(CDumpContext& dc) const
{
    CView::Dump(dc);
}
#endif _DEBUG
CZAxisView message handlers
void CZAxisView::OnPaint()
{
    CPaintDC dc(this);  device context for painting
    GetClientRect(&m_ZViewRect);
    CMemDC memDC(&dc, &m_ZViewRect);
    if(m_dcBackground.GetSafeHdc() NULL|| (m_bitmapBackground.m_hObject  NULL))
    {
        m_dcBackground.CreateCompatibleDC(&dc);
```

---

```

        m_bitmapBackground.CreateCompatibleBitmap(&dc,m_ZViewRect.Width(),m_ZView
Rect.Height());
        m_pBitmapOldBackground
m_dcBackground.SelectObject(&m_bitmapBackground);
        DrawMeterBackground(&m_dcBackground, rect);
        CBrush brushFill, *pBrushOld;
        brushFill.DeleteObject();
        brushFill.CreateSolidBrush(RGB(255, 255, 255));
        pBrushOld m_dcBackground.SelectObject(&brushFill);
        m_dcBackground.Rectangle(m_ZViewRect);
        m_dcBackground.SelectObject(pBrushOld);
    }
    memDC.BitBlt(0, 0, m_ZViewRect.Width(), m_ZViewRect.Height(),
        &m_dcBackground, 0, 0, SRCCOPY) ;
    TODO: Add your message handler code here
    CRect rect;
    CPoint pt;
    pt m_ZViewRect.CenterPoint();
    int ndim 60;
    rect.bottom pt.y + ndim;
    rect.top pt.y - ndim;
    rect.left pt.x - ndim;
    rect.right pt.x + ndim;
    memDC.Ellipse(rect);
    COLORREF Color;
    CPen *pPen;
    Color RGB(0,255,0);
    CPen pen(PS_SOLID, 2, Color);
    pPen memDC.SelectObject(&pen);
    memDC.MoveTo(rect.right, pt.y);
    memDC.LineTo(rect.right + 5, pt.y );
    memDC.TextOut(rect.right + 10, pt.y - 7, "0");
    memDC.MoveTo(rect.left, pt.y);
    memDC.LineTo(rect.left - 5, pt.y );
    memDC.TextOut(rect.left - 30, pt.y - 7, "180");
    memDC.MoveTo(pt.x, rect.top);
    memDC.LineTo(pt.x, rect.top - 5);
    memDC.TextOut(pt.x - 7, rect.top - 20, "90");
    memDC.MoveTo(pt.x, rect.bottom);
    memDC.LineTo(pt.x, rect.bottom + 5);
    memDC.TextOut(pt.x - 10, rect.bottom + 7, "270");
    pen.DeleteObject();
    Color RGB(0,0,255);
    pen.CreatePen(PS_SOLID, 1, Color);
    pPen memDC.SelectObject(&pen);
    ndim 20;
    rect.bottom pt.y + ndim;
    rect.top pt.y - ndim;
    rect.left pt.x - ndim;

```

---

```

    rect.right  pt.x + ndim;
    memDC.Ellipse(rect);
    double dCurPos  3.14159;
    double angle  zpos10;
    memDC.MoveTo(pt.x + ndim*cos(angle), pt.y - ndim*sin(angle));
    memDC.LineTo(pt.x + 40*cos(angle), pt.y - 40*sin(angle));
    pen.DeleteObject();
    memDC.MoveTo(rect.bottom, rect.left);
    memDC.LineTo(rect.top, rect.right);
    Do not call CView::OnPaint() for painting messages
}
void CZAxisView::OnInitialUpdate()
{
    CView::OnInitialUpdate();
    TODO: Add your specialized code here andor call the base class
    m_pFr  (CMainFrame*)AfxGetApp()->m_pMainWnd;
}
BOOL CZAxisView::OnEraseBkgnd(CDC* pDC)
{
    TODO: Add your message handler code here andor call default
    return TRUE;
    return CView::OnEraseBkgnd(pDC);
}
#include "stdafx.h"
#include "WiseCut.h"
#include "WiseCutDoc.h"
#include "WiseCutView.h"
#include "MainFrm.h"
#include "Dfjzh6030Api.h"
#include "dfjzh6050dll.h"
#include "math.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[]  __FILE__;
#endif
const int X_OFF  0;
const int Y_OFF  0;
const int Z_OFF  0;
const double PI  3.1415926536;
const double mil_PI  0.0005;  PI360 0.0087266, 2*PI(250*r10)  0.0005026
const int size  10000;
const float SIZERATIO  1;
GCmdLine fPosition[size];
UINT iNodeLength  0;
float fZoomSize  3.0;
int iXoffset  0;
int iYoffset  0;
int iZoffset  0;
extern double xpos,ypos,zpos,apos;

```

---

```

extern int axis_X; 0;
extern int axis_Y; 1;
extern int axis_Z; 2;
extern int axis_W;
extern double LenPerPulse; 0.0025;
extern double Accuracy; 0.03;
CEvent gDrawLineEventKill;
IMPLEMENT_DYNCREATE(CWiseCutView, CView)
BEGIN_MESSAGE_MAP(CWiseCutView, CView)
    {{AFX_MSG_MAP(CWiseCutView)
        ON_WM_PAINT()
    }}AFX_MSG_MAP
    Standard printing commands
    ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_DIRECT, CView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)
END_MESSAGE_MAP()
CWiseCutView::CWiseCutView()
{
    TODO: add construction code here
}
CWiseCutView::~CWiseCutView()
{
}
BOOL CWiseCutView::PreCreateWindow(CREATESTRUCT& cs)
{
    TODO: Modify the Window class or styles here by modifying
    the CREATESTRUCT cs
    return CView::PreCreateWindow(cs);
}
void CWiseCutView::OnDraw(CDC* pDC)
{
    CWiseCutDoc* pDoc GetDocument();
    ASSERT_VALID(pDoc);
    CRect rect;
    GetClientRect(&rect);
    int nWidth rect.Width();
    int nHeight rect.Height();
    COLORREF Color;
    CPen *pPen;
    Color RGB(0,255,0);
    CPen pen(PS_SOLID, 1, Color);
    pPen pDC->SelectObject(&pen);
    pDC->MoveTo(20,20);
    pDC->LineTo(20,nHeight-20);
    pDC->LineTo(nWidth-20, nHeight-20);
    pDC->LineTo(nWidth-20, 20);
    pDC->LineTo(20,20);
    pen.DeleteObject();
    this->DrawSpindle(xpos,ypos);
}

```

---

```

        this->ParseCNCDData(m_strFileName);
    }
    BOOL CwiseCutView::OnPreparePrinting(CPrintInfo* pInfo)
    {
        default preparation
        return DoPreparePrinting(pInfo);
    }
    void CwiseCutView::OnBeginPrinting(CDC* *pDC*, CPrintInfo* *pInfo*)
    {
        TODO: add extra initialization before printing
    }
    void CwiseCutView::OnEndPrinting(CDC* *pDC*, CPrintInfo* *pInfo*)
    {
        TODO: add cleanup after printing
    }
#ifdef _DEBUG
    void CwiseCutView::AssertValid() const
    {
        CView::AssertValid();
    }
    void CwiseCutView::Dump(CDumpContext& dc) const
    {
        CView::Dump(dc);
    }
    CwiseCutDoc* CwiseCutView::GetDocument()    non-debug version is inline
    {
        ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CwiseCutDoc)));
        return (CwiseCutDoc*)m_pDocument;
    }
#endif _DEBUG
    void CwiseCutView::DrawSpindle(double x, double y)
    {
        CDC *pDC  GetDC();
        pDC->Ellipse(x-1, y-1, x+1, y+1);
    }
    UINT CwiseCutView::DrawLine(LPVOID lp)
    {
        CwiseCutView* pWiseCutView;
        pWiseCutView  (CwiseCutView*)lp;
        float Xpos 0.0F, Ypos  0.0F;
        float oldxpos,oldypos,oldzpos;
        POINT ptTemp;
        CDC* pDC  pWiseCutView->GetDC();
        COLORREF Color;
        Color  RGB(0,0,255);
        CPen pen(PS_DASH, 3, Color );
        CPen* pPen  pDC->SelectObject(&pen);
        oldxpos  actpos6030(axis_X) * LenPerPulse * 10;
        oldypos  actpos6030(axis_Y) * LenPerPulse * 10;
        oldzpos  actpos6030(axis_Z) * LenPerPulse * 10;
    }

```

---

```

Xpos  actpos6030(axis_X) * LenPerPulse * fZoomSize  + iXoffset ;
Ypos  actpos6030(axis_Y) * LenPerPulse * fZoomSize  + iYoffset ;
ptTemp.x  Xpos;
ptTemp.y  Ypos;
pDC->MoveTo(ptTemp);
while(TRUE)
{
    Len  LM_GetBufferLen6030();
    Sleep(25);
    xpos  actpos6030(axis_X) * LenPerPulse * 10;
    ypos  actpos6030(axis_Y) * LenPerPulse * 10;
    zpos  actpos6030(axis_Z) * LenPerPulse * 10;
    if(fabs(oldxpos -xpos) > 0.001)
    {
        pWiseCutView->m_pFr->m_pStatusView->ShowXCoordinate();
    }
    if(fabs(oldypos - ypos) > 0.001)
    {
        pWiseCutView->m_pFr->m_pStatusView->ShowYCoordinate();
    }
    if(fabs(oldzpos - zpos) > 0.001)
    {
        pWiseCutView->m_pFr->m_pStatusView->ShowZCoordinate();
    }
    pWiseCutView->m_pFr->m_pZAxisView->Invalidate();
    oldxpos  xpos;
    oldypos  ypos;
    oldzpos  zpos;
    Xpos  actpos6030(axis_X) * LenPerPulse * fZoomSize  + iXoffset ;
    Ypos  actpos6030(axis_Y) * LenPerPulse * fZoomSize  + iYoffset ;
    ptTemp.x  Xpos;
    ptTemp.y  Ypos;
    pDC->LineTo(Xpos,Ypos);
    if( ::WaitForSingleObject(gDrawLineEventKill, 0)  WAIT_OBJECT_0 )
    {
        break;
    }
}
return 0;
}

UINT CWiseCutView::DrawLine_6050(LPVOID lp)
{
    CWiseCutView* pWiseCutView;
    pWiseCutView  (CWiseCutView*)lp;
    double Xpos 0.0F, Ypos  0.0F;
    POINT ptTemp;
    CDC* pDC  pWiseCutView->GetDC();
    COLORREF Color;
    Color  RGB(0,0,255);
    CPen pen(PS_SOLID, 1, Color );

```

---

```

        CPen* pPen  pDC->SelectObject(&pen);
        Xpos  ReadAxisPos_6050(0, axis_X) * fZoomSize * LenPerPulse + X_OFF ;
        Ypos  ReadAxisPos_6050(0, axis_Y) * fZoomSize * LenPerPulse + Y_OFF ;
        ptTemp.x  (long)Xpos;
        ptTemp.y  (long)Ypos;
        pDC->MoveTo(ptTemp);
        while(TRUE)
        {
Xpos  ReadAxisPos_6050(0, axis_X) * fZoomSize * LenPerPulse + X_OFF ;
        Ypos  ReadAxisPos_6050(0, axis_Y) * fZoomSize * LenPerPulse + Y_OFF ;
                ptTemp.x  Xpos;
                ptTemp.y  Ypos;
                pDC->LineTo(Xpos,Ypos);
                if( ::WaitForSingleObject(gDrawLineEventKill, 0)  WAIT_OBJECT_0 )
                {
                        break;
                }
        }
        return 0;
}

void CWiseCutView::draw_ellipse()
{
        double x_c  428.642207113819;
        double y_c  321.503012698986;
        double z_c  0.0;
        double x_end_major  30.1474702235905;
        double y_end_major  70.1062787003219;
        double z_end_major  0.0;
        double ratio  0.501665766070093;
        double start  2.06178120457755;
        double end  4.64634969578438;
        double ang0;
        double ang1;
        double a,b;
        if(fabs(y_end_major) > fabs(x_end_major))
        {
                b  sqrt(pow((z_end_major),2) +
                        pow((y_end_major),2) +
                        pow((x_end_major),2) );
                a  fabs(b * ratio);
        }
        {
                a  sqrt(pow((z_end_major),2) +
                        pow((y_end_major),2) +
                        pow((x_end_major),2) );
                b  fabs(a * ratio);
        }
        ang0  atan2(a * tan(start), b);
        if(fabs(ang0 - 3.14159) < 0.001)
        {

```

---

```

    ang0  -3.14159;
}
    ang1  atan2(a * tan(end), b);
    if(fabs(3.14159 + ang1) < 0.001)
    {
        ang1  3.14159;
    }
    ang0  start;
    ang1  end;
    double ang  ang0;
    double rotate  atan2( y_end_major , x_end_major ) ;
    double rotate  0;
    CDC* pDC  GetDC();
    COLORREF Color;
    CPen *pPen;
    Color  RGB(0,0,255);
    CPen pen(PS_SOLID, 1, Color);
    pPen  pDC->SelectObject(&pen);
    double ptx,pty;
    pDC->MoveTo(x_c, y_c);
    for(int i  0; ang < ang1; i++)
    {
        ang  ang0 + i*mil_PI;
        ptx  a* cos(ang) * cos(rotate) - b*sin(ang)*sin(rotate);
        pty  a* cos(ang) * sin(rotate) + b*sin(ang)*cos(rotate);
        pDC->LineTo((ptx + x_c), (pty + y_c));
    }
}

void CWiseCutView::drawCCWArc(CDC *pDC,
                               Point3D pt, double radius,double ang0, double ang1)
{
    double ang  ang0;
    if(fabs(ang1-ang0) < 0.001)situation 1
    {
        ang1  ang0 + 2*PI;
        for(int i  0; ang < ang1; i++)
        {
            ang  (ang0 + i * mil_PI);
            Sleep(animatetime);
            pDC->LineTo(pt.x +radius * cos(ang),pt.y + radius * sin(ang));
        }
        pDC->LineTo(pt.x +radius * cos(ang1),pt.y + radius * sin(ang1));
    }
    else if(ang1 < 0 && ang0 > 0)  situation 2
    {
        for(int i  0; ang < PI; i++)
        {
            ang  (ang0 + i * mil_PI);
            Sleep(animatetime);
            pDC->LineTo(pt.x + radius * cos(ang), pt.y + radius * sin(ang));
        }
    }
}

```



---

```

    }
    ang  ang0  -1 * PI;
    for(i  0; ang < ang1; i++)
    {
        ang  (ang0 + i * mil_PI);
                                Sleep(animatetime);
        pDC->LineTo(pt.x + radius * cos(ang), pt.y + radius * sin(ang));
    }
    pDC->LineTo(pt.x +radius * cos(ang1),pt.y + radius * sin(ang1));
}
else          situation 3
{
    for(int i  0; ang < ang1; i++)
    {
        ang  (ang0 + i * mil_PI);
                                Sleep(animatetime);
        pDC->LineTo(pt.x +radius * cos(ang),pt.y + radius * sin(ang));
    }
    pDC->LineTo(pt.x +radius * cos(ang1),pt.y + radius * sin(ang1));
}
}

void CWiseCutView::drawCWArc(CDC *pDC,
                                Point3D pt, double radius,double ang0, double ang1)
{
    double ang  ang0;
    if(fabs(ang1-ang0) < 0.001)
    {
        ang1 ang0 - 2*PI;
        for(int i  0; ang > ang1; i++)
        {
            ang  (ang0 - i * mil_PI);
                                Sleep(animatetime);
            pDC->LineTo(pt.x + radius * cos(ang), pt.y + radius * sin(ang));
        }
    }
    else if( ang1 > 0 && ang0 < 0)
    {
        for(int i  0; ang > -1*PI; i++)
        {
            ang  (ang0 - i * mil_PI);
                                Sleep(animatetime);
            pDC->LineTo(pt.x + radius * cos(ang),pt.y + radius * sin(ang));
        }
        ang  ang0  PI;
        for( i  0; ang > ang1; i++)
        {
            ang  (ang0 - i * mil_PI);
                                Sleep(animatetime);
            pDC->LineTo(pt.x + radius * cos(ang),pt.y + radius * sin(ang));
        }
    }
}

```

```

    }
    else
    {
        for(int i = 0; ang > ang1; i++)
        {
            ang = (ang0 - i * mil_PI);

            Sleep(animatetime);
            pDC->LineTo(pt.x + radius * cos(ang), pt.y + radius * sin(ang));
        }
    }
}

void CWiseCutView::DrawGraph()
{
    CDC *pDC = GetDC();
    CPen *pPen;
    COLORREF Color;
    pDC->MoveTo(iXoffset, iYoffset);
    for(int i = 0; i < iNodeLength; i++)
    {
        if(fPosition[i].nG == 0)
        {
            Color = RGB(255,0,0);
            CPen pen(PS_DOT, 1, Color); PS_DASH
            pPen = pDC->SelectObject(&pen);
            pDC->LineTo(fPosition[i].x, fPosition[i].y);
        }
        else if(fPosition[i].nG == 2)
        {
            Color = RGB(255,0,0);
            CPen pen(PS_SOLID, 1, Color);
            pPen = pDC->SelectObject(&pen);
            drawCWArc(pDC, fPosition[i].ptCent, fPosition[i].r,
                    fPosition[i].radian0, fPosition[i].radian1);
            pDC->LineTo(fPosition[i].x, fPosition[i].y);
        }
        else if(fPosition[i].nG == 3)
        {
            Color = RGB(255,0,0);
            CPen pen(PS_SOLID, 1, Color);
            pPen = pDC->SelectObject(&pen);
            drawCCWArc(pDC, fPosition[i].ptCent, fPosition[i].r,
                    fPosition[i].radian0, fPosition[i].radian1);
            pDC->LineTo(fPosition[i].x, fPosition[i].y);
        }
    }
}

#include "stdafx.h"
#include "WiseCut.h"
#include "CtrlPanel.h"

```

---

```

#include "math.h"
#include "Dfjzh6030Api.h"
#include "dfjzh6050dll.h"
#ifdef _DEBUG
define new DEBUG_NEW
undef THIS_FILE
static char THIS_FILE[] __FILE__;
endif
int gLen;
long g_position 0L;
int g_iCurLineNumb 0;
CEvent gPauseEvent;
CEvent gResumeEvent;
CEvent gStartEventKill;
extern double xpos,ypos,zpos,apos;
IMPLEMENT_DYNCREATE(CCtrlPanel, CFormView)
CCtrlPanel::CCtrlPanel()
    : CFormView(CCtrlPanel::IDD)
{
    {{AFX_DATA_INIT(CCtrlPanel)
        NOTE: the ClassWizard will add member initialization here
    }}AFX_DATA_INIT
    m_dZpos 0.0;
    m_bRun FALSE;
}
CCtrlPanel::~CCtrlPanel()
{
}
void CCtrlPanel::DoDataExchange(CDataExchange* pDX)
{
    CFormView::DoDataExchange(pDX);
    {{AFX_DATA_MAP(CCtrlPanel)
        NOTE: the ClassWizard will add DDX and DDV calls here
    }}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CCtrlPanel, CFormView)
    {{AFX_MSG_MAP(CCtrlPanel)
        ON_WM_TIMER()
        ON_BN_CLICKED(IDC_BUTTON_LMSTART, OnButtonLmstart)
        ON_BN_CLICKED(IDC_BUTTON_LMPAUSE, OnButtonLmpause)
    }}AFX_MSG_MAP
END_MESSAGE_MAP()
#ifdef _DEBUG
void CCtrlPanel::AssertValid() const
{
    CFormView::AssertValid();
}
void CCtrlPanel::Dump(CDumpContext& dc) const
{
    CFormView::Dump(dc);
}

```

```

}
endif _DEBUG
void CCtrlPanel::OnInitialUpdate()
{
    CFormView::OnInitialUpdate();
    TODO: Add your specialized code here andor call the base class
    m_pFr (CMainFrame*)AfxGetApp()->m_pMainWnd;
    CEdit *pEdit (CEdit*)GetDlgItem(IDC_EDIT_FEEDRATE);
    SetDlgItemText(IDC_EDIT_FEEDRATE, "100");
    SetDlgItemText(IDC_EDIT_SPEED, "1000");
    CButton *pBt;
    pBt (CButton*)GetDlgItem(IDC_BUTTON_LMSTART);
    if(!m_bRun)
    {
        pBt (CButton*)GetDlgItem(IDC_BUTTON_LMPAUSE);
        pBt->EnableWindow(FALSE);
        pBt (CButton*)GetDlgItem(IDC_BUTTON_LMSTOP);
        pBt->EnableWindow(FALSE);
    }
}
BOOL CCtrlPanel::PreTranslateMessage(MSG* pMsg)
{
    TODO: Add your specialized code here andor call the base class
    CWnd *wnd WindowFromPoint(pMsg->pt) ;
    int iID wnd->GetDlgCtrlID();
    CString str;
    if(pMsg->messageWM_LBUTTONDOWN)
    {
        if(iID IDC_BUTTON_ZFW) Z+
        {
            SetTimer(1,10,NULL);
            nAxisCheckedFlag 2;
        }
        else if(iID IDC_BUTTON_ZBW) Z-
        {
            SetTimer(2,10,NULL);
            nAxisCheckedFlag 2;
        }
    }
    else if(pMsg->message WM_LBUTTONUP)
    {
        KillTimer(1);
        KillTimer(2);
    }
    return CFormView::PreTranslateMessage(pMsg);
}
void CCtrlPanel::OnTimer(UINT nIDEvent)
{
    TODO: Add your message handler code here andor call default
    switch(nIDEvent)

```

```

{
case 1:
{
    m_dZpos++;
    zpos  m_dZpos100;
    m_pFr->m_pWMView->Invalidate();
    m_pFr->m_pZAxisView->Invalidate();
}
break;
case 2:
{
    m_dZpos--;
    zpos  m_dZpos100;
    m_pFr->m_pWMView->Invalidate();
    m_pFr->m_pZAxisView->Invalidate();
}
break;
default:
break;
}
CFormView::OnTimer(nIDEvent);
}
void CCtrlPanel::OnButtonLmstart()
{
    TODO: Add your control notification handler code here
    CStdioFile file;
    m_strFileName  m_pFr->m_FilePath;
    if( !file.Open(*"auto.cnc"*m_strFileName,CFile::modeRead) ) *linmu.cnc*
    {
        AfxMessageBox("File open Failure!");
        return;
    }
    if(!m_bRun)
    {
        m_bRun  TRUE;
        CButton *pBt;
        pBt  (CButton*)GetDlgItem(IDC_BUTTON_LMSTART);
        pBt->EnableWindow(FALSE);
        pBt  (CButton*)GetDlgItem(IDC_BUTTON_LMPAUSE);
        pBt->EnableWindow(TRUE);
        pBt  (CButton*)GetDlgItem(IDC_BUTTON_LMSTOP);
        pBt->EnableWindow(TRUE);
        AfxBeginThread(LMStart, this);
        AfxBeginThread(m_pFr->m_pWiseCutView->DrawLine, this);
    }
}
UINT CCtrlPanel::LMStart(LPVOID lp)
{
    CCtrlPanel* pCtrlPanel;
    pCtrlPanel  (CCtrlPanel*)lp;

```

---

```

CString strFileName;
strFileName  pCtrlPanel->m_strFileName;
CDC* pDC  pCtrlPanel->GetDC();
CStdioFile file;
DWORD dwPosition  0;
CString strTest;
CString strTemp;
BOOL bNoEnd  TRUE;
BOOL bReRead  FALSE;
char chCmd;
int nLength;
int nLineNumb 0;
int nval_M  0;
int nval_G  0;
float fval_x 0.0,\
      fval_y 0.0,\
      fval_z 0.0,\
      fval_i 0.0,\
      fval_j 0.0,\
      fval_k 0.0,\
      fval_r 0.0,\
      fval_F 0.0;
float fSpeed  1000.0;
Point3D StartPoint,EndPoint;
Point3D ptCent;
ptCent.x  0;
ptCent.y  0;
ptCent.z  0;
CRect rect;
    Open Data File
if( !file.Open(strFileName,CFile::modeRead) ) *linmu.cnc*
{
    AfxMessageBox("File open Failure!");
}
else
pFile  fopen("corel3.cnc", "r");
pFile  fopen("auto.cnc", "r");
pFile  fopen(".cnc", "r");
pFile  fopen(".cnc", "r");
{
    while( bNoEnd ) It returns 0 if the current position is not end of file
    {
        LM_ResumeIpol6030();
        gLen  LM_GetBufferLen6030();
        if(gLen > 30) if(gLen != 0)
        {
            Reads a single line of text
            file.Seek(g_position, CFile::begin);
            bNoEnd  file.ReadString( strTest );
            g_position  file.GetPosition();

```

---

```

do
{
    bNoEnd  file.ReadString( strTest );
    pCtrlPanel->m_pFr->m_pGCodeView->ShowCurLine(g_iCurLineNumb);
    g_position  file.GetPosition();
    if(0  strTest.Compare("") || strTest.Find('%') > 0 || strTest.Find('(') > 0)
    {
        bReRead  TRUE;
        break;
    }
    else
    {
        bReRead  FALSE;
        break;
    }
    g_iCurLineNumb++;
}
while(bReRead && bNoEnd);
Parse data
nLength  strTest.GetLength();
for (int i  0; i < nLength; i++)
{
    chCmd  strTest.GetAt(i);0_based
    switch(chCmd)
    {
    case 'N':
        {
            strTemp  strTest.Right(nLength -i-1);
            nLineNumb  atoi(strTemp);
        }
        break;
    case 'G':
        {
            strTemp  strTest.Right(nLength -i-1);
            nval_G  atoi(strTemp);
        }
        break;
    case 'X':
        {
            strTemp  strTest.Right(nLength -i-1);
            fval_x  atof(strTemp);
        }
        break;
    case 'Y':
        {
            strTemp  strTest.Right(nLength -i-1);
            fval_y  atof(strTemp);
        }
        break;
    case 'Z':

```

---

```

        {
            strTemp  strTest.Right(nLength -i-1);
            fval_z  atof(strTemp);
        }
        break;
    case 'I':
        {
            strTemp  strTest.Right(nLength -i-1);
            fval_i  atof(strTemp);
        }
        break;
    case 'J':
        {
            strTemp  strTest.Right(nLength -i-1);
            fval_j  atof(strTemp);
        }
        break;
    case 'K':
        {
            strTemp  strTest.Right(nLength -i-1);
            fval_k  atof(strTemp);
        }
        break;
    case 'R':
        {
            strTemp  strTest.Right(nLength -i-1);
            fval_r  atof(strTemp);
        }
        break;
    case 'M':
        {
            strTemp  strTest.Right(nLength -i-1);
            nval_M  atoi(strTemp);
        }
        break;
    case 'F':
        {
            strTemp  strTest.Right(nLength -i-1);
            fval_F  atof(strTemp);
        }
        break;
    default:
        break;
    }end_of_switch
}end_of_for,finish parsing a single line cmd
if(nval_G  0 && nval_M  0)
{
    LM_SetLineIpPos6030(fval_x, fval_y, fval_z,fSpeed, nLineNumb);
    tempLineNum0[k]  nLineNumb;
    k++;
}

```



---

```

        StartPoint.x  fval_x;
        StartPoint.y  fval_y;
        StartPoint.z  fval_z;
    }
    if(nval_G  1 && nval_M  0)
    {
        LM_SetLineIpolPos6030(fval_x,      fval_y,      fval_z,fSpeed,
nLineNumb);

        tempLineNum1[k]  nLineNumb;
        k++;
        StartPoint.x  fval_x;
        StartPoint.y  fval_y;
        StartPoint.z  fval_z;
    }
    if(nval_G  2 && nval_M  0)
    {
        if( fval_r>0.001)
        {
            pCtrlPanel->get_cent_coordinate(nval_G,fval_r,
                StartPoint.x,StartPoint.y,
                fval_x,fval_y,
                &ptCent);
            LM_SetArcIpolPos6030(fval_x, fval_y, fval_z,
                ptCent.x,
                ptCent.y ,
                ptCent.z ,
                17,
                nval_G,
                fSpeed,
                nLineNumb);
            tempLineNum1[k]  nLineNumb;
            k++;
            StartPoint.x  fval_x;
            StartPoint.y  fval_y;
            StartPoint.z  fval_z;
            nval_G  0;G02
        }
        else if(fval_r < -0.001)
        {
            pCtrlPanel->get_cent_coordinate(nval_G,fval_r,
                StartPoint.x,StartPoint.y,
                fval_x,fval_y,
                &ptCent);
            LM_SetArcIpolPos6030(fval_x, fval_y, fval_z,
                ptCent.x,
                ptCent.y ,
                ptCent.z ,
                17,
                nval_G,
                fSpeed,

```

---

```

        nLineNumb);
    tempLineNum1[k] = nLineNumb;
    k++;
    StartPoint.x = fval_x;
    StartPoint.y = fval_y;
    StartPoint.z = fval_z;
    nval_G = 0;G02
}
else
{
    LM_SetArcIpolPos6030(fval_x, fval_y, fval_z,
        StartPoint.x + fval_i,
        StartPoint.y + fval_j,
        StartPoint.z + fval_k,
        17,
        nval_G,
        fSpeed,
        nLineNumb);
    tempLineNum1[k] = nLineNumb;
    k++;
    StartPoint.x = fval_x;
    StartPoint.y = fval_y;
    StartPoint.z = fval_z;
    nval_G = 0;G02
}
}
if(nval_G == 3 && nval_M == 0)
{
    if( fval_r > 0.001)
    {
        pCtrlPanel->get_cent_coordinate(nval_G,fval_r,
            StartPoint.x,StartPoint.y,
            fval_x,fval_y,
            &ptCent);
        LM_SetArcIpolPos6030(fval_x, fval_y, fval_z,
            ptCent.x,
            ptCent.y,
            ptCent.z,
            17,
            nval_G,
            fSpeed,
            nLineNumb);
        tempLineNum1[k] = nLineNumb;
        k++;
        StartPoint.x = fval_x;
        StartPoint.y = fval_y;
        StartPoint.z = fval_z;
    }
    else if(fval_r < -0.001)
    {

```

---

```

        pCtrlPanel->get_cent_coordinate(nval_G,fval_r,
            StartPoint.x,StartPoint.y,
            fval_x,fval_y,
            &ptCent);
        LM_SetArcIpolPos6030(fval_x, fval_y, fval_z,
            ptCent.x,
            ptCent.y ,
            ptCent.z ,
            17,
            nval_G,
            fSpeed,
            nLineNumb);
        tempLineNum1[k]  nLineNumb;
        k++;
        StartPoint.x  fval_x;
        StartPoint.y  fval_y;
        StartPoint.z  fval_z;
    }
    else
    {
        LM_SetArcIpolPos6030(fval_x, fval_y, fval_z,
            StartPoint.x + fval_i,
            StartPoint.y + fval_j,
            StartPoint.z + fval_k,
            17,
            nval_G,
            fSpeed,
            nLineNumb);
        tempLineNum1[k]  nLineNumb;
        k++;
        StartPoint.x  fval_x;
        StartPoint.y  fval_y;
        StartPoint.z  fval_z;
    }
}
if(nval_M  30)
{
    LM_End6030(0);
}
LM_Start6030();
}
if( ::WaitForSingleObject(gPauseEvent, 0)  WAIT_OBJECT_0 )
{
    LM_PauseIpol6030();
    break;
}
if( ::WaitForSingleObject(gResumeEvent, 0)  WAIT_OBJECT_0 )
{
    LM_ResumeIpol6030();
}
}

```

```

    }
    file.Close();
    pCtrlPanel->m_bRun FALSE;
    gPauseEvent.SetEvent();
    if(!bNoEnd)
    {
        g_position  0L;
        g_iCurLineNumb  0;
    }
}
return 0;
}
UINT CCtrlPanel::LMStart_6050(LPVOID lp)
{
    CCtrlPanel* pCtrlPanel;
    pCtrlPanel  (CCtrlPanel*)lp;
    CString strFileName;
    strFileName  pCtrlPanel->m_pFr->m_FilePath;
    CDC* pDC  pCtrlPanel->GetDC();
    CStdioFile file;
    DWORD dwPosition  0;
    CString strTest;
    CString strTemp;
    BOOL bNoEnd  TRUE;
    BOOL bReRead  FALSE;
    char chCmd;
    int nLength;
    int nLineNumb 0;
    int nval_M  0;
    int nval_G  0;
    float fval_x 0.0,\
        fval_y 0.0,\
        fval_z 0.0,\
        fval_w 0.0,\
        fval_i 0.0,\
        fval_j 0.0,\
        fval_k 0.0,\
        fval_r 0.0,\
        fval_F 0.0;
    float fSpeed  1000.0;
    Point3D StartPoint,EndPoint;
    Point3D ptCent;
    ptCent.x  0;
    ptCent.y  0;
    ptCent.z  0;
    CRect rect;
    Open Data File
    if( !file.Open(*"auto.cnc"*strFileName,CFile::modeRead) ) *linmu.cnc*
    {
        AfxMessageBox("File open Failure!");
    }

```

```

    }
else
{
    file.Seek(dwPosition, CFile::current);
    while( bNoEnd )It returns 0 if the current position is not end of file
    {
        LM_ResumeIpol6030();
        LM_Resume_6050(0);
        gLen  LM_GetBufferLen6030();
        gLen  LM_GetBuffLen_6050(0);
        if(gLen > 12)if(gLen != 0)
        {
            Reads a single line of text
            file.Seek(g_position, CFile::begin);
            bNoEnd  file.ReadString( strTest );
            g_position  file.GetPosition();
            do
            {
                bNoEnd  file.ReadString( strTest );
pCtrlPanel->m_pFr->m_pGCodeView->ShowCurLine(g_iCurLineNumb);
                g_position  file.GetPosition();
                if(0  strTest.Compare("") || strTest.Find('%') > 0 ||strTest.Find('(') >
0)

                {
                    bReRead  TRUE;
                    break;
                }
            else
            {
                bReRead  FALSE;
                break;
            }
            g_iCurLineNumb++;
        }
        while(bReRead && bNoEnd);
        Parse data
        nLength  strTest.GetLength();
        for (int i  0; i < nLength; i++)
        {
            chCmd  strTest.GetAt(i);0_based
            switch(chCmd)
            {
                case 'N':
                {
                    strTemp  strTest.Right(nLength -i-1);
                    nLineNumb  atoi(strTemp);
                }
                break;
                case 'G':
                {

```

---

```
        strTemp  strTest.Right(nLength -i-1);
        nval_G   atoi(strTemp);
    }
    break;
case 'X':
    {
        strTemp  strTest.Right(nLength -i-1);
        fval_x   atof(strTemp);
    }
    break;
case 'Y':
    {
        strTemp  strTest.Right(nLength -i-1);
        fval_y   atof(strTemp);
    }
    break;
case 'Z':
    {
        strTemp  strTest.Right(nLength -i-1);
        fval_z   atof(strTemp);
    }
    break;
case 'I':
    {
        strTemp  strTest.Right(nLength -i-1);
        fval_i   atof(strTemp);
    }
    break;
case 'J':
    {
        strTemp  strTest.Right(nLength -i-1);
        fval_j   atof(strTemp);
    }
    break;
case 'K':
    {
        strTemp  strTest.Right(nLength -i-1);
        fval_k   atof(strTemp);
    }
    break;
case 'R':
    {
        strTemp  strTest.Right(nLength -i-1);
        fval_r   atof(strTemp);
    }
    break;
case 'M':
    {
        strTemp  strTest.Right(nLength -i-1);
        nval_M   atoi(strTemp);
```

---

```

        }
        break;
    case 'F':
    {
        strTemp  strTest.Right(nLength -i-1);
        fval_F  atof(strTemp);
    }
    break;
    default:
        break;
    }end_of_switch
}end_of_for,finish parsing a single line cmd
if(nval_G  0 && nval_M  0)
{
    LM_SetLineIpPos6030(fval_x, fval_y, fval_z,fSpeed, nLineNumb);
    LM_Line_6050(0, fval_x, fval_y, fval_z, 0, fSpeed, nLineNumb);
    StartPoint.x  fval_x;
    StartPoint.y  fval_y;
    StartPoint.z  fval_z;
}
if(nval_G  1 && nval_M  0)
{
    *
    LM_SetLineIpPos6030(fval_x, fval_y, fval_z,fSpeed, nLineNumb);
    *
    LM_Line_6050(0, fval_x, fval_y, fval_z, 0, fSpeed, nLineNumb);
    StartPoint.x  fval_x;
    StartPoint.y  fval_y;
    StartPoint.z  fval_z;
}
if(nval_G  2 && nval_M  0)
{
    if( fval_r>0.001)
    {
        pCtrlPanel->get_cent_coordinate(nval_G,fval_r,
            StartPoint.x,StartPoint.y,
            fval_x,fval_y,
            &ptCent);
        *
        LM_SetArcIpPos6030(fval_x, fval_y, fval_z
            ptCent.x,
            ptCent.y ,
            ptCent.z ,
            17,
            nval_G,
            fSpeed,
            nLineNumb);*
        LM_ArcCW_6050(0, fval_x, fval_y, fval_z, fval_w,
            ptCent.x, ptCent.y,
            fSpeed,

```

---

```

        nLineNumb);
    tempLineNum1[k] = nLineNumb;
    k++;
    StartPoint.x = fval_x;
    StartPoint.y = fval_y;
    StartPoint.z = fval_z;
    nval_G = 0;G02
}
else if(fval_r < -0.001)
{
    pCtrlPanel->get_cent_coordinate(nval_G,fval_r,
        StartPoint.x,StartPoint.y,
        fval_x,fval_y,
        &ptCent);
    LM_ArcCW_6050(0, fval_x, fval_y, fval_z, fval_w,
        ptCent.x, ptCent.y,
        fSpeed,
        nLineNumb);
    StartPoint.x = fval_x;
    StartPoint.y = fval_y;
    StartPoint.z = fval_z;
    nval_G = 0;G02
}
else
{
    *
    LM_SetArcIpolPos6030(fval_x, fval_y, fval_z,
        StartPoint.x + fval_i,
        StartPoint.y + fval_j,
        StartPoint.z + fval_k,
        17,
        nval_G,
        fSpeed,
        nLineNumb);
    tempLineNum1[k] = nLineNumb;
    k++;
    StartPoint.x = fval_x;
    StartPoint.y = fval_y;
    StartPoint.z = fval_z;

    nval_G = 0;G02*
    AfxMessageBox("0.001");
}
}
if(nval_G == 3 && nval_M == 0)
{
    if( fval_r>0.001)
    {
        pCtrlPanel->get_cent_coordinate(nval_G,fval_r,
            StartPoint.x,StartPoint.y,
            fval_x,fval_y,

```



---

```

        &ptCent);
    LM_ArcCCW_6050(0, fval_x, fval_y, fval_z, fval_w,
        ptCent.x, ptCent.y,
        fSpeed,
        nLineNumb);
    StartPoint.x  fval_x;
    StartPoint.y  fval_y;
    StartPoint.z  fval_z;
}
else if(fval_r < -0.001)
{
    pCtrlPanel->get_cent_coordinate(nval_G,fval_r,
        StartPoint.x,StartPoint.y,
        fval_x,fval_y,
        &ptCent);
    LM_ArcCCW_6050(0, fval_x, fval_y, fval_z, fval_w,
        ptCent.x, ptCent.y,
        fSpeed,
        nLineNumb);
    StartPoint.x  fval_x;
    StartPoint.y  fval_y;
    StartPoint.z  fval_z;
}
else
{
    *
    LM_SetArcIpolPos6030(fval_x, fval_y, fval_z,
        StartPoint.x + fval_i,
        StartPoint.y + fval_j,
        StartPoint.z + fval_k,
        17,
        nval_G,
        fSpeed,
        nLineNumb);
    tempLineNum1[k]  nLineNumb;
    k++;
    StartPoint.x  fval_x;
    StartPoint.y  fval_y;
    StartPoint.z  fval_z;

    AfxMessageBox("0.001");
}
}
if(nval_M  30)
{
    LM_End_6050(0,0);
}
LM_Start_6050(0,1);
}
if( ::WaitForSingleObject(gPauseEvent, 0)  WAIT_OBJECT_0 )
{

```

---

```

        LM_Pause_6050(0);
        break;
    }
    if( ::WaitForSingleObject(gResumeEvent, 0)  WAIT_OBJECT_0 )
    {
        LM_Resume_6050(0);
    }
    *
    if( ::WaitForSingleObject(gStartEventKill, 0)  WAIT_OBJECT_0 )
    {
        LM_Resume_6050(0);
    }*
}
file.Close();
pCtrlPanel->m_bRun FALSE;
gPauseEvent.SetEvent();
if(!bNoEnd)
{
    g_position  0L;
    g_iCurLineNumb  0;
}
}
return 0;
}
void CCtrlPanel::get_cent_coordinate(int nFlagArc, double R,
                                     double x0, double y0, double x1, double y1,
                                     Point3D *cent)
{
    double dx,dy,angle;
    dx  x1 - x0;
    dy  y1 - y0;
    angle  atan2(dy, dx);
    double a  (x0 + x1)2;
    double b  (y0 + y1)2;
    double AB  sqrt(dx*dx + dy*dy);
    double OC  sqrt(R*R -AB*AB4.00);
    if(nFlagArc  2)
    {
        if(R > 0.001)
        {
            cent->x  a + OC*sin(angle);
            cent->y  b - OC*cos(angle);
        }
        if(R < -0.001)
        {
            cent->x  a - OC*sin(angle);
            cent->y  b + OC*cos(angle);
        }
    }
    if(nFlagArc  3)

```

```

    {
        if(R > 0.001)
        {
            cent->x  a - OC*sin(angle);
            cent->y  b + OC*cos(angle);
        }
        if(R < -0.001)
        {
            cent->x  a + OC*sin(angle);
            cent->y  b - OC*cos(angle);
        }
    }
}

void CCtrlPanel::OnButtonLmpause()
{
    TODO: Add your control notification handler code here
    gPauseEvent.SetEvent();
    m_bRun  FALSE;
    CButton *pBt;
    pBt  (CButton*)GetDlgItem(IDC_BUTTON_LMSTART);
    pBt->EnableWindow(!m_bRun);
    pBt  (CButton*)GetDlgItem(IDC_BUTTON_LMPAUSE);
    pBt->EnableWindow(m_bRun);
}

DJListBox.cpp : implementation file
#include "stdafx.h"
#include "WiseCut.h"
#include "DJListBox.h"
#ifdef _DEBUG
define new DEBUG_NEW
undef THIS_FILE
static char THIS_FILE[]  __FILE__;
#endif

CDJListBox
CDJListBox::CDJListBox()
{
    m_nMaxWidth  0;
}

CDJListBox::~CDJListBox()
{
}

BEGIN_MESSAGE_MAP(CDJListBox, CListBox)
    {{AFX_MSG_MAP(CDJListBox)
        NOTE - the ClassWizard will add and remove mapping macros here.
    }}AFX_MSG_MAP
END_MESSAGE_MAP()

int CDJListBox::AddString( LPCTSTR lpszItem )
{
    int nRet  CListBox::AddString(lpszItem);
    SCROLLINFO scrollInfo;

```

---

```

    memset(&scrollInfo, 0, sizeof(SCROLLINFO));
    scrollInfo.cbSize = sizeof(SCROLLINFO);
    scrollInfo.fMask = SIF_ALL;
    GetScrollInfo(SB_VERT, &scrollInfo, SIF_ALL);
    int nScrollWidth = 0;
    if(GetCount() > 1 && ((int)scrollInfo.nMax > (int)scrollInfo.nPage))
    {
        nScrollWidth = GetSystemMetrics(SM_CXVSCROLL);
    }
    SIZE sSize;
    CClientDC myDC(this);
    CFont* pListBoxFont = GetFont();
    if(pListBoxFont != NULL)
    {
        CFont* pOldFont = myDC.SelectObject(pListBoxFont);
        GetTextExtentPoint32(myDC.m_hDC,
            lpzItem, strlen(lpzItem), &sSize);
        m_nMaxWidth = max(m_nMaxWidth, (int)sSize.cx);
        SetHorizontalExtent(m_nMaxWidth + 3);
        myDC.SelectObject(pOldFont);
    }
    return nRet;
}

#include "stdafx.h"
#include "WiseCut.h"
#include "DlgSetParam.h"
#ifdef _DEBUG
define new DEBUG_NEW
undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

CDlgSetParam::CDlgSetParam(CWnd* pParent *NULL*)
: CDialog(CDlgSetParam::IDD, pParent)
{
    {{AFX_DATA_INIT(CDlgSetParam)
        NOTE: the ClassWizard will add member initialization here
    }}AFX_DATA_INIT
}

void CDlgSetParam::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    {{AFX_DATA_MAP(CDlgSetParam)
        NOTE: the ClassWizard will add DDX and DDV calls here
    }}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CDlgSetParam, CDialog)
    {{AFX_MSG_MAP(CDlgSetParam)
        NOTE: the ClassWizard will add message map macros here
    }}AFX_MSG_MAP
END_MESSAGE_MAP()

```

---

```

#include "stdafx.h"
#include "WiseCut.h"
#include "GCodeView.h"
#ifdef _DEBUG
define new DEBUG_NEW
undef THIS_FILE
static char THIS_FILE[] __FILE__;
endif
IMPLEMENT_DYNCREATE(CGCodeView, CFormView)
CGCodeView::CGCodeView()
    : CFormView(CGCodeView::IDD)
{
    {{AFX_DATA_INIT(CGCodeView)
        NOTE: the ClassWizard will add member initialization here
    }}AFX_DATA_INIT
}
CGCodeView::~CGCodeView()
{
}
void CGCodeView::DoDataExchange(CDataExchange* pDX)
{
    CFormView::DoDataExchange(pDX);
    {{AFX_DATA_MAP(CGCodeView)
        DDX_Control(pDX, IDC_LIST_GCODE, m_lListTest);
    }}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CGCodeView, CFormView)
    {{AFX_MSG_MAP(CGCodeView)
        NOTE - the ClassWizard will add and remove mapping macros here.
    }}AFX_MSG_MAP
END_MESSAGE_MAP()
#ifdef _DEBUG
void CGCodeView::AssertValid() const
{
    CFormView::AssertValid();
}
void CGCodeView::Dump(CDumpContext& dc) const
{
    CFormView::Dump(dc);
}
#endif
void CGCodeView::OnInitialUpdate()
{
    CFormView::OnInitialUpdate();
    TODO: Add your specialized code here and/or call the base class
    *
    *         void CMainFrame::GetWndPointer()
    *         this->m_pFr  (CMainFrame*)AfxGetApp()->m_pMainWnd;
    *
    m_pFr  (CMainFrame*)AfxGetApp()->m_pMainWnd;

```

```

}
UINT CGCodeView::ReadFile(LPVOID lp)
{
    CGCodeView *pGCodeView (CGCodeView*)lp;
    pGCodeView->DisplayGCode(pGCodeView->m_strFileName);
    return 0L;
}
void CGCodeView::DisplayGCode(CString strFileName)
{
    CStdioFile file;
    CString strGCode;
    CString str;
    BOOL bNoEnd TRUE;
    int nLine 1;
    if( file.Open(strFileName, CFile::modeRead) )
    {
        while(bNoEnd)
        {
            str.Format(_T("%.5d "),nLine);
            bNoEnd file.ReadString( strGCode );
            str + strGCode;
            m_IListTest.AddString(str);
            nLine++;
        }
    }
    file.Close();
    ::SendMessage(this->m_pFr->m_pWMView->m_hWnd,MESSAGE_GETDATAFINISH
ED,0,0);
}
void CStatusView::OnInitialUpdate()
{
    CFormView::OnInitialUpdate();
    TODO: Add your specialized code here and/or call the base class
    m_pFr (CMainFrame*)AfxGetApp()->m_pMainWnd;
    CStatic *pStatic (CStatic*)GetDlgItem(IDC_STATIC_X_COORD);
    SetDlgItemText(IDC_STATIC_X_COORD, "0.001");
    AfxBeginThread(ShowAllCoordinate, this);
}
void CStatusView::ShowXCoordinate()
{
    CString strabs,strrel;
    m_fRel_X_coordinate m_fAbs_X_coordinate xpos10;
    strabs.Format(_T("%.3lf"),m_fAbs_X_coordinate);
    strrel.Format(_T("%.3lf"),m_fRel_X_coordinate);
    SetDlgItemText(IDC_STATIC_X_COORD,strabs);
    SetDlgItemText(IDC_STATIC_X_RELCOORD,strrel);
}
void CStatusView::ShowYCoordinate()
{
    CString strabs,strrel;

```

```

        m_fRel_Y_coordinate  m_fAbs_Y_coordinate  ypos10;
        strabs.Format(_T("%8.3lf"),m_fAbs_Y_coordinate);
        strrel.Format(_T("%8.3lf"),m_fRel_Y_coordinate);
        SetDlgItemText(IDC_STATIC_Y_COORD,strabs);
        SetDlgItemText(IDC_STATIC_Y_RELCOORD,strrel);
    }
void CStatusView::ShowZCoordinate()
{
    CString strabs,strrel;
    m_fRel_Z_coordinate  m_fAbs_Z_coordinate  zpos10;
    strabs.Format(_T("%8.3lf"),m_fAbs_Z_coordinate);
    strrel.Format(_T("%8.3lf"),m_fRel_Z_coordinate);
    SetDlgItemText(IDC_STATIC_Z_COORD,strabs);
    SetDlgItemText(IDC_STATIC_Z_RELCOORD,strrel);
}
void CStatusView::ShowACoordinate()
{
    CString strabs,strrel;
    m_fRel_A_coordinate  m_fAbs_A_coordinate  apos10;
    strabs.Format(_T("A %8.3lf"),m_fAbs_A_coordinate);
    strrel.Format(_T("A %8.3lf"),m_fRel_A_coordinate);
    SetDlgItemText(IDC_STATIC_ABSCOORD3,strabs);
    SetDlgItemText(IDC_STATIC_RELCOORD3,strrel);
}
*
UINT CStatusView::ShowAllCoordinate(LPVOID lp)
{
    CStatusView *pStatusView  (CStatusView*)lp;
    pStatusView->ShowXCoordinate();
    pStatusView->ShowYCoordinate();
    pStatusView->ShowZCoordinate();
    return 0L;
}
void CGCodeView::ShowCurLine(int i)
{
    m_pListBox  (CListBox*)GetDlgItem(IDC_LIST_GCODE);
    m_lListTest.SetCurSel(i);
}
#include "stdafx.h"
#include "WiseCut.h"
#include "MainFrm.h"
#include "dlgSetparam.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[]  __FILE__;
#endif
IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    {{AFX_MSG_MAP(CMainFrame)
        ON_WM_CREATE()
    }}

```

---

```

        ON_COMMAND(ID_FILE_OPEN, OnFileOpen)
        ON_COMMAND(ID_MENUITEM_AXIS_PARAM, OnMenuItemAxisParam)
    } }AFX_MSG_MAP
END_MESSAGE_MAP()
static UINT indicators[]
{
    ID_SEPARATOR,           status line indicator
    ID_INDICATOR_CAPS,
    ID_INDICATOR_NUM,
    ID_INDICATOR_SCRL,
};
CMainFrame::CMainFrame()
{
    TODO: add member initialization code here
}
CMainFrame::~CMainFrame()
{
}
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
        return -1;
    if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD | WS_VISIBLE |
CBRs_TOP
        | CBRs_GRIPPER    | CBRs_TOOLTIPS    | CBRs_FLYBY    |
CBRs_SIZE_DYNAMIC) ||
        !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
    {
        TRACE0("Failed to create toolbar\n");
        return -1;          fail to create
    } *
    if (!m_wndStatusBar.Create(this) ||
        !m_wndStatusBar.SetIndicators(indicators,
            sizeof(indicators)/sizeof(UINT)))
    {
        TRACE0("Failed to create status bar\n");
        return -1;          fail to create
    }
    m_wndToolBar.EnableDocking(CBRs_ALIGN_ANY);
    EnableDocking(CBRs_ALIGN_ANY);
    DockControlBar(&m_wndToolBar);*
    this->GetWndPointer();
    return 0;
}
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CFrameWnd::PreCreateWindow(cs) )
        return FALSE;
    TODO: Modify the Window class or styles here by modifying
    the CREATESTRUCT cs

```



---

```

        return TRUE;
    }
#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
    CFrameWnd::AssertValid();
}
void CMainFrame::Dump(CDumpContext& dc) const
{
    CFrameWnd::Dump(dc);
}
#endif _DEBUG
BOOL CMainFrame::OnCreateClient(LPCREATESTRUCT lpcs, CCreateContext* pContext)
{
    TODO: Add your specialized code here and/or call the base class
    int cx  GetSystemMetrics(SM_CXSCREEN);
    int cy  GetSystemMetrics(SM_CYSCREEN);
    if(m_Sp1.CreateStatic(this,2,1)NULL)
        return FALSE;
    m_Sp1.SetRowInfo(0,cy*58,100);
    m_Sp3m_Sp1(0,0)(1,2)
    if(m_Sp3.CreateStatic(&m_Sp1,1,2,WS_CHILD|WS_VISIBLE,
m_Sp1.IdFromRowCol(0, 0))NULL)
        return FALSE;
    m_Sp3.SetRowInfo(0,224,200);
    m_Sp3.CreateView(0,0,RUNTIME_CLASS(CZAxisView),CSize(100,224),pContext);
    m_Sp3.CreateView(0,0,RUNTIME_CLASS(CWiseCutView),CSize(700,224),pContext);
    m_Sp3.CreateView(0,1,RUNTIME_CLASS(CStatusView),CSize(224,224),pContext);
    m_Sp2m_Sp1(1,0)(1,3)
    if(m_Sp2.CreateStatic(&m_Sp1,1,3,WS_CHILD|WS_VISIBLE,
m_Sp1.IdFromRowCol(1, 0))NULL)
        return FALSE;
    m_Sp2.CreateView(0,0,RUNTIME_CLASS(CZAxisView),CSize(250,224),pContext);
    m_Sp2.CreateView(0,1,RUNTIME_CLASS(CGCodeView),CSize(300,224),pContext);
    m_Sp2.CreateView(0,2,RUNTIME_CLASS(CCtrlPanel),CSize(450,224),pContext);
    return TRUE;
    return CFrameWnd::OnCreateClient(lpcs, pContext);
}
void CMainFrame::GetWndPointer()
{
    CWnd *pWnd  NULL;
    pWnd  this->m_Sp2.GetPane(0,0);
    this->m_pZAxisView  DYNAMIC_DOWNCAST(CZAxisView,pWnd);
    pWnd  this->m_Sp2.GetPane(0,1);
    this->m_pGCodeView  DYNAMIC_DOWNCAST(CGCodeView,pWnd);
    pWnd  this->m_Sp2.GetPane(0,2);
    this->m_pCtrlPanel  DYNAMIC_DOWNCAST(CCtrlPanel,pWnd);
    pWnd  this->m_Sp3.GetPane(0,0);
    this->m_pWiseCutView  DYNAMIC_DOWNCAST(CWiseCutView,pWnd);
    pWnd  this->m_Sp3.GetPane(0,1);

```

---

```

        this->m_pStatusView  DYNAMIC_DOWNCAST(CStatusView,pWnd);
    }
void CMainFrame::OnFileOpen()
{
    TODO: Add your command handler code here
    CFileDialog
    dlg(TRUE,_T("cnc"),_T("*.cnc"),OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT,_T
("NC(*.cnc)|*.cnc|"));
    if (IDCANCEL  dlg.DoModal())
        return;
    m_pWMView->Invalidate();
    m_FilePath  dlg.GetPathName();
    m_pGCodeView->m_strFileName  m_FilePath;
    m_pWiseCutView->m_strFileName  m_FilePath;
    AfxBeginThread(CGCodeView::ReadFile,m_pGCodeView);
    m_pWiseCutView->ParseCNCDData(m_pWiseCutView->m_strFileName);
    m_pWiseCutView->GetGCodeInfo();
    m_pWiseCutView->DataAmplify();
    m_pWiseCutView->DrawGraph();
    m_pWiseCutView->Invalidate();
    AfxGetMainWnd()->SetWindowText(m_FilePath);
}
void CMainFrame::OnMenuItemAxisParam()
{
    TODO: Add your command handler code here
    CDlgSetParam dlgAxisParam;
    if(dlgAxisParam.DoModal()  IDOK)
    {
    }
}
#include "stdafx.h"
#ifdef _DEBUG
define new DEBUG_NEW
undef THIS_FILE
static char THIS_FILE[]  __FILE__;
endif
MySp::MySp()
{
}
MySp::~MySp()
{
}
BEGIN_MESSAGE_MAP(MySp, CSplitterWnd)
    {{AFX_MSG_MAP(MySp)
        ON_WM_LBUTTONDOWN()
        ON_WM_MOUSEMOVE()
    }}AFX_MSG_MAP
END_MESSAGE_MAP()
void MySp::OnLButtonDown(UINT nFlags, CPoint point)
{

```

---

```

        TODO: Add your message handler code here and/or call default
        return;
        CSplitterWnd::OnLButtonDown(nFlags, point);
    }
void MySp::OnMouseMove(UINT nFlags, CPoint point)
{
    return;
    CSplitterWnd::OnMouseMove(nFlags, point);
}
#include "stdafx.h"
#include "WiseCut.h"
#include "StatusView.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] __FILE__;
#endif
#include "Mainfrm.h"
extern double xpos, ypos, zpos, apos;
extern int axis_X; 0;
extern int axis_Y; 1;
extern int axis_Z; 2;
extern double LenPerPulse; 0.0025;
extern double Accuracy; 0.03;
IMPLEMENT_DYNCREATE(CStatusView, CFormView)
CStatusView::CStatusView()
    : CFormView(CStatusView::IDD)
{
    {{AFX_DATA_INIT(CStatusView)
        NOTE: the ClassWizard will add member initialization here
    }}AFX_DATA_INIT
    m_fAbs_X_coordinate 0.0;
    m_fAbs_Y_coordinate 0.0;
    m_fAbs_Z_coordinate 0.0;
    m_fAbs_A_coordinate 0.0;
    m_fRel_X_coordinate 0.0;
    m_fRel_Y_coordinate 0.0;
    m_fRel_Z_coordinate 0.0;
    m_fRel_A_coordinate 0.0;
}
CStatusView::~CStatusView()
{
}
void CStatusView::DoDataExchange(CDataExchange* pDX)
{
    CFormView::DoDataExchange(pDX);
    {{AFX_DATA_MAP(CStatusView)
        NOTE: the ClassWizard will add DDX and DDV calls here
    }}AFX_DATA_MAP
}

```

---

```

BEGIN_MESSAGE_MAP(CStatusView, CFormView)
    {{AFX_MSG_MAP(CStatusView)
        NOTE - the ClassWizard will add and remove mapping macros here.
    }}AFX_MSG_MAP
END_MESSAGE_MAP()

#ifdef _DEBUG
void CStatusView::AssertValid() const
{
    CFormView::AssertValid();
}

void CStatusView::Dump(CDumpContext& dc) const
{
    CFormView::Dump(dc);
}

#endif _DEBUG

CStatusView message handlers
void CStatusView::OnInitialUpdate()
{
    CFormView::OnInitialUpdate();
    TODO: Add your specialized code here and/or call the base class
    m_pFr (CMainFrame*)AfxGetApp()->m_pMainWnd;
    CStatic *pStatic (CStatic*)GetDlgItem(IDC_STATIC_X_COORD);
    SetDlgItemText(IDC_STATIC_X_COORD, "0.001");
    AfxBeginThread(ShowAllCoordinate, this);
}

void CStatusView::ShowXCoordinate()
{
    CString strabs,strrel;
    m_fRel_X_coordinate m_fAbs_X_coordinate xpos10;
    strabs.Format(_T("%.3lf"),m_fAbs_X_coordinate);
    strrel.Format(_T("%.3lf"),m_fRel_X_coordinate);
    SetDlgItemText(IDC_STATIC_X_COORD,strabs);
    SetDlgItemText(IDC_STATIC_X_RELCOORD,strrel);
}

void CStatusView::ShowYCoordinate()
{
    CString strabs,strrel;
    m_fRel_Y_coordinate m_fAbs_Y_coordinate ypos10;
    strabs.Format(_T("%.3lf"),m_fAbs_Y_coordinate);
    strrel.Format(_T("%.3lf"),m_fRel_Y_coordinate);
    SetDlgItemText(IDC_STATIC_Y_COORD,strabs);
    SetDlgItemText(IDC_STATIC_Y_RELCOORD,strrel);
}

void CStatusView::ShowZCoordinate()
{
    CString strabs,strrel;
    m_fRel_Z_coordinate m_fAbs_Z_coordinate zpos10;
    strabs.Format(_T("%.3lf"),m_fAbs_Z_coordinate);
    strrel.Format(_T("%.3lf"),m_fRel_Z_coordinate);
    SetDlgItemText(IDC_STATIC_Z_COORD,strabs);

```

---

```

        SetDlgItemText(IDC_STATIC_Z_RELCOORD, strrel);
    }
void CStatusView::ShowACoordinate()
{
    CString strabs, strrel;
    m_fRel_A_coordinate  m_fAbs_A_coordinate  apos10;
    strabs.Format(_T("A %8.3lf"), m_fAbs_A_coordinate);
    strrel.Format(_T("A %8.3lf"), m_fRel_A_coordinate);
    SetDlgItemText(IDC_STATIC_ABSCOORD3, strabs);
    SetDlgItemText(IDC_STATIC_RELCOORD3, strrel);
}
/*
UINT CStatusView::ShowAllCoordinate(LPVOID lp)
{
    CStatusView *pStatusView  (CStatusView*)lp;
    pStatusView->ShowXCoordinate();
    pStatusView->ShowYCoordinate();
    pStatusView->ShowZCoordinate();
    return 0L;
}
#include "stdafx.h"
#include "WiseCut.h"
#include "MainFrm.h"
#include "WiseCutDoc.h"
#include "WiseCutView.h"
#include "Dfjzh6030Api.h"
#include "dfjzh6050dll.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[]  __FILE__;
#endif
BEGIN_MESSAGE_MAP(CWiseCutApp, CWinApp)
    {{AFX_MSG_MAP(CWiseCutApp)
        ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
    }}AFX_MSG_MAP
    ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
    ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
    ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()
CWiseCutApp::CWiseCutApp()
{
    TODO: add construction code here,
    Place all significant initialization in InitInstance
}
CWiseCutApp theApp;
double xpos  0.00, ypos  0.00, zpos  0.00, apos  0.00;
int axis_X  0;
int axis_Y  1;
int axis_Z  2;
int axis_W  3;

```

---

```

double LenPerPulse 0.0005;
double Accuracy 0.03;
BOOL CwiseCutApp::InitInstance()
{
    AfxEnableControlContainer();
    Standard initialization
    If you are not using these features and wish to reduce the size
    of your final executable, you should remove from the following
    the specific initialization routines you do not need.
#ifdef _AFXDLL
    Enable3dControls();           Call this when using MFC in a shared DLL
#else
    Enable3dControlsStatic();     Call this when linking to MFC statically
#endif
    SetRegistryKey(_T("Local AppWizard-Generated Applications"));
    LoadStdProfileSettings();     Load standard INI file options (including MRU)
    CSingleDocTemplate* pDocTemplate;
    pDocTemplate = new CSingleDocTemplate(
        IDR_MAINFRAME,
        RUNTIME_CLASS(CWiseCutDoc),
        RUNTIME_CLASS(CMainFrame),       main SDI frame window
        RUNTIME_CLASS(CWiseCutView));
    AddDocTemplate(pDocTemplate);
    CCommandLineInfo cmdInfo;
    ParseCommandLine(cmdInfo);
    if (!ProcessShellCommand(cmdInfo))
        return FALSE;
    m_pMainWnd->ShowWindow(SW_SHOWMAXIMIZED);
    m_pMainWnd->UpdateWindow();
    InitBoard_6030();
    return TRUE;
}

class CAboutDlg : public CDialog
{
public:
    CAboutDlg();
    {{AFX_DATA(CAboutDlg)
    enum { IDD = IDD_ABOUTBOX };
    }}AFX_DATA
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    DDXDDV support
    {{AFX_VIRTUAL
    Implementation
    protected:
    {{AFX_MSG(CAboutDlg)
        No message handlers
    }}AFX_MSG
    DECLARE_MESSAGE_MAP()
    };
    CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)

```

---

```

{
    {{AFX_DATA_INIT(CAboutDlg)
    }}AFX_DATA_INIT
}
void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    {{AFX_DATA_MAP(CAboutDlg)
    }}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
    {{AFX_MSG_MAP(CAboutDlg)
        No message handlers
    }}AFX_MSG_MAP
END_MESSAGE_MAP()
void CWISECutApp::OnAppAbout()
{
    CAboutDlg aboutDlg;
    aboutDlg.DoModal();
}
void CWISECutApp::InitBoard_6030()
{
    add additional init code
    xpos  0.0;
    ypos  0.0;
    zpos  0.0;
    apos  0.0;
    Exit6030Card(0);
    short init_ok0;
    short initIpol_ok  0;
    init_okInit6030Card(0,3,5,0);Interpolation interval was changed into 5ms from 10ms
    and it performs with higher accuracy.
    if(init_ok0)
    {
        initIpol_ok  LM_InitIpolEngine6030();
        if( initIpol_ok > 0 )
        {
            bIpol  TRUE;
            LM_SetIpol_X_Axis6030(axis_X, LenPerPulse, Accuracy);
            LM_SetIpol_Y_Axis6030(axis_Y, LenPerPulse, Accuracy);
            LM_SetIpol_Z_Axis6030(axis_Z, LenPerPulse, Accuracy);
            LM_SetIpolMinValue6030(0.005,6000);
            LM_SetIpolAccDec6030(100000, 100000);
            LM_SetIpolEnginePara1_6030(45, 60, 2);
            LM_SetIpolDecCoefficient6030(1.65);
            LM_CleanBuffer6030();
            Len  LM_GetBufferLen6030();
        }
        else
        {

```

---

```

        bIpol FALSE;
        AfxMessageBox("!");
    }
}
else if(init_ok! 0)
{
    AfxMessageBox("6030!");
}
}
void CwiseCutApp::InitBoard_6050()
{
    6050
    int statusInitCard_6050(0,1,1,1,1);
    if(status!0)
    {
        AfxMessageBox("6050");
    }
    Home_6050(0,0);
    Home_6050(0,1);
    Home_6050(0,2);
    Home_6050(0,3);
    SetAxisIO_6050(0,0,0,0,0);
    SetAxisIO_6050(0,1,0,0,0);
    SetAxisIO_6050(0,2,0,0,0);
    SetAxisIO_6050(0,3,0,0,0);
    LM_SetXAxis_6050(0,axis_X, LenPerPulse, Accuracy);
    LM_SetYAxis_6050(0,axis_Y, LenPerPulse, Accuracy);
    LM_SetZAxis_6050(0,axis_Z, LenPerPulse, Accuracy);
    LM_SetWAxis_6050(0,axis_W, LenPerPulse, Accuracy);
    LM_SetACCDec_6050(0,50000,50000);
}
#include "stdafx.h"
#include "WiseCut.h"
#include "WiseCutDoc.h"
#ifdef _DEBUG
define new DEBUG_NEW
undef THIS_FILE
static char THIS_FILE[] __FILE__;
endif
IMPLEMENT_DYNCREATE(CWiseCutDoc, CDocument)
BEGIN_MESSAGE_MAP(CWiseCutDoc, CDocument)
    } }AFX_MSG_MAP
END_MESSAGE_MAP()
CWiseCutDoc::CWiseCutDoc()
{
}
CWiseCutDoc::~CWiseCutDoc()
{
}
BOOL CWiseCutDoc::OnNewDocument()

```



---

```

{
    if (!CDocument::OnNewDocument())
        return FALSE;
    return TRUE;
}
void CwiseCutDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
    {
        TODO: add storing code here
    }
    else
    {
        TODO: add loading code here
    }
}
#ifdef _DEBUG
void CwiseCutDoc::AssertValid() const
{
    CDocument::AssertValid();
}
void CwiseCutDoc::Dump(CDumpContext& dc) const
{
    CDocument::Dump(dc);
}
#endif _DEBUG
#include "stdafx.h"
#include "WiseCut.h"
#include "WiseCutDoc.h"
#include "WiseCutView.h"
#include "MainFrm.h"
#include "Dfjzh6030Api.h"
#include "dfjzh6050dll.h"
#include "math.h"
#ifdef _DEBUG
define new DEBUG_NEW
undef THIS_FILE
static char THIS_FILE[] __FILE__;
#endif
const int X_OFF 0;
const int Y_OFF 0;
const int Z_OFF 0;
const double PI 3.1415926536;
const double mil_PI 0.0005; PI360 0.0087266, 2*PI(250*r10) 0.0005026
const int size 10000;
const float SIZERATIO 1;
GCmdLine fPosition[size];
UINT iNodeLength 0;
float fZoomSize 3.0;
int iXoffset 0;

```

---

```

int iYoffset 0;
int iZoffset 0;
extern double xpos,ypos,zpos,apos;
extern int axis_X; 0;
extern int axis_Y; 1;
extern int axis_Z; 2;
extern int axis_W;
extern double LenPerPulse; 0.0025;
extern double Accuracy; 0.03;
CEvent gDrawLineEventKill;
IMPLEMENT_DYNCREATE(CWiseCutView, CView)
BEGIN_MESSAGE_MAP(CWiseCutView, CView)
    {{AFX_MSG_MAP(CWiseCutView)
        ON_WM_PAINT()
    }}AFX_MSG_MAP
        Standard printing commands
        ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
        ON_COMMAND(ID_FILE_PRINT_DIRECT, CView::OnFilePrint)
        ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)
    END_MESSAGE_MAP()
CWiseCutView::CWiseCutView()
{
    TODO: add construction code here
}
CWiseCutView::~CWiseCutView()
{
}
BOOL CWiseCutView::PreCreateWindow(CREATESTRUCT& cs)
{
    TODO: Modify the Window class or styles here by modifying
        the CREATESTRUCT cs
    return CView::PreCreateWindow(cs);
}
void CWiseCutView::OnDraw(CDC* pDC)
{
    CWiseCutDoc* pDoc GetDocument();
    ASSERT_VALID(pDoc);
    CRect rect;
    GetClientRect(&rect);
    int nWidth rect.Width();
    int nHeight rect.Height();
    COLORREF Color;
    CPen *pPen;
    Color RGB(0,255,0);
    CPen pen(PS_SOLID, 1, Color);
    pPen pDC->SelectObject(&pen);
    pDC->MoveTo(20,20);
    pDC->LineTo(20,nHeight-20);
    pDC->LineTo(nWidth-20, nHeight-20);
    pDC->LineTo(nWidth-20, 20);
}

```

---

```

        pDC->LineTo(20,20);
        pen.DeleteObject();
        this->DrawSpindle(xpos,ypos);
        this->ParseCNCDData(m_strFileName);
    }
    BOOL CwiseCutView::OnPreparePrinting(CPrintInfo* pInfo)
    {
        default preparation
        return DoPreparePrinting(pInfo);
    }
    void CwiseCutView::OnBeginPrinting(CDC* *pDC*, CPrintInfo* *pInfo*)
    {
        TODO: add extra initialization before printing
    }
    void CwiseCutView::OnEndPrinting(CDC* *pDC*, CPrintInfo* *pInfo*)
    {
        TODO: add cleanup after printing
    }
#ifdef _DEBUG
    void CwiseCutView::AssertValid() const
    {
        CView::AssertValid();
    }
    void CwiseCutView::Dump(CDumpContext& dc) const
    {
        CView::Dump(dc);
    }
    CwiseCutDoc* CwiseCutView::GetDocument() non-debug version is inline
    {
        ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CwiseCutDoc)));
        return (CwiseCutDoc*)m_pDocument;
    }
#endif _DEBUG
    void CwiseCutView::DrawSpindle(double x, double y)
    {
        CDC *pDC  GetDC();
        pDC->Ellipse(x-1, y-1, x+1, y+1);
    }
    UINT CwiseCutView::DrawLine(LPVOID lp)
    {
        CwiseCutView* pWiseCutView;
        pWiseCutView  (CwiseCutView*)lp;
        float Xpos 0.0F, Ypos  0.0F;
        float oldxpos,oldypos,oldzpos;
        POINT ptTemp;
        CDC* pDC  pWiseCutView->GetDC();
        COLORREF Color;
        Color  RGB(0,0,255);
        CPen pen(PS_DASH, 3, Color );
        CPen* pPen  pDC->SelectObject(&pen);

```

---

```

oldxpos  actpos6030(axis_X) * LenPerPulse * 10;
oldypos  actpos6030(axis_Y) * LenPerPulse * 10;
oldzpos  actpos6030(axis_Z) * LenPerPulse * 10;
Xpos  actpos6030(axis_X) * LenPerPulse * fZoomSize  + iXoffset ;
Ypos  actpos6030(axis_Y) * LenPerPulse * fZoomSize  + iYoffset ;
ptTemp.x  Xpos;
ptTemp.y  Ypos;
pDC->MoveTo(ptTemp);
while(TRUE)
{
    Len  LM_GetBufferLen6030();
    Sleep(25);
    xpos  actpos6030(axis_X) * LenPerPulse * 10;
    ypos  actpos6030(axis_Y) * LenPerPulse * 10;
    zpos  actpos6030(axis_Z) * LenPerPulse * 10;
    if(fabs(oldxpos -xpos) > 0.001)
    {
        pWiseCutView->m_pFr->m_pStatusView->ShowXCoordinate();
    }
    if(fabs(oldypos - ypos) > 0.001)
    {
        pWiseCutView->m_pFr->m_pStatusView->ShowYCoordinate();
    }
    if(fabs(oldzpos - zpos) > 0.001)
    {
        pWiseCutView->m_pFr->m_pStatusView->ShowZCoordinate();
    }
    pWiseCutView->m_pFr->m_pZAxisView->Invalidate();
    oldxpos  xpos;
    oldypos  ypos;
    oldzpos  zpos;
    Xpos  actpos6030(axis_X) * LenPerPulse * fZoomSize  + iXoffset ;
    Ypos  actpos6030(axis_Y) * LenPerPulse * fZoomSize  + iYoffset ;
    ptTemp.x  Xpos;
    ptTemp.y  Ypos;
    pDC->LineTo(Xpos,Ypos);
    if( ::WaitForSingleObject(gDrawLineEventKill, 0)  WAIT_OBJECT_0 )
    {
        break;
    }
}
return 0;
}
UINT CWiseCutView::DrawLine_6050(LPVOID lp)
{
    CWiseCutView* pWiseCutView;
    pWiseCutView  (CWiseCutView*)lp;
    double Xpos 0.0F, Ypos  0.0F;
    POINT ptTemp;
    CDC* pDC  pWiseCutView->GetDC();

```

---

```

        COLORREF Color;
        Color RGB(0,0,255);
        CPen pen(PS_SOLID, 1, Color );
        CPen* pPen  pDC->SelectObject(&pen);
        Xpos  ReadAxisPos_6050(0, axis_X) * fZoomSize *LenPerPulse +X_OFF ;
        Ypos  ReadAxisPos_6050(0, axis_Y) * fZoomSize *LenPerPulse +Y_OFF ;
        ptTemp.x  (long)Xpos;
        ptTemp.y  (long)Ypos;
        pDC->MoveTo(ptTemp);
        while(TRUE)
        {
Xpos  ReadAxisPos_6050(0, axis_X) * fZoomSize *LenPerPulse +X_OFF ;
        Ypos  ReadAxisPos_6050(0, axis_Y) * fZoomSize *LenPerPulse +Y_OFF ;
                ptTemp.x  Xpos;
                ptTemp.y  Ypos;
                pDC->LineTo(Xpos,Ypos);
                if( ::WaitForSingleObject(gDrawLineEventKill, 0)  WAIT_OBJECT_0 )
                {
                        break;
                }
        }
        return 0;
}
void CWISECutView::draw_ellipse()
{
        double x_c  428.642207113819;
        double y_c  321.503012698986;
        double z_c  0.0;
        double x_end_major  30.1474702235905;
        double y_end_major  70.1062787003219;
        double z_end_major  0.0;
        double ratio  0.501665766070093;
        double start  2.06178120457755;
        double end  4.64634969578438;
        double ang0;
        double ang1;
        double a,b;
        if(fabs(y_end_major) > fabs(x_end_major))
        {
                b  sqrt(pow((z_end_major ),2) +
                        pow((y_end_major ),2) +
                        pow((x_end_major ),2) );
                a  fabs(b * ratio);
        }
        {
                a  sqrt(pow((z_end_major ),2) +
                        pow((y_end_major ),2) +
                        pow((x_end_major ),2) );
                b  fabs(a * ratio);
        }
}

```

---

```

    ang0  atan2(a * tan(start), b);
    if(fabs(ang0 - 3.14159) < 0.001)
    {
        ang0  -3.14159;
    }
    ang1  atan2(a * tan(end), b);
    if(fabs(3.14159 + ang1) < 0.001)
    {
        ang1  3.14159;
    }
    ang0  start;
    ang1  end;
    double ang  ang0;
    double rotate  atan2( y_end_major , x_end_major ) ;
    double rotate  0;
    CDC* pDC  GetDC();
    COLORREF Color;
    CPen *pPen;
    Color  RGB(0,0,255);
    CPen pen(PS_SOLID, 1, Color);
    pPen  pDC->SelectObject(&pen);
    double ptx,pty;
    pDC->MoveTo(x_c, y_c);
    for(int i  0; ang < ang1; i++)
    {
        ang  ang0 + i*mil_PI;
        ptx  a* cos(ang) * cos(rotate) - b*sin(ang)*sin(rotate);
        pty  a* cos(ang) * sin(rotate) + b*sin(ang)*cos(rotate);
        pDC->LineTo((ptx + x_c), (pty + y_c));
    }
}

void CWiseCutView::drawCCWArc(CDC *pDC,
                               Point3D pt, double radius,double ang0, double ang1)
{
    double ang  ang0;
    if(fabs(ang1-ang0) < 0.001)situation 1
    {
        ang1  ang0 + 2*PI;
        for(int i  0; ang < ang1; i++)
        {
            ang  (ang0 + i * mil_PI);
            Sleep(animatetime);
            pDC->LineTo(pt.x +radius * cos(ang),pt.y + radius * sin(ang));
        }
        pDC->LineTo(pt.x +radius * cos(ang1),pt.y + radius * sin(ang1));
    }
    else if(ang1 < 0 && ang0 > 0)  situation 2
    {
        for(int i  0; ang < PI; i++)
        {

```

---

```

        ang  (ang0 + i * mil_PI);
        Sleep(animatetime);
        pDC->LineTo(pt.x + radius * cos(ang), pt.y + radius * sin(ang));
    }
    ang  ang0  -1 * PI;
    for(i  0; ang < ang1; i++)
    {
        ang  (ang0 + i * mil_PI);
        Sleep(animatetime);
        pDC->LineTo(pt.x + radius * cos(ang), pt.y + radius * sin(ang));
    }
    pDC->LineTo(pt.x +radius * cos(ang1),pt.y + radius * sin(ang1));
}
else          situation 3
{
    for(int i  0; ang < ang1; i++)
    {
        ang  (ang0 + i * mil_PI);
        Sleep(animatetime);
        pDC->LineTo(pt.x +radius * cos(ang),pt.y + radius * sin(ang));
    }
    pDC->LineTo(pt.x +radius * cos(ang1),pt.y + radius * sin(ang1));
}
}
void CWiseCutView::drawCWArc(CDC *pDC,
                             Point3D pt, double radius,double ang0, double ang1)
{
    double ang  ang0;
    if(fabs(ang1-ang0) < 0.001)
    {
        ang1 ang0 - 2*PI;
        for(int i  0; ang > ang1; i++)
        {
            ang  (ang0 - i * mil_PI);
            Sleep(animatetime);
            pDC->LineTo(pt.x + radius * cos(ang), pt.y + radius * sin(ang));
        }
    }
    else if( ang1 > 0 && ang0 < 0)
    {
        for(int i  0; ang > -1*PI; i++)
        {
            ang  (ang0 - i * mil_PI);
            Sleep(animatetime);
            pDC->LineTo(pt.x + radius * cos(ang),pt.y + radius * sin(ang));
        }
        ang  ang0  PI;
        for( i  0; ang > ang1; i++)
        {
            ang  (ang0 - i * mil_PI);

```

---

```

        Sleep(animatetime);
        pDC->LineTo(pt.x + radius * cos(ang),pt.y + radius * sin(ang));
    }
}
else
{
    for(int i = 0; ang > ang1; i++)
    {
        ang = (ang0 - i * mil_PI);
        Sleep(animatetime);
        pDC->LineTo(pt.x + radius * cos(ang),pt.y + radius * sin(ang));
    }
}
}
void CWiseCutView::DrawGraph()
{
    CDC *pDC = GetDC();
    CPen *pPen;
    COLORREF Color;
    pDC->MoveTo(iXoffset,iYoffset);
    for(int i = 0; i < iNodeLength; i++)
    {
        if(fPosition[i].nG == 0 )
        {
            Color = RGB(255,0,0);
            CPen pen(PS_DOT, 1, Color);PS_DASH
            pPen = pDC->SelectObject(&pen);
            pDC->LineTo(fPosition[i].x,fPosition[i].y);
        }
        else if(fPosition[i].nG == 1)
        {
            Color = RGB(255,0,0);
            CPen pen(PS_SOLID, 1, Color);
            pPen = pDC->SelectObject(&pen);
            pDC->LineTo(fPosition[i].x,fPosition[i].y);
        }
        else if(fPosition[i].nG == 2)
        {
            Color = RGB(255,0,0);
            CPen pen(PS_SOLID, 1, Color);
            pPen = pDC->SelectObject(&pen);
            drawCArc(pDC, fPosition[i].ptCent, fPosition[i].r,
                    fPosition[i].radian0, fPosition[i].radian1);
            pDC->LineTo(fPosition[i].x,fPosition[i].y);
        }
        else if(fPosition[i].nG == 3)
        {
            Color = RGB(255,0,0);
            CPen pen(PS_SOLID, 1, Color);
            pPen = pDC->SelectObject(&pen);

```



---

```

        drawCCWArc(pDC, fPosition[i].ptCent, fPosition[i].r,
                    fPosition[i].radian0, fPosition[i].radian1);
        pDC->LineTo(fPosition[i].x,fPosition[i].y);
    }
}
}
void CWiseCutView::DataAmplify()
{
    for(int i = 0; i < iNodeLength; i++)
    {
        fPosition[i].ptCent.x = fPosition[i].ptCent.x * fZoomSize + iXoffset;
        fPosition[i].ptCent.y = fPosition[i].ptCent.y * fZoomSize + iYoffset;
        fPosition[i].ptCent.z = fPosition[i].ptCent.z * fZoomSize + iZoffset;
        fPosition[i].r = fPosition[i].r * fZoomSize;
        fPosition[i].x = fPosition[i].x * fZoomSize + iXoffset;
        fPosition[i].y = fPosition[i].y * fZoomSize + iYoffset;
        fPosition[i].z = fPosition[i].z * fZoomSize + iZoffset;
    }
}
void CWiseCutView::get_cent_coordinate(int nFlagArc,
                                       double R,
                                       double x0, double y0,
                                       double x1, double y1, Point3D *cent)
{
    double dx,dy,angle;
    dx = x1 - x0;
    dy = y1 - y0;
    angle = atan2(dy, dx);
    double a = (x0 + x1)/2;
    double b = (y0 + y1)/2;
    double AB = sqrt(dx*dx + dy*dy);
    double OC = sqrt(R*R - AB*AB/4.00);
    if(nFlagArc == 2)
    {
        if(R > 0.001)
        {
            cent->x = a + OC*sin(angle);
            cent->y = b - OC*cos(angle);
        }
        if(R < -0.001)
        {
            cent->x = a - OC*sin(angle);
            cent->y = b + OC*cos(angle);
        }
    }
    if(nFlagArc == 3)
    {
        if(R > 0.001)
        {
            cent->x = a - OC*sin(angle);

```

---

```

        cent->y  b + OC*cos(angle);
    }
    if(R < -0.001)
    {
        cent->x  a + OC*sin(angle);
        cent->y  b - OC*cos(angle);
    }
}

}

void CwiseCutView::GetData(LPARAM lp, WPARAM wp)
{
    this->ParseCNCData(m_strFileName);
}

void CwiseCutView::GetGCodeInfo()
{
    double dMaxX  fPosition[0].x;
    double dMinX  fPosition[0].x;
    double dMaxY  fPosition[0].y;
    double dMinY  fPosition[0].y;
    double dMaxZ  fPosition[0].z;
    double dMinZ  fPosition[0].z;
    for(int i  1; i < iNodeLength; i++)
    {
        if(fPosition[i].nG  2 || fPosition[i].nG  3)
        {
            if(dMaxX < fPosition[i].ptCent.x + fabs(fPosition[i].r))
            {
                dMaxX  fPosition[i].ptCent.x + fabs(fPosition[i].r);
            }
            if(dMinX > fPosition[i].ptCent.x - fabs(fPosition[i].r))
            {
                dMinX  fPosition[i].ptCent.x - fabs(fPosition[i].r);
            }
            if(dMaxY < fPosition[i].ptCent.y + fabs(fPosition[i].r))
            {
                dMaxY  fPosition[i].ptCent.y + fabs(fPosition[i].r);
            }
            if(dMinY > fPosition[i].ptCent.y - fabs(fPosition[i].r))
            {
                dMinY  fPosition[i].ptCent.y - fabs(fPosition[i].r);
            }
            if(dMaxZ < fPosition[i].z)
            {
                dMaxZ  fPosition[i].z;
            }
            if(dMinZ > fPosition[i].z)
            {
                dMinZ  fPosition[i].z;
            }
        }
    }
}

```

---

```

        else if(fPosition[i].nG == 0 || fPosition[i].nG == 1)
        {
            if(dMaxX < fPosition[i].x)
            {
                dMaxX = fPosition[i].x;
            }
            else if(dMinX > fPosition[i].x)
            {
                dMinX = fPosition[i].x;
            }
            if(dMaxY < fPosition[i].y)
            {
                dMaxY = fPosition[i].y;
            }
            else if(dMinY > fPosition[i].y)
            {
                dMinY = fPosition[i].y;
            }
            if(dMaxZ < fPosition[i].z)
            {
                dMaxZ = fPosition[i].z;
            }
            else if(dMinZ > fPosition[i].z)
            {
                dMinZ = fPosition[i].z;
            }
        }
    }
    CRect rect;
    GetClientRect(&rect);
    float f1 = (rect.Width()-100)(dMaxX - dMinX);
    float f2 = (rect.Height()-100)(dMaxY - dMinY);
    f1 > f2 ? fZoomSize = f2 : fZoomSize = f1;
    iXoffset = 50 - dMinX*fZoomSize;
    iYoffset = 50 - dMinY*fZoomSize;
}

void CwiseCutView::ParseCNCData(const char *pFileName)
{
    int index = 0;
    for(index; index < size; index++)
    {
        fPosition[index].x = X_OFF;
        fPosition[index].y = Y_OFF;
        fPosition[index].z = Z_OFF;
    }
    index = 0;
    CDC* pDC = GetDC();
    CRect rt;
    GetClientRect(&rt);
    COLORREF Color;

```

---

```

CPen *pPen;
Point3D ptEnd;
Point3D ptCenter;
CStdioFile file;
DWORD dwPosition  0;
CString strTest;
CString strTemp;
BOOL bNoEnd  TRUE;
BOOL bReRead  FALSE;
char chCmd;
int nLength;
int nLineNumb 0;
int nval_M  0;
int nval_G  0;
double fval_x 0.0,\
        fval_y 0.0,\
        fval_z 0.0,\
        fval_i 0.0,\
        fval_j 0.0,\
        fval_k 0.0,\
        fval_r 0.0,\
        fval_F 0.0,\
        fval_a 0.0,\
        fval_b 0.0;
double caculate_R  0;
if( !file.Open(pFileName,CFile::modeRead) )
{
    AfxMessageBox("File open Failure!");
}
else
{
    pDC->MoveTo(X_OFF,Y_OFF);
    while(bNoEnd)
    {
        Reads a single line of text
        file.Seek(dwPosition, CFile::begin);
        do
        {
            bNoEnd  file.ReadString( strTest );
            dwPosition  file.GetPosition();
            if(0  strTest.Compare("") )
            {
                bReRead  TRUE;
                break;
            }
            else if( strTest.Find('%') > 0)
            {
                bReRead  TRUE;
            }
            else if(strTest.Find('(') > 0)

```

---

```

        {
            bReRead  TRUE;
        }
    else
    {
        bReRead  FALSE;
        break;
    }
}
while(bReRead && bNoEnd);
bNoEnd  file.ReadString( strTest );
dwPosition  file.GetPosition();
Parse data
nLength  strTest.GetLength();
for (int i  0; i < nLength; i++)
{
    chCmd  strTest.GetAt(i);0_based
    switch(chCmd)
    {
    case 'N':
        {
            strTemp  strTest.Right(nLength -i-1);
            nLineNumb  atoi(strTemp);
        }
        break;
    case 'G':
        {
            strTemp  strTest.Right(nLength -i-1);
            nval_G  atoi(strTemp);
        }
        break;
    case 'X':
        {
            strTemp  strTest.Right(nLength -i-1);
            fval_x  atof(strTemp);
        }
        break;
    case 'Y':
        {
            strTemp  strTest.Right(nLength -i-1);
            fval_y  atof(strTemp);
        }
        break;
    case 'Z':
        {
            strTemp  strTest.Right(nLength -i-1);
            fval_z  atof(strTemp);
        }
        break;
    case 'T':

```

```
        {
            strTemp  strTest.Right(nLength -i-1);
            fval_i  atof(strTemp);
        }
        break;
case 'J':
    {
        strTemp  strTest.Right(nLength -i-1);
        fval_j  atof(strTemp);
    }
    break;
case 'K':
    {
        strTemp  strTest.Right(nLength -i-1);
        fval_k  atof(strTemp);
    }
    break;
case 'R':
    {
        strTemp  strTest.Right(nLength -i-1);
        fval_r  atof(strTemp);
    }
    break;
case 'M':
    {
        strTemp  strTest.Right(nLength -i-1);
        nval_M  atoi(strTemp);
    }
    break;
case 'F':
    {
        strTemp  strTest.Right(nLength -i-1);
        fval_F  atof(strTemp);
    }
    break;
case 'A':
    {
        strTemp  strTest.Right(nLength -i-1);
        fval_a  atof(strTemp);
    }
    break;
case 'B':
    {
        strTemp  strTest.Right(nLength -i-1);
        fval_b  atof(strTemp);
    }
    break;
default:
    break;
}end_of_switch
```

```

}end_of_for
if(nval_G == 0 && nval_M == 0)
{
    Color = RGB(0,255,0);
    CPen pen(PS_DOT, 1, Color);PS_DOT PS_DASH
    pPen = pDC->SelectObject(&pen);
    ptEnd.x = fval_x * SIZERATIO + X_OFF;
    ptEnd.y = fval_y * SIZERATIO + Y_OFF;
    pDC->LineTo(ptEnd.x, ptEnd.y);
    fPosition[index].nG = nval_G;
    fPosition[index].x = fval_x * SIZERATIO + X_OFF;
    fPosition[index].y = fval_y * SIZERATIO + Y_OFF;
    fPosition[index].z = 0;
    gChain.Insert(index, fPosition[index]);
    index++;
}
else if(nval_G == 1 && nval_M == 0)
{
    Color = RGB(255,0,0);
    CPen pen(PS_SOLID, 1, Color);
    pPen = pDC->SelectObject(&pen);
    ptEnd.x = fval_x * SIZERATIO + X_OFF;
    ptEnd.y = fval_y * SIZERATIO + Y_OFF;
    pDC->LineTo(ptEnd.x, ptEnd.y);
    fPosition[index].nG = nval_G;
    fPosition[index].x = fval_x * SIZERATIO + X_OFF;
    fPosition[index].y = fval_y * SIZERATIO + Y_OFF;
    fPosition[index].z = 0;
    gChain.Insert(index, fPosition[index]);
    index++;
}
else if(nval_G == 3 && nval_M == 0)
{
    Color = RGB(255,0,0);
    CPen pen(PS_SOLID, 1, Color);
    pPen = pDC->SelectObject(&pen);
    if( fval_r > 0.001 *|| fval_r < -0.001 *)R
    {
        get_cent_coordinate(nval_G, fval_r * SIZERATIO,
                           ptEnd.x, ptEnd.y,
                           fval_x*SIZERATIO + X_OFF,
                           fval_y*SIZERATIO + Y_OFF,
                           &ptCenter);
        caculate_R = fabs(fval_r * SIZERATIO);
        PI
        double ang0;
        ang0 = atan2(ptEnd.y - ptCenter.y, ptEnd.x - ptCenter.x);
        if(fabs(ang0 - 3.14159) < 0.001)
        {
            ang0 = -3.14159;

```

---

```

    }
    -PI
    double ang1;
    ang1 = atan2(fval_y*SIZERATIO + Y_OFF - ptCenter.y,
fval_x*SIZERATIO + X_OFF - ptCenter.x);
    if(fabs(3.14159 + ang1) < 0.001)
    {
        ang1 = 3.14159;
    }
    ang1 > ang0
    if(ang1 < ang0)
    {
        ang1 = ang1 + 3.14159 * 2;
    }
    drawCCWArc(pDC, ptCenter, caculate_R, ang0, ang1);
    ptEnd.x = fval_x * SIZERATIO + X_OFF;
    ptEnd.y = fval_y * SIZERATIO + Y_OFF;
    fPosition[index].nG = nval_G;
    fPosition[index].x = fval_x * SIZERATIO + X_OFF;
    fPosition[index].y = fval_y * SIZERATIO + Y_OFF;
    fPosition[index].z = 0;
    fPosition[index].ptCent = ptCenter;
    fPosition[index].r = caculate_R;
    fPosition[index].radian0 = ang0;
    fPosition[index].radian1 = ang1;
    gChain.Insert(index, fPosition[index]);
    index++;
}
else if(fval_r < -0.001)
{
    get_cent_coordinate(nval_G, fval_r * SIZERATIO,
        ptEnd.x, ptEnd.y,
        fval_x*SIZERATIO + X_OFF,
        fval_y*SIZERATIO + Y_OFF,
        &ptCenter);
    caculate_R = fabs(fval_r * SIZERATIO);
    double ang0;
    ang0 = atan2(ptEnd.y - ptCenter.y, ptEnd.x - ptCenter.x);
    if(fabs(ang0 - 3.14159) < 0.001)
    {
        ang0 = -3.14159;
    }
    double ang1;
    ang1 = atan2(fval_y*SIZERATIO + Y_OFF - ptCenter.y,
fval_x*SIZERATIO + X_OFF - ptCenter.x);
    -PI
    if(fabs(3.14159 + ang1) < 0.001)
    {
        ang1 = 3.14159;
    }
}

```



---

```

        ang1> ang0
        if(ang1 < ang0)
        {
            ang1 = ang1 + 3.14159 * 2;
        }
        drawCCWArc(pDC, ptCenter, caculate_R, ang0, ang1);
        ptEnd.x = fval_x * SIZERATIO + X_OFF;
        ptEnd.y = fval_y * SIZERATIO + Y_OFF;
        fPosition[index].nG = nval_G;
        fPosition[index].x = fval_x * SIZERATIO + X_OFF;
        fPosition[index].y = fval_y * SIZERATIO + Y_OFF;
        fPosition[index].z = 0;
        fPosition[index].ptCent = ptCenter;
        fPosition[index].r = caculate_R;
        fPosition[index].radian0 = ang0;
        fPosition[index].radian1 = ang1;
        index++;
    }
    else if(fval_r > -0.001 && fval_r < 0.001)
    {
    }
    else IJK
    {
        ptCenter.x = ptEnd.x + fval_i * SIZERATIO;
        ptCenter.y = ptEnd.y + fval_j * SIZERATIO;
        caculate_R = sqrt(pow(ptEnd.x - ptCenter.x, 2) + pow(ptEnd.y -
ptCenter.y, 2));

        PI
        double ang0;
        ang0 = atan2(ptEnd.y - ptCenter.y, ptEnd.x - ptCenter.x);
        if(fabs(ang0 - 3.14159) < 0.001)
        {
            ang0 = -3.14159;
        }
        -PI
        double ang1;
        ang1 = atan2(fval_y * SIZERATIO + Y_OFF - ptCenter.y,
fval_x * SIZERATIO + X_OFF - ptCenter.x);
        if(fabs(3.14159 + ang1) < 0.001)
        {
            ang1 = 3.14159;
        }
        ang1> ang0
        if(ang1 < ang0)
        {
            ang1 = ang1 + 3.14159 * 2;
        }
        drawCCWArc(pDC, ptCenter, caculate_R, ang0, ang1);
        ptEnd.x = fval_x * SIZERATIO + X_OFF;
        ptEnd.y = fval_y * SIZERATIO + Y_OFF;

```

---

```

        fPosition[index].nG    nval_G;
        fPosition[index].x    fval_x * SIZERATIO + X_OFF;
        fPosition[index].y    fval_y * SIZERATIO + Y_OFF;
        fPosition[index].z    0;
        fPosition[index].ptCent    ptCenter;
        fPosition[index].r    caculate_R;
        fPosition[index].radian0    ang0;
        fPosition[index].radian1    ang1;
        gChain.Insert(index, fPosition[index]);
        index++;
    }
    nval_G    3;
}
else if(nval_G    2 && nval_M    0)
{
    Color    RGB(255,0,0);
    CPen pen(PS_SOLID, 1, Color);
    pPen    pDC->SelectObject(&pen);
    if( fval_r > 0.001 || fval_r < -0.001)R
    {
        get_cent_coordinate(nval_G, fval_r * SIZERATIO,
                            ptEnd.x, ptEnd.y,
                            fval_x*SIZERATIO + X_OFF,
                            fval_y*SIZERATIO + Y_OFF,
                            &ptCenter);
        caculate_R    fabs(fval_r * SIZERATIO);
        -PI
        double ang0;
        ang0    atan2(ptEnd.y - ptCenter.y, ptEnd.x - ptCenter.x);
        if(fabs(ang0 + 3.14159) < 0.001)
        {
            ang0    3.14159;
        }
        PI
        double ang1;
        ang1    atan2(fval_y*SIZERATIO    +    Y_OFF    -    ptCenter.y,
fval_x*SIZERATIO + X_OFF - ptCenter.x);
        if(fabs(ang1 - 3.14159) < 0.001)
        {
            ang1    -3.14159;
        }
        ang1<ang0
        if(ang1 > ang0)
        {
            ang1    ang1 - 2 * 3.14159;
        }
        drawCArc(pDC, ptCenter, caculate_R, ang0, ang1);
        ptEnd.x    fval_x * SIZERATIO + X_OFF;
        ptEnd.y    fval_y * SIZERATIO + Y_OFF;
        fPosition[index].nG    nval_G;
    }
}

```

```

        fPosition[index].x  fval_x * SIZERATIO + X_OFF;
        fPosition[index].y  fval_y * SIZERATIO + Y_OFF;
        fPosition[index].z  0;
        fPosition[index].ptCent  ptCenter;
        fPosition[index].r  caculate_R;
        fPosition[index].radian0  ang0;
        fPosition[index].radian1  ang1;
        gChain.Insert(index, fPosition[index]);
        index++;
    }
    else if(fval_r > -0.001 && fval_r < 0.001)
    {
    }
    else
    {
        ptCenter.x  ptEnd.x + fval_i *SIZERATIO;
        ptCenter.y  ptEnd.y + fval_j *SIZERATIO;
        caculate_R  sqrt(pow(ptEnd.x -ptCenter.x, 2) + pow(ptEnd.y -
ptCenter.y, 2));

        -PI
        double ang0;
        ang0  atan2(ptEnd.y - ptCenter.y, ptEnd.x - ptCenter.x);
        if(fabs(ang0 + 3.14159) < 0.001)
        {
            ang0  3.14159;
        }
        PI
        double ang1;
        ang1  atan2(fval_y*SIZERATIO + Y_OFF - ptCenter.y,
fval_x*SIZERATIO + X_OFF - ptCenter.x);
        if(fabs(ang1 - 3.14159) < 0.001)
        {
            ang1  -3.14159;
        }
        ang1<ang0
        if(ang1 > ang0)
        {
            ang1  ang1 - 2 * 3.14159;
        }
        drawCWArc(pDC, ptCenter, caculate_R, ang0, ang1);
        ptEnd.x  fval_x * SIZERATIO + X_OFF;
        ptEnd.y  fval_y * SIZERATIO + Y_OFF;
        fPosition[index].nG  nval_G;
        fPosition[index].x  fval_x * SIZERATIO + X_OFF;
        fPosition[index].y  fval_y * SIZERATIO + Y_OFF;
        fPosition[index].ptCent  ptCenter;
        fPosition[index].r  caculate_R;
        fPosition[index].radian0  ang0;
        fPosition[index].radian1  ang1;
        gChain.Insert(index, fPosition[index]);
    }
}

```

```

        index++;
    }
    nval_G = 2;
}
}end_of_while
iNodeLength = index;
}
file.Close();
}
void CwiseCutView::OnPaint()
{
    CPaintDC dc(this); // device context for painting
    // TODO: Add your message handler code here
    CRect rect;
    GetClientRect(&rect);
    int nShade = 62;
    int nShade1 = 192;
    CPen darkerpen (PS_SOLID, 1, RGB (nShade, nShade, nShade));
    CPen* pOldPen = dc.SelectObject (&darkerpen);
    dc.MoveTo(30, 30);
    dc.LineTo(30, rect.Height());
    dc.MoveTo(30, 30);
    dc.LineTo(rect.Width(), 30);
    dc.MoveTo(26, rect.Height()/2 + 15);
    dc.LineTo(rect.Width(), rect.Height()/2 + 15);
    CPen brighterpen (PS_SOLID, 1, RGB (nShade1, nShade1, nShade1));
    pOldPen = dc.SelectObject (&brighteren);
    for(int i = 1; i < 10; i++)
    {
        dc.MoveTo(30, rect.Height()/2 + 15 + i*Gridis);
        dc.LineTo(rect.Width(), rect.Height()/2 + 15 + i*Gridis);
        dc.MoveTo(30, rect.Height()/2 + 15 - i*Gridis);
        dc.LineTo(rect.Width(), rect.Height()/2 + 15 - i*Gridis);
    }
    for(int j = 1; j < yGrid+1 ; j++)
    {
        dc.MoveTo(rect.Width()/2 + 15 + j*Gridis, 30);
        dc.LineTo(rect.Width()/2 + 15 + j*Gridis, rect.Height());
        dc.MoveTo(rect.Width()/2 + 15 - j*Gridis, 30);
        dc.LineTo(rect.Width()/2 + 15 - j*Gridis, rect.Height());
    }
    this->DrawGraph();
    // Do not call CView::OnPaint() for painting messages
}
void CwiseCutView::OnInitialUpdate()
{
    CView::OnInitialUpdate();
    m_pFr = (CMainFrame*)AfxGetApp()->m_pMainWnd;
    AfxBeginThread(DrawLine, this);
}

```