Przemysłowy Instytut Automatyki i Pomiarów
Al. Jerozolimskie 202, 02-486 Warszawa

440

ZESPÓŁ AUTOMATYKI ELETRONICZNEJ BE 10
PRACOWNIA TESTERÓW ELEKTRONICZNYCH

.....
Nazwa ONB/ZNB

Główny wykonawca mgr inż. Tadeusz Goszczyński

Wykonawcy: .....

.....

.....

Zestaw Lon Works do sterowania urządzeniami,
współpracujący z systemem zbierania danych
o mediach energetycznych
Etap 4 :
Oprogramowanie graficzne wizualizacji systemu zbierania danych o zużyciu mediów energetycznych i sterowania odbiornikami energii.
/wersja demonstracyjna uruchomiona do 28.02.94/

(Tytuł pracy, numer i tytuł etapu).

Zleceniodawca Praca statutowa PIAP

.....

Kierownik Pracowni
.....
mgr inż.T.Goszczyński

Z-ca Dyr. d/s
Bad.-Rozwojowych
dr inż.J.Jabłkowski

Kierownik Zespołu
.....
doc.dr inż.J.Korytkowski

Pracę zakończono dnia 10.10.1995

Nr arch. 7243 Nr zlecenia S1527

Analiza deskryptorowa

SYSTEM LON WORKS + ZBIERANIE DANYCH + STEROWANIE

Abstrakt

Dokumentacja zawiera wydruk oprogramowania graficznego dla systemu LonWorks

Tytuły poprzednich sprawozdań

Zestaw LonWorks do sterowania urządzeniami,współpracujący z systemem zbierania danych o mediach energetycznych
Etap 1 : Zaprojektowanie i wykonanie stanowisk sterująco-wykonawczych i instalacja sieci /dokumentacja w postaci wykazu podzespołów i schematu połączeń/
Nr arch.7202
Etap 3 : Oprogramowanie PC do sterowania i zbierania danych o pracy zestawu /dokumentacja w postaci wydruku oprogramowania z komentarzami/
Nr arch.7195

Rozdzielnik

Egz. 1. OIN

Egz. 2. ZAE-1

Egz. 3. ZAE-3

```
// chancfg.cpp
//

#include "stdafx.h"
#include "lonworks.h"
#include "londoc.h"
#include "chancfg.h"
#include "validate.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CChannelCfgDlg dialog

CChannelCfgDlg::CChannelCfgDlg(CChannelCfg *pChannelCfg)
        : CDialog(CChannelCfgDlg::IDD, NULL)
{
        //{{AFX_DATA_INIT(CChannelCfgDlg)
        m_dCoeff = 0;
        m_dDisp = 0;
        m_dHiAlarm = 0;
        m_dLowAlarm = 0;
        m_dUnit = 0;
        m_sName = "";
        m_dNomin = 0;
        //}}AFX_DATA_INIT

        m_pChannelCfg = pChannelCfg;
        m_nChannelNo = 0;
        PrepareData();
}

void CChannelCfgDlg::PrepareData (void)
{
        ASSERT (m_pChannelCfg);
        m_cbUnit.SetCurSel (m_pChannelCfg->m_nUnit);
        m_cbKindConv.SetCurSel (m_pChannelCfg->m_nKindConv);
        m_cbConv.SetCurSel (m_pChannelCfg->m_nConv);

        m_dCoeff = m_pChannelCfg->m_dCoeff;
        m_dDisp = m_pChannelCfg->m_dDisp;
        m_dHiAlarm = m_pChannelCfg->m_dHiAlarm;
        m_dLowAlarm = m_pChannelCfg->m_dLowAlarm;
        m_dUnit = m_pChannelCfg->m_dUnit;
        m_sName = m_pChannelCfg->m_pszName;
        m_dNomin = m_pChannelCfg->m_dNominal;
}

void CChannelCfgDlg::SetData (void)
{
        m_pChannelCfg->m_nUnit = m_cbUnit.GetCurSel ();
        m_pChannelCfg->m_nKindConv = m_cbKindConv.GetCurSel ();
        m_pChannelCfg->m_nConv = m_cbConv.GetCurSel ();

        m_pChannelCfg->m_dCoeff = m_dCoeff;
        m_pChannelCfg->m_dDisp = m_dDisp;
        m_pChannelCfg->m_dHiAlarm = m_dHiAlarm;
        m_pChannelCfg->m_dLowAlarm = m_dLowAlarm;
        m_pChannelCfg->m_dUnit = m_dUnit;
        m_pChannelCfg->m_dNominal = m_dNomin;

        strncpy (m_pChannelCfg->m_pszName, m_sName, MAX_CHANNEL_NAME);
}
```

SPIS PROGRAMOW PRZEDSTAWIONYCH W DOKUMENTACJI

CHANCFG	CPP	3 876	95.06.24	21:05
CHANCHOI	CPP	1 111	95.06.25	14:45
CHARTVW	CPP	9 162	95.06.28	2:00
COMP_DLL	CPP	1 414	95.09.27	18:45
COMPATIB	CPP	1 398	95.09.27	0:23
LONDOC	CPP	12 022	95.10.03	10:07
LONVIEW	CPP	5 879	95.10.22	19:11
LONWORKS	CPP	9 141	95.10.22	19:27
MAINFRM	CPP	6 438	95.10.24	23:05
MEASDOC	CPP	3 592	95.10.24	23:28
MEASTABL	CPP	5 561	95.06.28	2:01
MONCTRLD	CPP	3 216	95.03.03	10:20
MONMEASD	CPP	3 677	95.03.03	10:04
SPLITER	CPP	2 322	95.10.24	23:10
STDAFX	CPP	204	95.02.19	15:56
VALIDATE	CPP	645	95.02.19	20:41
CHANCFG	H	979	95.06.24	14:24
CHANCHOI	H	702	95.06.25	14:45
CHARTVW	H	1 727	95.06.25	11:10
DISPLAY	H	6 283	93.12.09	2:02
HAUIF	H	840	93.12.09	2:02
LDVINTFC	H	1 260	93.12.09	2:02
LONDOC	H	2 582	95.10.24	11:32
LONVIEW	H	1 785	95.09.27	0:06
LONWORKS	H	1 307	95.06.28	10:53
MAINFRM	H	1 344	95.10.24	23:05
MEASDOC	H	1 112	95.06.28	1:55
MEASTABL	H	1 081	95.06.23	21:21
MONCTRLD	H	1 001	95.03.02	10:10
MONMEASD	H	905	95.02.21	15:36
NI_MGMT	H	5 406	93.12.09	2:02
NI_MSG	H	26 465	93.12.09	2:02
RESOURCE	H	4 766	95.10.24	23:10
SPLITER	H	828	95.10.24	23:10
STDAFX	H	299	95.02.19	15:56
VALIDATE	H	95	95.02.19	20:37

```
void CChannelCfgDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CChannelCfgDlg)
        DDX_Control(pDX, IDC_UNIT_COMBO, m_cbUnit);
        DDX_Control(pDX, IDC_KIND_COMBO, m_cbKindConv);
        DDX_Control(pDX, IDC_CONV_COMBO, m_cbConv);
        DDX_Text(pDX, IDC_COEFFICIENT, m_dCoeff);
        DDX_Text(pDX, IDC_DISPLACEMENT, m_dDisp);
        DDX_Text(pDX, IDC_HIGH_ALARM2, m_dHiAlarm);
        DDX_Text(pDX, IDC_LOW_ALARM, m_dLowAlarm);
        DDX_Text(pDX, IDC_UNIT_VALUE, m_dUnit);
        DDX_Text(pDX, IDC_CHAN_NAME, m_sName);
        DDV_MaxChars(pDX, m_sName, 30);
        DDX_Text(pDX, IDC_NOMIN_VALUE, m_dNomin);
        //}}AFX_DATA_MAP
        DDV_MinChars(pDX, m_sName, 1);

        char buf[30];
        sprintf (buf, "Kana| nr %d", m_nChannelNo+1);
        GetDlgItem (IDC_CHANNEL_NO) -> SetWindowText (buf);
}

BEGIN_MESSAGE_MAP(CChannelCfgDlg, CDialog)
        //{{AFX_MSG_MAP(CChannelCfgDlg)
        ON_BN_CLICKED(IDNEXT, OnNext)
        ON_BN_CLICKED(IDPREV, OnPrev)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////////
// CChannelCfgDlg message handlers

BOOL CChannelCfgDlg::OnInitDialog()
{
        CDialog::OnInitDialog();

        // TODO: Add extra initialization here

        PrepareData ();

        return TRUE;  // return TRUE  unless you set the focus to a control
}

void CChannelCfgDlg::OnNext()
{
        // TODO: Add your control notification handler code here
        if (*(m_pChannelCfg + 1) -> m_pszName == '\0')
          {
                MessageBeep (-1);
            return;
          }
        if (UpdateData () == FALSE)
          return;
        SetData ();
        m_nChannelNo ++;
        m_pChannelCfg ++;
        PrepareData ();
        UpdateData (FALSE);
}

void CChannelCfgDlg::OnPrev()
{
        // TODO: Add your control notification handler code here
```

```
        if (m_nChannelNo == 0)
          {
                MessageBeep (-1);
            return;
          }
        if (UpdateData () == FALSE)
          return;
        SetData ();
        m_nChannelNo --;
        m_pChannelCfg --;
        PrepareData ();
        UpdateData (FALSE);
}

void CChannelCfgDlg::OnOK()
{
        // TODO: Add extra validation here

        if (UpdateData () == FALSE)
          return;
        SetData ();
        CDialog::OnOK();
}
```

```
// chanchoi.cpp : implementation file
//

#include "stdafx.h"
#include "lonworks.h"
#include "chanchoi.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CChanChoiceDlg dialog


CChanChoiceDlg::CChanChoiceDlg(CWnd* pParent /*=NULL*/)
        : CDialog(CChanChoiceDlg::IDD, pParent)
{
        //{{AFX_DATA_INIT(CChanChoiceDlg)
        m_nChannelNo = 0;
        m_bPercent = FALSE;
        m_nScale = -1;
        //}}AFX_DATA_INIT
}

void CChanChoiceDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CChanChoiceDlg)
        DDX_Radio(pDX, IDC_CHANNEL1, m_nChannelNo);
        DDX_Check(pDX, IDC_PERCENT, m_bPercent);
        DDX_Radio(pDX, IDC_SCALE1, m_nScale);
        //}}AFX_DATA_MAP
}

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


/////////////////////////////////////////////////////////////////////////////
// CChanChoiceDlg message handlers
```

```
// chartvw.cpp
//

#include "stdafx.h"
#include "lonworks.h"
#include "chartvw.h"
#include "lonview.h"
#include "measdoc.h"
#include "londoc.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] =   FILE  ;
#endif

/////////////////////////////////////////////////////////////////////////////
// CChartView

IMPLEMENT_DYNCREATE(CChartView, CScrollView)

CChartView::CChartView()
{
}

CChartView::~CChartView()
{
}

BEGIN_MESSAGE_MAP(CChartView, CScrollView)
        //{{AFX_MSG_MAP(CChartView)
        ON_MESSAGE(WM_OPEN_FILE, OnOpenFile)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

BOOL CChartView::OnPreparePrinting(CPrintInfo* pInfo)
{
        pInfo->SetMaxPage(1);       // one page printing/preview
        return DoPreparePrinting(pInfo);
}

void CChartView::Print()
{
        OnFilePrint();
}

void CChartView::Preview()
{
        OnFilePrintPreview();
}

/////////////////////////////////////////////////////////////////////////////
// CChartView drawing

void CChartView::OnInitialUpdate()
{
        CScrollView::OnInitialUpdate();

        CSize sizeTotal;
        // TODO: calculate the total size of this view
        sizeTotal.cx = sizeTotal.cy = 100;
        SetScrollSizes(MM_TEXT, sizeTotal);

        CMeasDoc* pDoc = (CMeasDoc* ) GetDocument();
        pDoc->m_nChannelNo = 0;
        pDoc->OnHistoriaKanal ();
}
```

```
static COLORREF colorTable[] =
{
        RGB(255,   0,   0),     // Red
        RGB(  0, 255,   0),     // Green
        RGB(  0,   0, 255),     // Blue
        RGB(255, 255,   0),     // Yellow
        RGB(255,   0, 255),     // Magenta
        RGB(  0, 255, 255),     // Cyan
        RGB(  0,   0,   0),     // Black
        RGB(128, 128, 128),     // Grey
        RGB(128, 128,   0),     // Brown
        RGB(128,   0, 128),     // Purple
        RGB(  0, 128, 128)      // Aqua
};

extern const char *pszUnitOfChannel[20];

void CChartView::OnDraw(CDC* pDC)
{
        CMeasDoc* pDoc = (CMeasDoc* ) GetDocument();
        // TODO: add draw code here
        double dMaxValue = GetMaxValue();
        if (pDoc->m_bChartInPercent)
                switch (pDoc->m_nScale)
                {
                        case 1: dMaxValue = 120; break;
                        case 2: dMaxValue = 150; break;
                        case 3: dMaxValue = 200; break;
                }

        int nValuePerTick = GetTickValue( dMaxValue, 5);

        CSize sizeGraph;
        CSize sizePixelsPerTick;
        CPoint ptOrigin;
        int nCharHeight = GetChartMetrics(pDC, pDoc->m_nDataCols, 5,
                                                        sizeGraph, sizePixelsPer
Tick, ptOrigin);

        char buf[100];
        sprintf (buf, "Kana| %d [%s]", pDoc->m_nChannelNo + 1,
                                (pDoc->m_bChartInPercent)    ? "% warto£ci nominalne
" :
                                        pszUnitOfChannel[CLonworksDoc::channelCfg[pDoc-
m_nChannelNo].m_nUnit]);
        CString s (buf);
        CRect rectGraph (CPoint (0,0), sizeGraph);
        pDC->DrawText(s, -1, rectGraph, DT_TOP | DT_SINGLELINE | DT_CENTER);

        PlotAxes(pDC, ptOrigin, sizePixelsPerTick, pDoc->m_nDataCols, 5, TICKS_BOTH);

        PlotCaptions(pDC, ptOrigin, sizePixelsPerTick, nValuePerTick,
                                                6, nCharHeight, TRUE, FALSE);

        PlotCaptions(pDC, ptOrigin, sizePixelsPerTick, 1,
                                                pDoc->m_nDataCols, nCharHeight, FALSE,
RUE);

//      for (int nRow = 0; nRow < pDoc->m_nDataRows; nRow++)
        int nRow = pDoc->m_nChannelNo;
        {
                CPen pen(PS_SOLID, 3, colorTable[nRow]);
                CPen* pOldPen = pDC->SelectObject(&pen);

                double* pdData = pDoc->m_pdData + nRow;
```

```
                int nHPos = ptOrigin.x;
                for (WORD nCol = 0; nCol < pDoc->m_nDataCols; nCol++)
                {
                        double dData = (pDoc->m_bChartInPercent) ? *pdData/pDoc->m_dNomi
nal*100 : *pData;
                        int nVPos = (int) (dData/nValuePerTick*sizePixelsPerTick.cy);
                        if (nCol == 0)
                                pDC->MoveTo(nHPos, ptOrigin.y - nVPos);
                        else
                                pDC->LineTo(nHPos, ptOrigin.y - nVPos);
                        nHPos += sizePixelsPerTick.cx;
                        pdData += pDoc->m_nDataRows;
                }
                pDC->SelectObject(pOldPen);
        }
}

/////////////////////////////////////////////////////////////////////////////
// CChartView message handlers

/////////////////////////////////////////////////////////////////////////////
double CChartView::GetMaxValue()
{
        // Scan data to determine max value
        CMeasDoc* pDoc = (CMeasDoc* ) GetDocument();
        double *pdData = pDoc->m_pdData + pDoc->m_nChannelNo;
        double dMax    = 0;

        for (WORD i = 0; i < pDoc->m_nDataCols; i++)
        {
                double dData = (pDoc->m_bChartInPercent) ? *pdData/pDoc->m_dNominal*100
: *pdData;
                if (dData > dMax)
                        dMax = dData;
                pdData += pDoc->m_nDataRows;
        }
        return dMax;
}

static int nValidGraduations[] =
{
        1, 2, 5, 10, 20, 25, 30, 40, 50, 100, 200, 500, 1000, 2000, 5000, 0
};

int CChartView::GetTickValue(double dMaxValue, int nNumDiv)
{
        // Determine values for vertical ticks
        int nValuePerTick = (int) ((dMaxValue + nNumDiv - 1) / nNumDiv);
        for (int i = 0; nValidGraduations[i] > 0; i++)
        {
                if (nValidGraduations[i] >= nValuePerTick)
                {
                        return nValidGraduations[i];
                }
        }
        return nValidGraduations[i-1];
}

//
// GetChartMetrics determines several things:
// Returns Character Height
// &sizeGraph contains overall dimensions of the graph (may not be window size)
// &sizePixelsPerTick contains pixels per tick in both directions
// &ptOrigin contains the origin point
//
```

```
//
//  pDC    - DC for display
//  nVDiv - Number of vertical Divisions
//  nHDiv - Number of horizontal Divisions
//
int CChartView::GetChartMetrics(CDC* pDC, int nHDiv, int nVDiv,
        CSize& sizeGraph, CSize& sizePixelsPerTick, CPoint& ptOrigin)
{
        CSize sizeText = pDC->GetTextExtent("0", 1);

        // Determine minimum size for graph
        sizeGraph = sizeText;

        // Minimum size of a Division == 5 chars
        // Allow 5 chars for Vertical Captions and 2.5 chars on each side of chart

//      sizeGraph.cx *= (5 * nHDiv) + 10; G.G
        sizeGraph.cx *= (2 * nHDiv);
        sizeGraph.cy *= nVDiv + 3;

        // Grow size if permitted by display area
        CRect rect = GetDisplayRect(pDC);
        if (rect.Width() > sizeGraph.cx)
                sizeGraph.cx = rect.Width();
        if (rect.Height() > sizeGraph.cy)
                sizeGraph.cy = rect.Height();

        sizePixelsPerTick.cx = (sizeGraph.cx - 10 * sizeText.cx) / nHDiv;
        sizePixelsPerTick.cy = (sizeGraph.cy -  3 * sizeText.cy) / nVDiv;

        ptOrigin.x = sizeText.cx * 15 / 2;
        ptOrigin.y = sizePixelsPerTick.cy * nVDiv + sizeText.cy * 3 / 2;

        return sizeText.cy;
}

void CChartView::PlotCaptions(CDC* pDC, CPoint ptOrigin,
                        CSize sizePixelsPerTick, int nValuePerTick,
                        int nNumTicks, int nCharHeight, BOOL bVert, BOOL bAlpha)
{
        char buf[80];

        int nTickSize = (bVert ? sizePixelsPerTick.cx : sizePixelsPerTick.cy) / 30;
        if (nTickSize < 2)
                nTickSize = 2;

        int nPos  = bVert ? ptOrigin.y : ptOrigin.x;
        int nStep = bVert ? -sizePixelsPerTick.cy : sizePixelsPerTick.cx;

        pDC->SetTextAlign(bVert ? TA_RIGHT : TA_CENTER);

        int nValue = 0;

        CMeasDoc* pDoc = (CMeasDoc* ) GetDocument();

        for (int i = 0; i < nNumTicks; i++)
        {
                if (bAlpha)
                {
                        if (bVert)
                                ASSERT(TRUE);//wsprintf(buf, "%c", nValue);
                        else
                        {
                                CString *psOpis;
                                if (pDoc->m_mapDsc.Lookup ((WORD) i+1, (void*&) psOpis))
                                  strncpy (buf, *psOpis, 80);
```


11

```
                        else
                                buf[0]='\0';
                        }
                }
                else
                        wsprintf(buf, "%d", nValue);

                if (bVert)
                        pDC->TextOut(ptOrigin.x - 2 * nTickSize,
                                                                        nPos - n
CharHeight / 2, buf);
                else
                        pDC->TextOut(nPos, ptOrigin.y + nCharHeight / 2, buf);

                nValue += nValuePerTick;
                nPos   += nStep;
        }
}


void CChartView::PlotAxes(CDC* pDC, CPoint ptOrigin, CSize sizePixelsPerTick,
                                int nHorzTicks, int nVertTicks, int nDrawTicks)
{
        int nHeight = sizePixelsPerTick.cy * nVertTicks;
        int nWidth  = sizePixelsPerTick.cx * nHorzTicks;

        // Draw Axes
        pDC->MoveTo(ptOrigin);
        pDC->LineTo(ptOrigin.x, ptOrigin.y - nHeight);
        pDC->MoveTo(ptOrigin);
        pDC->LineTo(ptOrigin.x + nWidth, ptOrigin.y);

        if (nDrawTicks & TICKS_VERT)
        {
                // Draw Vertical Ticks
                int nTickSize = (sizePixelsPerTick.cx / 30);
                if (nTickSize < 2)
                        nTickSize = 2;

                int nVPos = ptOrigin.y;
                int xLeft  = ptOrigin.x - nTickSize;
                int xRight = ptOrigin.x + nTickSize + 1;

                for (int i = 0; i <= nVertTicks; i++)
                {
                        pDC->MoveTo(xLeft, nVPos);
                        pDC->LineTo(xRight, nVPos);
                        nVPos -= sizePixelsPerTick.cy;
                }
        }

        if (nDrawTicks & TICKS_HORZ)
        {
                // Draw Horizontal Ticks
                int nTickSize = (sizePixelsPerTick.cy / 30);
                if (nTickSize < 2)
                        nTickSize = 2;

                int nHPos = ptOrigin.x;
                int yTop    = ptOrigin.y - nTickSize;
                int yBottom = ptOrigin.y + nTickSize + 1;

                for (int i = 0; i <= nHorzTicks; i++)
                {
                        pDC->MoveTo(nHPos, yTop);
                        pDC->LineTo(nHPos, yBottom);
```

```
                    }
            }
}

CRect CChartView::GetDisplayRect(CDC* pDC)
{
        CRect rect;

        if (pDC->IsPrinting())
        {
                rect.left = rect.top = 0;
                rect.right  = pDC->GetDeviceCaps(HORZRES);
                rect.bottom = pDC->GetDeviceCaps(VERTRES);
        }
        else
        {
                GetClientRect(&rect);
        }
        return rect;
}

LRESULT CChartView::OnOpenFile (WPARAM, LPARAM fileName)
{
        CMeasDoc* pDoc = (CMeasDoc* ) GetDocument();
        pDoc->Serialize ((LPCSTR) fileName);
        return 0L;
}
```

```
#include <windows.h>
#include <stdio.h>
#include <string.h>
#define ANSI            /* Comment out for UNIX version     */
#include <stdarg.h>
#include <stdlib.h>

#define WM_ADD_LOG (WM_USER + 1)

void perror (const char *pszText)
{
  char buf[200];
  strncpy (buf, pszText, 160);
  strcat (buf, "\nContinue anyway?");
  if (MessageBox (NULL, buf, "Error", MB_YESNO | MB_ICONSTOP) == IDNO)
    exit (1);
}

int printf( const char *format, ... )
{
   char buf[1000];
   char buf2[1000];

   va_list marker;

   va_start( marker, format );     /* Initialize variable arguments. */

   int result = vsprintf (buf, format, marker);
   va_end( marker );               /* Reset variable arguments.      */

   unsigned i=0;
   for (unsigned i=0; i<strlen (buf); i++)
     {
       if (buf[i] == '\n')
         buf2[j++] = '\r';
       buf2[j++] = buf[i];
     }

   buf2[j] = '\0';

   SendMessage (HWND_BROADCAST, WM_ADD_LOG, 0, (LONG) (const char far *) buf2);

   return result;
}


char *gets( char *buffer)
{
  return NULL;
}

char keys[] = "NTE";

extern "C" int getche (void)
{
  static pos = 0;
  int c = keys[pos];
  pos = (pos + 1) % strlen (keys);
  return c;
}

extern "C" int getch (void)
{
  return getche ();
}
```

```
extern "C" int kbhit (void)
{
  static int count = 0;
  count ++;
  if (count == 0)
    return 1;
  else
    return 0;
}
```

```
#include "stdafx.h"
#define ANSI            /* Comment out for UNIX version     */
#include <stdarg.h>

#include "lonview.h"

void perror (const char *pszText)
{
  char buf[200];

  strncpy (buf, pszText, 160);
  strcat (buf, "\nContinue anyway?");
  if (AfxMessageBox (buf, MB_YESNO | MB_ICONSTOP) == IDNO)
    exit (1);
}

int printf( const char *format, ... )
{
   char buf[1000];
   CString buf2;

   va_list marker;

   va_start( marker, format );     /* Initialize variable arguments. */

   int result = vsprintf (buf, format, marker);
   va_end( marker );              /* Reset variable arguments.      */

   for (unsigned i=0; i<strlen (buf); i++)
     {
       if (buf[i] == '\n')
         buf2 += '\r';
       buf2 += buf[i];
     }

   SendMessage (HWND_BROADCAST, WM_ADD_LOG, 0, (LONG) (const char far *) buf2);

   static CStdioFile logFile ("lonworks.log", CFile::modeWrite | CFile::modeCreate);

   logFile.WriteString (buf2);

//   AfxMessageBox (buf);
   return result;
}



char *gets( char *buffer)
{
  return NULL;
}


char keys[] = "NTE";

extern "C" int getche (void)
{
  static pos = 0;
  int c = keys[pos];
  pos = (pos + 1) % strlen (keys);
  return c;
}

extern "C" int getch (void)
{
  return getche ();
```

```
// londoc.cpp : implementation of the CLonworksDoc class
//

#include "stdafx.h"
#include "lonworks.h"

#include "londoc.h"

#include "chancfg.h"
#include "monmeasd.h"
#include "monctrld.h"

#include "chartvw.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

extern "C" void FAR PASCAL SetGrzeje(int grzejeX, int value);
extern "C" void FAR PASCAL ExecuteNV_update( int nr_danej ); // w applmsg.c


/////////////////////////////////////////////////////////////////////////////
// CLonworksDoc

IMPLEMENT_DYNCREATE(CLonworksDoc, CDocument)

BEGIN_MESSAGE_MAP(CLonworksDoc, CDocument)
        //{{AFX_MSG_MAP(CLonworksDoc)
        ON_COMMAND(ID_OPTIONS_CHANNELS, OnOptionsChannels)
        ON_COMMAND(ID_MONITOR_MONITOROWANIEPOMIARW, OnMonitorMonitorowanie)
        ON_UPDATE_COMMAND_UI(ID_MONITOR_MONITOROWANIEPOMIARW, OnUpdateMonitorMonitorowa
ie)
        ON_COMMAND(ID_MONITOR_STEROWANIEPOMIARAMI, OnMonitorSterowanie)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
CChannelCfg CLonworksDoc::channelCfg[] =
 { { "Ciep|omierz",       8,  100, 0, 1, 3.60, 0.0, 06.0, 0.4, 0.1 },
   { "Wodomierz ciep",    9,  200, 1, 0, 1.00, 0.0, 20.0, 0.5, 0.1 },
   { "Wodomierz zimn",    9,  300, 1, 0, 1.00, 0.0, 01.0, 0.5, 0.1 },
   { "Licznik elektr Sz", 10, 400, 0, 1, 0.02, 0.0, 55.0, 0.4, 1.0 },
   { "Licznik elektr DAM",10, 500, 0, 1, 0.01, 0.0, 45.0, 0.4, 1.0 },
   { "" }
 };

CMonControl CLonworksDoc::monControl[] =
 { { 20, 23.5 },
   { 19, 25.2 }};

CMonControl CLonworksDoc::monInfo[] =
 { { 1, 23.2 },
   { 0, 25.7 }};

CString GetAppPath()
{
        CString sPath;
        char fullPath[_MAX_PATH];
        if (GetModuleFileName(AfxGetInstanceHandle(), fullPath, _MAX_PATH))
        {
                char drive[_MAX_DRIVE];
                char dir[_MAX_DIR];
                _splitpath(fullPath, drive, dir, NULL, NULL);
                sPath = drive;
```

```
                sPath += dir;
        }
        return sPath;
}

// CLonworksDoc construction/destruction

CLonworksDoc::CLonworksDoc()
{
        // TODO: add one-time construction code here
        pMonMeasDlg = NULL;
        pMonCtrlDlg = NULL;

        for (int i=0; i<TOTAL_CHANNEL; i++)
          {
            for (int j=0; j<3; j++)
              dMeasVal [i][j] = 0;
            bFocused[i] = 0;
            bChannelInit[i] = FALSE;
            dMax[i][0] = 0;
            dMax[i][1] = 0;
            dMax[i][2] = 0;
          }
        bEnable [0] = TRUE;
        bEnable [1] = FALSE;

        timeLog = CTime::GetCurrentTime();
        m_sLogDir = GetAppPath();
}

BOOL CLonworksDoc::CanCloseFrame( CFrameWnd* /*pFrame*/ )
{
        return FALSE;
}

CLonworksDoc::~CLonworksDoc()
{
}

BOOL CLonworksDoc::OnNewDocument()
{
        if (!CDocument::OnNewDocument())
                return FALSE;

        // TODO: add reinitialization code here
        // (SDI documents will reuse this document)

        return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CLonworksDoc serialization

void CLonworksDoc::Serialize(CArchive& ar)
{
        if (ar.IsStoring())
        {
                // TODO: add storing code here
        }
        else
        {
                // TODO: add loading code here
        }
}

/////////////////////////////////////////////////////////////////////////////
```

```
// CLonworksDoc d:

#ifdef  DEBUG
void CLonworksDoc::AssertValid() const
{
        CDocument::AssertValid();
}

void CLonworksDoc::Dump(CDumpContext& dc) const
{
        CDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CLonworksDoc commands

void CLonworksDoc::OnOptionsChannels()
{
        // TODO: Add your command handler code here
        CChannelCfg tempChannelCfg[TOTAL_CHANNEL+1];
        memcpy (tempChannelCfg, channelCfg, sizeof (tempChannelCfg));
        CChannelCfgDlg dlg (tempChannelCfg);
        if (dlg.DoModal() != IDOK)
          return;
        memcpy (channelCfg, tempChannelCfg, sizeof (tempChannelCfg));

        if (pMonMeasDlg == NULL)
          return;

        pMonMeasDlg -> UpdateData (FALSE);
}

void CLonworksDoc::OnMonitorMonitorowanie()
{
        // TODO: Add your command handler code here
        ASSERT (pMonMeasDlg == NULL);
//      if (pMonMeasDlg != NULL)
//        pMonMeasDlg -> DestroyWindow ();
        pMonMeasDlg = new CMonMeasDlg (channelCfg, &dMeasVal[0][0], &bFocused[0], &pMonM
easDlg);
}


#ifdef _NOLAN
#define ExecuteNV_update(int)
#endif

void CLonworksDoc::UpdateMeas(CChannelMeas *pChanMeas)
{
  UpdateLog ();

  static CTime zeg_te1, zeg_te2;
  static CTimeSpan  roznica;
  int roznica_grzania;
  double  roznica_zeg;
  static unsigned licznik = 0;
  static unsigned priorytet = 6;
  static unsigned grzeje1 = 0;
  static unsigned grzeje2 = 0;

  CChannelMeas pom = *pChanMeas;

  WORD pom_kanal = pom.nChannelNo -1;

  CTime zegar_new = CTime::GetCurrentTime();
```

```
  if(pom_kanal<TOTAL_CHANNEL)
  {
        if(bChannelInit[pom_kanal]==FALSE)
        {
          zegar_old[pom_kanal] = zegar_new;
          bChannelInit[pom_kanal]=TRUE;
          if(pom_kanal==4)
          {  // tymczas "zerowanie" 1.elektr
            zeg_te1 = zegar_new;
            zeg_te2 = zegar_new;
          }
          return;
        }

        roznica = zegar_new - zegar_old[pom_kanal];
        roznica_zeg = (double) roznica.GetTotalSeconds();
        zegar_old[pom_kanal] = zegar_new;

        dMeasVal [pom_kanal][2] +=  channelCfg[pom_kanal].m_dUnit;
        if(roznica_zeg != 0)
        {
              double dValue = 3600.0 * channelCfg[pom_kanal].m_dUnit
                    * channelCfg[pom_kanal].m_dCoeff / roznica_zeg;
              dMeasVal [pom_kanal][1] = dValue;
              if (dMeasVal [pom_kanal][0] < dValue) // wartoœæ wyœwietlana w okienku d
ialogowym
                    dMeasVal [pom_kanal][0] = dValue;
              for (WORD nType = 0; nType < 3; nType++)
              {
                    if (dMax [pom_kanal][nType] < dValue) // wartoœæ zapisywana do l
ogu
                          dMax[pom_kanal][nType] = dValue;
              }
        }

        bFocused[pom_kanal] = (dMeasVal [pom_kanal][1] >
                              channelCfg[pom_kanal].m_dHiAlarm );
  }
  else
  {

        if(pom.nChannelNo == 6 )
        if( priorytet == 6 )
        priorytet =  7;
     else
        priorytet =  6;

//tg
        if(pom.nChannelNo == 6 &&  priorytet == 6   && grzeje2 == 0 )
        {
          monInfo [0].m_dTemp = pom.dValue;
          if(monInfo [0].m_dTemp < monControl [0].m_dTemp)    {
           SetGrzeje(1, 1);
           grzeje1 = 1;
          }
          else
          {
            zeg_te1= zegar_new;
            SetGrzeje(1, 0);
            grzeje1 = 0          ;
          }

          roznica = zegar_new - zeg_te1;
          roznica_grzania = (int) roznica.GetTotalSeconds();
          monInfo [0].m_nTime =  roznica_grzania;
```

```
          if(roznica grzania > monControl [0].m_nTime ) {
        zeg_te1= zegar_new;
            SetGrzeje(1, 0);
            grzeje1 = 0;
          }

          ExecuteNV_update(7);

        }
// never tg
        if(pom.nChannelNo == 7   &&  priorytet == 7  && grzeje1 == 0)
        {
          monInfo [1].m_dTemp = pom.dValue;
          if(monInfo [1].m_dTemp < monControl [1].m_dTemp)  {
           SetGrzeje(2, 1);
            grzeje2 = 1;
          }
          else
          {
            zeg_te2= zegar_new;  // zerowanie roznicy czasu
            SetGrzeje(2, 0);
              grzeje2 = 0;
          }

          roznica = zegar_new - zeg_te2;
          roznica_grzania = (int) roznica.GetTotalSeconds();
          monInfo [1].m_nTime =  roznica_grzania;

          if(roznica_grzania > monControl [1].m_nTime )  {
        zeg_te2= zegar_new;
            SetGrzeje(2, 0);
              grzeje2 = 0;
          }
          ExecuteNV_update(8);

        }  // if kanal 7

        bEnable [0] = monInfo [0].m_dTemp < monControl [0].m_dTemp;
        bEnable [1] = monInfo [1].m_dTemp < monControl [1].m_dTemp;
  }// else kanal<=5

  if (pMonMeasDlg != NULL)
          pMonMeasDlg -> UpdateData (FALSE);

  if (pMonCtrlDlg != NULL)
          pMonCtrlDlg -> PrepareData ();
}

void CLonworksDoc::UpdateLog (void)
{
// za|ofenie fe w kafdej godzinie co najmniej jeden pomiar
#ifdef _NOLAN
        static CTime currentTime = CTime::GetCurrentTime ();

        currentTime += CTimeSpan (0, 0, 4, 0); // 4 minuty
#else
        CTime currentTime = CTime::GetCurrentTime ();
#endif
        int hour1 = currentTime.GetHour(),
                hour2 = timeLog.GetHour ();

//      char ttt[100];
//      sprintf (ttt, "current: %d, log: %d", hour1, hour2);
//      AfxMessageBox (ttt);
```

```
        if (hour1 != hour2)
        { // zapisz probke
//              AfxMessageBox ("WchodzΩ");
                WriteLog (0, timeLog);
                int day1 = currentTime.GetDay(),
                        day2 = timeLog.GetDay ();
                if (day1 != day2)
                        {
                                WriteLog (1, timeLog);
                                int month1 = currentTime.GetMonth(),
                                        month2 = timeLog.GetMonth ();
                                if (month1 != month2)
                                        WriteLog (2, timeLog);
                        }
                timeLog = currentTime;
        }
}

void CLonworksDoc::WriteLog(BYTE nLogType, const CTime &currentTime)
{
                char pszFileName[13];
                char *pszFormats[] = {"%04d%02d%02d.hdz",
                                                        "%04d-%02d.hmi",
                                                        "%04d.hro"};
                sprintf (pszFileName, pszFormats[nLogType], currentTime.GetYear(),
                                                        currentTime.GetMonth(), currentT
ime.GetDay());

                CString sFileName (m_sLogDir);
                sFileName += pszFileName;

                int nSampleNo;
                nSampleNo = 1 + GetPrivateProfileInt ("Parametry", "LiczbaProbek", 0, s'
ileName);

                char pszValue[20];

                if (nSampleNo == 1)
                { // nowy plik
                        sprintf (pszValue, "%d", TOTAL_CHANNEL);
                        WritePrivateProfileString ("Parametry", "LiczbaKanalow", pszValu
e, sFileName);
                }

                sprintf (pszValue, "%d", nSampleNo);
                WritePrivateProfileString ("Parametry", "LiczbaProbek", pszValue, sFileN
ame);

                char pszSection[20];
                sprintf (pszSection, "Probka%d", nSampleNo);

                int nValue;

                switch (nLogType)
                {
                        case 0: nValue = currentTime.GetHour(); break;
                        case 1: nValue = currentTime.GetDay(); break;
                        case 2: nValue = currentTime.GetMonth(); break;
                }

                sprintf (pszValue, "%d", nValue);

                WritePrivateProfileString (pszSection, "Opis", pszValue, sFileName);

                char pszEntry[10];
                for (WORD nChannelNo = 0; nChannelNo < TOTAL_CHANNEL; nChannelNo++)
```

```
                        sprintf (pszEntry, "Kanal%d", nChannelNo+1);
                        sprintf (pszValue, "%lf", dMax[nChannelNo][nLogType]); // maksy
alna wartoś¿ z godziny, dnia lub miesi¶ca
                        WritePrivateProfileString (pszSection, pszEntry, pszValue, sFile
Name);
                        dMax[nChannelNo][nLogType] = 0; // bo nowa godzina, dzieł lub  m
iesi¶c
                }
}

void CLonworksDoc::OnUpdateMonitorMonitorowanie(CCmdUI* pCmdUI)
{
        // TODO: Add your command update UI handler code here
        pCmdUI -> Enable (pMonMeasDlg == NULL);
}

void CLonworksDoc::OnMonitorSterowanie()
{
        // TODO: Add your command handler code here
        CMonControl tempMonControl[2];

        tempMonControl[0] = monControl[0];
        tempMonControl[1] = monControl[1];

        CMonCtrlDlg dlg (&bEnable[0], tempMonControl, monInfo);
        pMonCtrlDlg = &dlg;

        BOOL bOK = (dlg.DoModal() == IDOK);

        pMonCtrlDlg = NULL;

        if (bOK)
          {
                monControl[0] = tempMonControl[0];
                monControl[1] = tempMonControl[1];
          }
}
```

```
// lonview.cpp : implementation of the CLonworksView class
//

#include "stdafx.h"
#include "lonworks.h"

#include "londoc.h"
#include "lonview.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] =  __FILE__;
#endif

#define OutOfMemory()


/////////////////////////////////////////////////////////////////////////////
// CLonworksView

IMPLEMENT_DYNCREATE(CLonworksView, CEditView)

BEGIN_MESSAGE_MAP(CLonworksView, CEditView)
        //{{AFX_MSG_MAP(CLonworksView)
                // NOTE - the ClassWizard will add and remove mapping macros here.
                //    DO NOT EDIT what you see in these blocks of generated code!
        //}}AFX_MSG_MAP
        // Standard printing commands
        ON_COMMAND(ID_FILE_PRINT, CEditView::OnFilePrint)
        ON_COMMAND(ID_FILE_PRINT_PREVIEW, CEditView::OnFilePrintPreview)
        ON_MESSAGE(WM_UPDATE_MEAS, OnUpdateMeas)
        ON_MESSAGE(WM_ADD_LOG,OnAddLog)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CLonworksView construction/destruction

CLonworksView::CLonworksView()
{
        // TODO: add construction code here
        m_bFull = FALSE;
}

CLonworksView::~CLonworksView()
{
}

/////////////////////////////////////////////////////////////////////////////

// CLonworksView drawing

void CLonworksView::OnInitialUpdate()
{
        CEditView::OnInitialUpdate();

        GetEditCtrl().SetReadOnly();

        CSize sizeTotal;
        // TODO: calculate the total size of this view
        sizeTotal.cx = sizeTotal.cy = 100;
//      SetScrollSizes(MM_TEXT, sizeTotal);
}

void CLonworksView::OnDraw(CDC* /*pDC*/)
{
        CLonworksDoc* pDoc = GetDocument();
```


24

```
        ASSERT_VALID( ... );

        // TODO: add draw code for native data here
}

/////////////////////////////////////////////////////////////////////////////
// CLonworksView printing

BOOL CLonworksView::OnPreparePrinting(CPrintInfo* pInfo)
{
        // default preparation
        return DoPreparePrinting(pInfo);
}

void CLonworksView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
        // TODO: add extra initialization before printing
}

void CLonworksView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
        // TODO: add cleanup after printing
}

/////////////////////////////////////////////////////////////////////////////
// CLonworksView diagnostics

#ifdef _DEBUG
void CLonworksView::AssertValid() const
{
        CEditView::AssertValid();
}

void CLonworksView::Dump(CDumpContext& dc) const
{
        CEditView::Dump(dc);
}

CLonworksDoc* CLonworksView::GetDocument() // non-debug version is inline
{
        ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CLonworksDoc)));
        return (CLonworksDoc*)m_pDocument;
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CLonworksView message handlers

LONG CLonworksView::OnUpdateMeas( WPARAM, LPARAM lParam )
{
        CLonworksDoc* pDoc = GetDocument();
        if (pDoc)
          pDoc -> UpdateMeas((CChannelMeas *)lParam);
        return 0;
}

BOOL CLonworksView::CopyToClipboard (char far * lpszTextToCopy)
{
  HANDLE hData;
  LPSTR lpData;


                        // Allocate memory and copy the string to it

                        if ((hData
                             = GlobalAlloc(GMEM_MOVEABLE, (DWORD) strlen(lpszTextToCopy)
```

```
))==0)
                        {
                            OutOfMemory();
                            return (TRUE);
                        }
                        if ((lpData = (LPSTR) GlobalLock(hData))==0)
                        {
                            OutOfMemory();
                            return (TRUE);
                        }

                        lstrcpy(lpData, lpszTextToCopy);
//                        lstrcpy(lpData + strlen (lpszTextToCopy), "\r\n");
                        GlobalUnlock(hData);

                        // Clear the current contents of the clipboard, and set
                        // the data handle to the new string.

                        if (OpenClipboard())
                        {
                            EmptyClipboard();
                            SetClipboardData(CF_TEXT, hData);
                            CloseClipboard();
                        }
                        hData = NULL;

                        return FALSE;
}

#define MAX_LINE_NO 300

LRESULT CLonworksView::OnAddLog( WPARAM, LPARAM lpszLog)
{
  if (m_bFull)
    return 0L;

  GetEditCtrl().SetReadOnly(FALSE);

  int nLineNo = GetEditCtrl().GetLineCount ();
  if (nLineNo >= MAX_LINE_NO)
    CopyToClipboard ( "Log buffer full !");
  else
    CopyToClipboard ((char far *) lpszLog);

  GetEditCtrl().Paste();
  GetEditCtrl().SetReadOnly();

  if (nLineNo >= MAX_LINE_NO)
    {
          m_bFull = TRUE;
          POSITION pos = AfxGetApp()->m_templateList.GetHeadPosition();
          AfxGetApp()->m_templateList.GetNext(pos);
          ((CDocTemplate *) (AfxGetApp()->m_templateList.GetAt(pos))) -> OpenDocumentFi
e(NULL);
        }

  return (LRESULT) this;
}

int CLonworksView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
        if (CEditView::OnCreate(lpCreateStruct) == -1)
                return -1;

        // TODO: Add your specialized creation code here
```

```
        // load status bar
        LOGFONT logfont;
        memset(&logfont, 0, sizeof(logfont));

        logfont.lfHeight = -12;
        logfont.lfWeight = FW_NORMAL;
        logfont.lfPitchAndFamily = FIXED_PITCH ;
        static char BASED_CODE szFaceName[] = "Courier New";
        lstrcpy(logfont.lfFaceName, szFaceName);

        if (m_font.CreateFontIndirect(&logfont))
        {
                SetFont(&m_font, FALSE);
                                // no need to redraw since invisible
        }

        return 0;
}
```

```
// lonworks.cpp : Defines the class behaviors for the application

#include "stdafx.h"
#include "lonworks.h"

#include "mainfrm.h"
#include "londoc.h"
#include "lonview.h"
#include "chartvw.h"
#include "spliter.h"
#include "measdoc.h"

#include "meastabl.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

extern "C" BOOL FAR PASCAL ExecutePom( CChannelMeas *p);
extern "C" int  FAR PASCAL ExecuteHaIni();
extern "C" void FAR PASCAL ExecuteHaExit();

class CLonDocTemplate : public CMultiDocTemplate
{
  public:
    virtual BOOL GetDocString( CString& rString, enum DocStringIndex index ) const;
    CLonDocTemplate (UINT nIDResource, CRuntimeClass* pDocClass,
                                        CRuntimeClass* pFrameClass, CRuntimeClass* pVi
ewClass) :
                CMultiDocTemplate(nIDResource, pDocClass, pFrameClass, pViewClass)
          {}
  static int nType;
  static char *filterName[];
  static char *filterExt[];
};

int CLonDocTemplate::nType = 0;

char *CLonDocTemplate::filterName[] = {
        "Historia dnia (*.HDZ)",
        "Historia tygodnia (*.HTY)",
        "Historia miesi¶ca (*.HMI)",
        "Historia roku (*.HRO)",
        "Prognoza dnia (*.PDZ)",
        "Prognoza tygodnia (*.PTZ)",
        "Prognoza miesi¶ca (*.PMI)",
        "Prognoza roku (*.PRO)"};

char *CLonDocTemplate::filterExt[] = {
        ".HDZ",
        ".HTY",
        ".HMI",
        ".HRO",
        ".PDZ",
        ".PTY",
        ".PMI",
        ".PRO"};


BOOL CLonDocTemplate::GetDocString( CString& rString, enum DocStringIndex index ) const
{
  switch (index)
    {
```

```
        case CDocTemplate::filterExt : nString = 4; break;
        default: return CMultiDocTemplate::GetDocString (rString, index);
      }
}

/////////////////////////////////////////////////////////////////////////////
// CLonworksApp

BEGIN_MESSAGE_MAP(CLonworksApp, CWinApp)
        //{{AFX_MSG_MAP(CLonworksApp)
        ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
        ON_COMMAND(ID_HISTORIA_WYKRESY_DZIE, OnHistoriaWykresyDzie)
        ON_COMMAND(ID_HISTORIA_WYKRESY_MIESIC, OnHistoriaWykresyMiesic)
        ON_COMMAND(ID_HISTORIA_WYKRESY_ROK, OnHistoriaWykresyRok)
        ON_COMMAND(ID_PROGNOZA_WYKRESY_DZIE, OnPrognozaWykresyDzie)
        ON_COMMAND(ID_PROGNOZA_WYKRESY_MIESIC, OnPrognozaWykresyMiesic)
        ON_COMMAND(ID_PROGNOZA_WYKRESY_ROK, OnPrognozaWykresyRok)
        //}}AFX_MSG_MAP
        // Standard file based document commands
        ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
        ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
        // Standard print setup command
        ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CLonworksApp construction

CLonworksApp::CLonworksApp()
{
        // TODO: add construction code here,
        // Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CLonworksApp object

CLonworksApp NEAR theApp;

/////////////////////////////////////////////////////////////////////////////
// CLonworksApp initialization

int CLonworksApp::ExitInstance()
{
#ifndef _NOLAN
        ExecuteHaExit();
#endif
        return CWinApp::ExitInstance();
}

BOOL CLonworksApp::InitInstance()
{
        // 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.

        SetDialogBkColor();        // Set dialog background color to gray
        LoadStdProfileSettings();  // Load standard INI file options (including MRU)
        EnableVBX();               // Initialize VBX support

        // Register the application's document templates.  Document templates
        //  serve as the connection between documents, frame windows and views.

        CMultiDocTemplate* pDocTemplate;
```

```
                IDR_CHARTTYPE,
                RUNTIME_CLASS(CMeasDoc),
                RUNTIME_CLASS(CSpliter),
                RUNTIME_CLASS(CMeasTable));
        AddDocTemplate(pDocTemplate);

        pDocTemplate = new CLonDocTemplate( // CMultiDocTemplate(
                IDR_LONWORTYPE,
                RUNTIME_CLASS(CLonworksDoc),
                RUNTIME_CLASS(CMDIChildWnd),
                RUNTIME_CLASS(CLonworksView));
        AddDocTemplate(pDocTemplate);

        if (LoadVBXFile("grid.vbx") <= HINSTANCE_ERROR)
        {
                AfxMessageBox("Cannot load GRID.VBX.\nPlease place the file on the path
\n");
                return FALSE;
        }
        else
        {
                UnloadVBXFile("grid.vbx");
        }

#ifndef _NOLAN

        if (ExecuteHaIni())
        {
                AfxMessageBox("B|d podczas inicjalizacji interfejsu sieci LonWorks.\n",
;
                return FALSE;
        }
#endif

        // create main MDI Frame window
        CMainFrame* pMainFrame = new CMainFrame;
        if (!pMainFrame->LoadFrame(IDR_MAINFRAME))
                return FALSE;
        m_pMainWnd = pMainFrame;

        // enable file manager drag/drop and DDE Execute open
        EnableShellOpen();
        RegisterShellFileTypes();

        // create a new (empty) document
        POSITION pos = AfxGetApp()->m_templateList.GetHeadPosition();
        AfxGetApp()->m_templateList.GetNext(pos);
        ((CDocTemplate *) (AfxGetApp()->m_templateList.GetAt(pos))) -> OpenDocumentFile(
NULL);

        m_pMainWnd->DragAcceptFiles();
        // The main window has been initialized, so show and update it.
        pMainFrame->ShowWindow(m_nCmdShow);
        pMainFrame->UpdateWindow();

        return TRUE;
}

BOOL CLonworksApp::OnIdle( LONG /*lCount*/ )
{
        CChannelMeas p;
        p.dValue = 0;
#ifdef _NOLAN
        static CTime t = CTime::GetCurrentTime();
        CTimeSpan t1sec = CTimeSpan (0,0,0,1+rand() % 3);
```

```
        return TRUE;
        t =  CTime::GetCurrentTime();

        p.nChannelNo = 1 + rand () % 7; // 1..7
        double dNominal;
        if (p.nChannelNo -1 >= TOTAL_CHANNEL)
        {
                ASSERT (p.nChannelNo-6 < 2);
                dNominal = CLonworksDoc::monControl[p.nChannelNo-6].m_dTemp;
                p.dValue = dNominal * 0.5 + rand () % (WORD) dNominal;
        }
    ::SendMessage (HWND_BROADCAST, WM_ADD_LOG, 0, (LPARAM)(char far *)
          "Kolejna bardzo d|uga linia (80 znak≤w) !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!\r\n");
#else

        if (! ExecutePom(&p))
          return TRUE;

#endif

    static CStdioFile measFile ("lonworks.pom", CFile::modeWrite | CFile::modeCreate);

        char buf2[100];
    sprintf (buf2, "Nr kana|u: % d, Warto£µ: %lf\n", unsigned (p.nChannelNo), p.dValue)

    measFile.WriteString (buf2);

        AfxGetMainWnd() -> SendMessageToDescendants (WM_UPDATE_MEAS, 0, (LPARAM) &p);
        return TRUE;
}


/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

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

// Dialog Data
        //{{AFX_DATA(CAboutDlg)
        enum { IDD = IDD_ABOUTBOX };
        //}}AFX_DATA

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

```
BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
        //{{AFX_MSG_MAP(CAboutDlg)
                // No message handlers
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

// App command to run the dialog
void CLonworksApp::OnAppAbout()
{
        CAboutDlg aboutDlg;
        aboutDlg.DoModal();
}

/////////////////////////////////////////////////////////////////////////////
// VB-Event registration
// (calls to AfxRegisterVBEvent will be placed here by ClassWizard)

//{{AFX_VBX_REGISTER_MAP()
//}}AFX_VBX_REGISTER_MAP

/////////////////////////////////////////////////////////////////////////////
// CLonworksApp commands

CDocument* CLonworksApp::OpenDocumentFile( LPCSTR lpszFileName )
{
  CDocument *pDoc = CWinApp::OpenDocumentFile(lpszFileName);
  if (pDoc)
    AfxGetMainWnd() -> SendMessageToDescendants (WM_OPEN_FILE, 0, (LPARAM) lpszFileName)
;
  return pDoc;
}

void CLonworksApp::OnHistoriaWykresyDzie()
{
        // TODO: Add your command handler code here
        CLonDocTemplate::nType = 0;
        OnFileOpen ();
}

void CLonworksApp::OnHistoriaWykresyMiesic()
{
        // TODO: Add your command handler code here
        CLonDocTemplate::nType = 2;
        OnFileOpen ();
}

void CLonworksApp::OnHistoriaWykresyRok()
{
        // TODO: Add your command handler code here
        CLonDocTemplate::nType = 3;
        OnFileOpen ();
}

void CLonworksApp::OnPrognozaWykresyDzie()
{
        // TODO: Add your command handler code here
        CLonDocTemplate::nType = 4;
        OnFileOpen ();
}

void CLonworksApp::OnPrognozaWykresyMiesic()
{
        // TODO: Add your command handler code here
```

```
        OnFileOpen ();
}

void CLonworksApp::OnPrognozaWykresyRok()
{
        // TODO: Add your command handler code here
        CLonDocTemplate::nType = 7;
        OnFileOpen ();
}
```

```
// mainfrm.cpp : implementation of the CMainFrame class
//

#include "stdafx.h"
#include "lonworks.h"
#include "londoc.h"
#include "mainfrm.h"
#include "lonview.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMainFrame

IMPLEMENT_DYNAMIC(CMainFrame, CMDIFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CMDIFrameWnd)
        //{{AFX_MSG_MAP(CMainFrame)
        ON_WM_CREATE()
        ON_WM_ENTERIDLE()
        ON_COMMAND(ID_KONFIGURACJA_EKRANY, OnKonfiguracjaEkrany)
        ON_COMMAND(ID_BILANSE_ENERGIA, OnBilanseEnergia)
        ON_COMMAND(ID_BILANSE_KONFIGURACJA, OnBilanseKonfiguracja)
        ON_COMMAND(ID_PARAMETRY_LIMITY, OnParametryLimity)
        ON_COMMAND(ID_PARAMETRY_TARYFY, OnParametryTaryfy)
        ON_COMMAND(ID_KONFIGURACJA_WYWIETLANIELOGU, OnKonfiguracjaWywietlanielogu)
        ON_UPDATE_COMMAND_UI(ID_KONFIGURACJA_WYWIETLANIELOGU, OnUpdateKonfiguracjaWywie
lanielogu)
        //}}AFX_MSG_MAP
        // Global help commands
        ON_COMMAND(ID_HELP_INDEX, CMDIFrameWnd::OnHelpIndex)
        ON_COMMAND(ID_HELP_USING, CMDIFrameWnd::OnHelpUsing)
        ON_COMMAND(ID_HELP, CMDIFrameWnd::OnHelp)
        ON_COMMAND(ID_CONTEXT_HELP, CMDIFrameWnd::OnContextHelp)
        ON_COMMAND(ID_DEFAULT_HELP, CMDIFrameWnd::OnHelpIndex)
        ON_MESSAGE(WM_ADD_LOG,OnAddLog)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// arrays of IDs used to initialize control bars

// toolbar buttons - IDs are command buttons
static UINT BASED_CODE buttons[] =
{
        // same order as in the bitmap 'toolbar.bmp'
        ID_FILE_NEW,
        ID_FILE_OPEN,
        ID_FILE_SAVE,
                ID_SEPARATOR,
        ID_EDIT_CUT,
        ID_EDIT_COPY,
        ID_EDIT_PASTE,
                ID_SEPARATOR,
        ID_FILE_PRINT,
        ID_APP_ABOUT,
        ID_CONTEXT_HELP,
};

static UINT BASED_CODE indicators[] =
{
        ID_SEPARATOR,           // status line indicator
        ID_INDICATOR_CAPS,
```

```
        ID_INDICATOR NUM,
        ID_INDICATOR SCRL,
};

/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction

CMainFrame::CMainFrame()
{
        // TODO: add member initialization code here
        m_bLog = FALSE;
}

CMainFrame::~CMainFrame()
{
}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
        if (CMDIFrameWnd::OnCreate(lpCreateStruct) == -1)
                return -1;
/*
        if (!m_wndToolBar.Create(this) ||
                !m_wndToolBar.LoadBitmap(IDR_MAINFRAME) ||
                !m_wndToolBar.SetButtons(buttons,
                  sizeof(buttons)/sizeof(UINT)))
        {
                TRACE("Failed to create toolbar\n");
                return -1;      // fail to create
        }
*/
        if (!m_wndStatusBar.Create(this) ||
                !m_wndStatusBar.SetIndicators(indicators,
                  sizeof(indicators)/sizeof(UINT)))
        {
                TRACE("Failed to create status bar\n");
                return -1;      // fail to create
        }

        return 0;
}

/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
        CMDIFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
        CMDIFrameWnd::Dump(dc);
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers

void CMainFrame::OnParentNotify( UINT message, LPARAM lParam)
{
        if (message == WM_DESTROY && HIWORD (lParam) == IDD_MONITORING)
          ((CLonworksDoc *) GetActiveDocument()) -> pMonMeasDlg = NULL;
}
```

```
void CMainFrame::OnEnterIdle( UINT /*nWhy*/, CWnd* /*pWho*/ )
{
        AfxGetApp () -> OnIdle (-1);
}

LRESULT CMainFrame::OnAddLog( WPARAM, LPARAM lParam )

{
  if (m_bLog)
        SendMessageToDescendants(WM_ADD_LOG, 0, lParam);
  return 0L;
}

void CMainFrame::OnKonfiguracjaEkrany()
{
        // TODO: Add your command handler code here
        CString sInfo (
            "Wersja Demonstracyjna programu nie zawiera mo¦liwo£ci\n"
            "programowania ekran≤w.\n"
            "Wersja dla klienta zawiera mo¦liwo£µ programowania\n"
            "kilkunastu ekran≤w z wieloma kana|ami pomiarowymi\n"
            "na ka¦dym z nich."
        );
        AfxMessageBox (sInfo);
}

void CMainFrame::OnBilanseEnergia()
{
        // TODO: Add your command handler code here
        CString sInfo (
            "Wersja Demonstracyjna programu nie zawiera mo¦liwo£ci\n"
            "bilansowania energii.\n"
            "Wersja dla klienta zawiera mo¦liwo£µ bilansowania\n"
            "energii z ka¦dego licznika, sumowanie wskaza± licznik≤w\n"
            "zgodnie z podzia|em na instalacje technologiczne i\n"
            "wΩz|y energetyczne oraz wyliczanie r≤¦nic bilansowych\n"
            "dla poszczeg≤lnych rozdzielni."
        );
        AfxMessageBox (sInfo);
}

void CMainFrame::OnBilanseKonfiguracja()
{
        // TODO: Add your command handler code here
        CString sInfo (
            "Wersja Demonstracyjna programu nie zawiera mo¦liwo£ci\n"
            "bilansowania energii i konfiguracji bilansowania.\n"
            "Wersja dla klienta zawiera mo¦liwo£µ bilansowania\n"
            "energii z ka¦dego licznika, sumowanie wskaza± licznik≤w\n"
            "zgodnie z podzia|em na instalacje technologiczne i\n"
            "wΩz|y energetyczne oraz wyliczanie r≤¦nic bilansowych\n"
            "dla poszczeg≤lnych rozdzielni. W tym punkcie wprowadza\n"
            "siΩ dane do sporz¦dzania bilans≤w."
        );
        AfxMessageBox (sInfo);
}

void CMainFrame::OnParametryLimity()
{
        // TODO: Add your command handler code here
        CString sInfo (
            "Wersja Demonstracyjna programu nie zawiera mo¦liwo£ci\n"
            "wprowadzania taryf, limit≤w  i innych sta|ych parametr≤w.\n"
            "Wersja dla klienta zawiera mo¦liwo£µ wprowadzania\n"
            "dowolnych danych sta|ych do oblicze± i wykres≤w.\n"
```

```
        AfxMessageBox (sInfo);
}

void CMainFrame::OnParametryTaryfy()
{
        // TODO: Add your command handler code here
        CString sInfo (
            "Wersja Demonstracyjna programu nie zawiera mo¦liwo£ci\n"
            "wprowadzania taryf, limit¦w  i innych sta|ych parametr¦w.\n"
            "Wersja dla klienta zawiera mo¦liwo£µ wprowadzania\n"
            "dowolnych danych sta|ych do oblicze± i wykres¦w.\n"
            );
        AfxMessageBox (sInfo);
}

void CMainFrame::OnKonfiguracjaWywietlanielogu()
{
        // TODO: Add your command handler code here
        m_bLog = !m_bLog;
}

void CMainFrame::OnUpdateKonfiguracjaWywietlanielogu(CCmdUI* pCmdUI)
{
        // TODO: Add your command update UI handler code here
        if (m_bLog)
            ,   pCmdUI -> SetCheck (TRUE);
        else
                pCmdUI -> SetCheck (FALSE);
}
```

```
// measdoc.cpp : implementation file
//

#include "stdafx.h"
#include "lonworks.h"
#include "measdoc.h"
#include "chartvw.h"
#include "chanchoi.h"
#include "londoc.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMeasDoc

IMPLEMENT_SERIAL(CMeasDoc, CDocument, 0 /* schema number*/ )

CMeasDoc::CMeasDoc()
{
        m_nChartType = CHART_LINE;
        m_pdData = NULL;
        m_nDataRows = 4;
        m_nDataCols = 12;
        m_nStartCol = 0;
        m_nStartRow = 0;
        m_nChannelNo = 0;
        m_dNominal = CLonworksDoc::channelCfg[m_nChannelNo].m_dNominal;
        m_bChartInPercent = TRUE;
        m_nScale = 0;
}

BOOL CMeasDoc::OnNewDocument()
{
        if (!CDocument::OnNewDocument())
                return FALSE;
        return TRUE;
}

void ClearMapWordToPtr(CMapWordToPtr* pMap)
{
    POSITION pos;
    WORD  key;
    CString *pString;

    if ( pMap->GetCount() )
    {
        pos = pMap->GetStartPosition();
        while ( NULL != pos )
        {
            pMap->GetNextAssoc(pos, key, (void *&) pString);
            delete pString;
        }
        pMap->RemoveAll();
    }
}


CMeasDoc::~CMeasDoc()
{
        ClearMapWordToPtr (&m_mapDsc);
        if (m_pdData)
          delete [] m_pdData;
}
```

```
BEGIN_MESSAGE_MAP(CMeasDoc, CDocument)
        //{{AFX_MSG_MAP(CMeasDoc)
        ON_COMMAND(ID_HISTORIA_KANA, OnHistoriaKanal)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////////
// CMeasDoc serialization

void CMeasDoc::Serialize(LPCSTR lpszFileName)
{
        if (m_pdData)
          return; //delete [] m_pdData;

        ClearMapWordToPtr (&m_mapDsc);

        m_nDataCols = (WORD) GetPrivateProfileInt ("Parametry", "LiczbaProbek", 1, lpszF
ileName);
        if (m_nDataCols > 31)
                m_nDataCols = 31;
        m_nDataRows = (WORD) GetPrivateProfileInt ("Parametry", "LiczbaKanalow", 1, lpsz
FileName);

        m_pdData = new double [m_nDataRows * m_nDataCols];

        double *pdData = m_pdData;

                // Initalize "Cell" Contents
                for (WORD nCol = 1; nCol <= m_nDataCols; nCol++)
                {
                        char szSample[30];
                        sprintf (szSample, "Probka%d", nCol);

            char buf[100];
                        GetPrivateProfileString (szSample, "Opis", "", buf, 80, lpszFile
Name);

                        if (*buf)
                        {
                                CString *psOpis = new CString(buf);
                                m_mapDsc.SetAt (nCol, psOpis);
                        }

                        for (WORD nRow = 1; nRow <= m_nDataRows; nRow++)
                        {
                                char szChannel[30];
                                sprintf (szChannel, "Kanal%d", nRow);
                                char szData [20];
                                if (!GetPrivateProfileString (szSample, szChannel, "0",
szData, 20, lpszFileName))
                                        TRACE2 ("Brak danych dla kana|u %d w pr≤bce %d\n
", nRow, nCol);
                sscanf (szData, "%lf", pdData++);
                        }
                }
}

void CMeasDoc::Serialize(CArchive& ar)
{
        if (ar.IsStoring())
        {
                // TODO: add storing code here
        }
```

```
        else
        {
                // TODO: add loading code here
        }
}

/////////////////////////////////////////////////////////////////////////////
// CMeasDoc commands

void CMeasDoc::OnHistoriaKanal()
{
        // TODO: Add your command handler code here
        CChanChoiceDlg dlg;
        dlg.m_nChannelNo = m_nChannelNo;
        dlg.m_bPercent = m_bChartInPercent;
        dlg.m_nScale = m_nScale;
        if (dlg.DoModal() != IDOK)
          return;
        m_nChannelNo = (WORD) dlg.m_nChannelNo;
        m_bChartInPercent = dlg.m_bPercent;
        m_nScale = dlg.m_nScale;
        m_dNominal = CLonworksDoc::channelCfg[m_nChannelNo].m_dNominal;
        UpdateAllViews (NULL);
}
```

```
// meastabl.cpp : impl
//

#include "stdafx.h"
#include "lonworks.h"
#include "meastabl.h"
#include "measdoc.h"
#include "lonview.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMeasTable

IMPLEMENT_DYNCREATE(CMeasTable, CFormView)

CMeasTable::CMeasTable()
        : CFormView(CMeasTable::IDD)
{
        //{{AFX_DATA_INIT(CMeasTable)
        m_pGrid = NULL;
        //}}AFX_DATA_INIT
}

CMeasTable::~CMeasTable()
{
}

void CMeasTable::OnInitialUpdate()     // first time after construct
{
        CFormView::OnInitialUpdate();       // tie into Document

        HWND hWndGrid = m_pGrid->GetSafeHwnd();
        DWORD dwStyles = ::GetWindowLong(hWndGrid, GWL_STYLE);
        ::SetWindowLong(hWndGrid, GWL_STYLE, dwStyles | WS_CLIPSIBLINGS);

        CMeasDoc* pDoc = (CMeasDoc*) GetDocument();
        ASSERT(pDoc->IsKindOf(RUNTIME_CLASS(CMeasDoc)));
}

#define TWIPS_PER_INCH 1440

void CMeasTable::SizeToFit()
{
        // Do not let control stay scrolled
        m_pGrid->SetNumProperty("LeftCol", m_pGrid->GetNumProperty("FixedCols"));
        m_pGrid->SetNumProperty("TopRow",  m_pGrid->GetNumProperty("FixedRows"));

        CPoint ptSize = FindCellPosition((int)m_pGrid->GetNumProperty("Rows"),
                                                                (int)m_pGrid->G
etNumProperty("Cols"));

        if (ptSize.x != (int)m_pGrid->GetNumProperty("Width") ||
                ptSize.y != (int)m_pGrid->GetNumProperty("Height"))
        {
                CRect rect;
                rect.left   = (int)m_pGrid->GetNumProperty("Left");
                rect.top    = (int)m_pGrid->GetNumProperty("Top");
                rect.right  = rect.left + ptSize.x;
                rect.bottom = rect.top  + ptSize.y;

                m_pGrid->Move(rect);
```

```
                ptSize.x += 2 * rect.left;
                ptSize.y += 2 * rect.top;

                SetScrollSizes(MM_TEXT, (CSize)ptSize);
        }
}

//
// Returns the position of the top-left corner of a cell.  Asking for
// nRow == maxRow and nCol == maxCol will give the extent of the grid
//
CPoint CMeasTable::FindCellPosition(int nRow, int nCol)
{
        ASSERT(nRow >= 0 && nRow <= m_pGrid->GetNumProperty("Rows"));
        ASSERT(nCol >= 0 && nCol <= m_pGrid->GetNumProperty("Cols"));

        CPoint ptPos(0, 0);

        // Get left edge of requested cell by summing widths of previous cells
        int nLeftCol = (int)m_pGrid->GetNumProperty("LeftCol");
        int nLeftFix = (int)m_pGrid->GetNumProperty("FixedCols");

        for (int i = 0; i < nCol; i++)
        {
                if (i < nLeftFix || i >= nLeftCol)  // only count displayed cells
                        ptPos.x += (int)m_pGrid->GetNumProperty("ColWidth", i);
        }

        // Get top edge of requested cell by summing Heights of previous cells
        int nTopRow = (int)m_pGrid->GetNumProperty("TopRow");
        int nTopFix = (int)m_pGrid->GetNumProperty("FixedRows");
        for (i = 0; i < nRow; i++)
        {
                if (i < nTopFix || i >= nTopRow)    // only count displayed cells
                        ptPos.y += (int)m_pGrid->GetNumProperty("RowHeight", i);
        }

        // ptPos is now in LOGICAL TWIPs
        // Convert to pixels

        CClientDC dc(this);

        ptPos.x = MulDiv(ptPos.x, dc.GetDeviceCaps(LOGPIXELSX), TWIPS_PER_INCH);
        ptPos.y = MulDiv(ptPos.y, dc.GetDeviceCaps(LOGPIXELSY), TWIPS_PER_INCH);

        ptPos.x += nCol;        // add one pixel per column for gap
        ptPos.y += nRow;        // add one pixel per row    for gap

        return ptPos;
}

void CMeasTable::DoDataExchange(CDataExchange* pDX)
{
        CFormView::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CMeasTable)
        DDX_VBControl(pDX, IDC_GRID, m_pGrid);
        //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CMeasTable, CFormView)
        //{{AFX_MSG_MAP(CMeasTable)
                // NOTE - the ClassWizard will add and remove mapping macros here.
        ON_MESSAGE(WM_OPEN_FILE, OnOpenFile)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

```
////////////////////////////////////////////////////////////////////////////
// CMeasTable message handlers

LRESULT CMeasTable::OnOpenFile (WPARAM, LPARAM fileName)
{
        CMeasDoc* pDoc = (CMeasDoc* ) GetDocument();
        pDoc -> Serialize ((LPCSTR) fileName);

                WORD nRows = pDoc -> m_nDataRows + 1;
                m_pGrid->SetNumProperty("Rows", nRows);
                WORD nCols = pDoc -> m_nDataCols + 1;
                m_pGrid->SetNumProperty("Cols", nCols);
                char buf[80];

                // Initialize row headings
                m_pGrid->SetNumProperty("Col", 0);
                for (WORD nRow = 1; nRow < nRows; nRow++)
                {
                        sprintf(buf, "Kana| %d", nRow);
                        m_pGrid->SetNumProperty("Row", nRow);
                        m_pGrid->SetStrProperty("Text", buf);
                }

                // Initialize column headings
                m_pGrid->SetNumProperty("Row", 0);
                for (WORD nCol = 1; nCol < nCols; nCol++)
                {
                        CString *psOpis;
                        if (pDoc->m_mapDsc.Lookup (nCol, (void*&) psOpis))
                                strncpy (buf, *psOpis, 80);
                        else
                                sprintf(buf, "Pr≤bka %d", nCol);
                        m_pGrid->SetNumProperty("Col", nCol);
                        m_pGrid->SetStrProperty("Text", buf);
                }

                // Initalize Cell Contents
                double *pdData = pDoc->m_pdData;

                m_pGrid->SetNumProperty("ColWidth", 1000, 0);

                for (nCol = 1; nCol < nCols; nCol++)
                {
                        m_pGrid->SetNumProperty("Col", nCol);
                        m_pGrid->SetNumProperty("ColWidth", 1000, nCol);
                        m_pGrid->SetNumProperty("ColAlignment", 1, nCol);       // right
, bo liczby
                        m_pGrid->SetNumProperty("FixedAlignment", 2, nCol); // center, b
o tytu|
                        for (nRow = 1; nRow < nRows; nRow++)
                        {
                                m_pGrid->SetNumProperty("Row", nRow);
                                sprintf(buf, "%.2lf", *(pdData++));
                                m_pGrid->SetStrProperty("Text", buf);
                        }
                }
                m_pGrid->SetNumProperty("Row", 1);
                m_pGrid->SetNumProperty("Col", 1);      // force edit to reposition

        SizeToFit();

        return 0L;
}
```

```
// monctrld.cpp : implementation file

//

#include "stdafx.h"
#include "lonworks.h"
#include "londoc.h"
#include "monctrld.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMonCtrlDlg dialog


CMonCtrlDlg::CMonCtrlDlg(BOOL *pbEnable, CMonControl *pMonControl, CMonControl *pMonInf
)
        : CDialog(CMonCtrlDlg::IDD, NULL)
{
        m_pbEnable = pbEnable;
        m_pMonControl = pMonControl;
        m_pMonInfo = pMonInfo;

        //{{AFX_DATA_INIT(CMonCtrlDlg)
        m_nMaxTime1 = pMonControl[0].m_nTime;
        m_nMaxTime2 = pMonControl[1].m_nTime;
        m_dTaskTemp1 = pMonControl[0].m_dTemp;
        m_dTaskTemp2 = pMonControl[1].m_dTemp;
        //}}AFX_DATA_INIT

//      PrepareData ();
}

void CMonCtrlDlg::PrepareData (void)
{
        char buf[20];

        sprintf (buf, "%lu", m_pMonInfo[0].m_nTime);
        m_sTime1 = buf;
        GetDlgItem (IDC_TIME1) -> SetWindowText (buf);

        sprintf (buf, "%lu", m_pMonInfo[1].m_nTime);
        m_sTime2 = buf;
```

```
        sprintf (buf, "%5.1lf", m_pMonInfo[0].m_dTemp);
        m_sTemp1 = buf;
        GetDlgItem (IDC_TEMP1) -> SetWindowText (buf);

        sprintf (buf, "%5.1lf", m_pMonInfo[1].m_dTemp);
        m_sTemp2 = buf;
        GetDlgItem (IDC_TEMP2) -> SetWindowText (buf);
}

void CMonCtrlDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CMonCtrlDlg)
        DDX_Text(pDX, IDC_MAXTIME1, m_nMaxTime1);
        DDV_MinMaxDWord(pDX, m_nMaxTime1, 1, 100000);
        DDX_Text(pDX, IDC_MAXTIME2, m_nMaxTime2);
        DDV_MinMaxDWord(pDX, m_nMaxTime2, 1, 100000);
        DDX_Text(pDX, IDC_TASK_TEMP1, m_dTaskTemp1);
        DDX_Text(pDX, IDC_TASK_TEMP2, m_dTaskTemp2);
        DDX_Text(pDX, IDC_TEMP1, m_sTemp1);




        DDX_Text(pDX, IDC_TEMP2, m_sTemp2);
        DDX_Text(pDX, IDC_TIME1, m_sTime1);
        DDX_Text(pDX, IDC_TIME2, m_sTime2);
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CMonCtrlDlg, CDialog)
        //{{AFX_MSG_MAP(CMonCtrlDlg)
        ON_WM_CTLCOLOR()
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////////
// CMonCtrlDlg message handlers

HBRUSH CMonCtrlDlg::OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor)
{
        // TODO: Add your message handler code here and/or call default

        HBRUSH brush =  CDialog::OnCtlColor(pDC, pWnd, nCtlColor);

        // sprawdf, czy jest to static dla pieca za|czonego

        if (nCtlColor == CTLCOLOR_STATIC)
          {
              if ( (m_pbEnable[0] && ( GetDlgItem (IDC_TIME1) -> m_hWnd == pWnd -> m_hWn
d ||
                                                GetDlgItem (IDC_TEMP1) -> m_hWnd ==
Wnd -> m_hWnd ))
                                              ||
                             (m_pbEnable[1] && ( GetDlgItem (IDC_TIME2) -> m_hWnd == pWnd
-> m_hWnd ||
                                                GetDlgItem (IDC_TEMP2) -> m_hWnd == p
Wnd -> m_hWnd )))
                              pDC -> SetTextColor ( RGB (0xff, 0, 0));
          }
        return brush;
}


void CMonCtrlDlg::OnOK()
{
        // TODO: Add extra validation here
```

```
        m_pMonControl[0].m_nTime = m_nMaxTime1;
        m_pMonControl[0].m_dTemp = m_dTaskTemp1;

        m_pMonControl[1].m_nTime = m_nMaxTime2;
        m_pMonControl[1].m_dTemp = m_dTaskTemp2;

        CDialog::OnOK();
}
```

```
// monmeasd.cpp : impl
//

#include "stdafx.h"
#include "lonworks.h"
#include "londoc.h"
#include "monmeasd.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED CODE THIS_FILE[] =  FILE ;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMonMeasDlg dialog

WORD names[] = { IDC_CHAN1_NAME, IDC_CHAN2_NAME, IDC_CHAN3_NAME,
                                 IDC_CHAN4_NAME, IDC_CHAN5_NAME };

WORD values[] = { IDC_CHAN1_VAL1, IDC_CHAN2_VAL1, IDC_CHAN3_VAL1,
                                 IDC_CHAN4_VAL1, IDC_CHAN5_VAL1 };

WORD units[] = { IDC_CHAN1_UNIT1, IDC_CHAN2_UNIT1, IDC_CHAN3_UNIT1,
                                 IDC_CHAN4_UNIT1, IDC_CHAN5_UNIT1 };

/*const char *pszKindOfChannel[] =
        { "nieaktywny",
          "mikrowolty",
          "milivolty",
          "miliampery",
          "przetwornik 4-20 mA",
          "termoelement typ K"
    };*/

const char *pszUnitOfChannel[] =
        { "kV","kV/h",
          "V", "V/h",
          "mV","mV/h",
          "uV","uV/h",
          "kA","kA/h",
          "A", "A/h",
          "mA","mA/h",
          "VA","VA/h",
          "GJ","MW",
          "m3", "m3/h",
          "kWh", "kW"
    };

/*const char *pszConvOfChannel[] =
        { "bez konwersji",
          "y=ax3 + bx2 + cx + d"
    };*/

CMonMeasDlg::CMonMeasDlg(CChannelCfg *pChannelCfg, double *pdMeasVal,
                                                BOOL *bFocused, CMonMeasDlg **ppThis)
        : CDialog(CMonMeasDlg::IDD, NULL)
{
        //{{AFX_DATA_INIT(CMonMeasDlg)
                // NOTE: the ClassWizard will add member initialization here
        //}}AFX_DATA_INIT

        m_pChannelCfg = pChannelCfg;
        m_pdMeasVal = pdMeasVal;
        m_bFocused = bFocused;
        m_ppThis = ppThis;
```

```
        Create (CMonMeasDlg, --);
}

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

        for (int i = 0; i < TOTAL_CHANNEL; i++)
          {
                GetDlgItem (names[i]) -> SetWindowText ((m_pChannelCfg+i) -> m_pszName),
                GetDlgItem (units[i]) -> SetWindowText (pszUnitOfChannel[1+2*(m_pChannel
Cfg+i) -> m_nUnit]);
                GetDlgItem (units[i]+1) -> SetWindowText (pszUnitOfChannel[1+2*(m_pChan.
elCfg+i) -> m_nUnit]);
                GetDlgItem (units[i]+2) -> SetWindowText (pszUnitOfChannel[2*(m_pChanne'
Cfg+i) -> m_nUnit]);

                for (int j=0; j < 3; j++)
                  {
                    char val[10];
                        sprintf (val, "%7.3lf", m_pdMeasVal[i*3+j]);
                        GetDlgItem (values[i] + j) -> SetWindowText (val);
                  }
          }
}

void CMonMeasDlg::PostNcDestroy( )
{
        *m_ppThis = NULL;

        delete this;
}

BEGIN_MESSAGE_MAP(CMonMeasDlg, CDialog)
        //{{AFX_MSG_MAP(CMonMeasDlg)
        ON_WM_CTLCOLOR()
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////////
// CMonMeasDlg message handlers

void CMonMeasDlg::OnCancel()
{
        // TODO: Add extra cleanup here
        DestroyWindow ();
}

HBRUSH CMonMeasDlg::OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor)
{
        // TODO: Add your message handler code here and/or call default

        HBRUSH brush =  CDialog::OnCtlColor(pDC, pWnd, nCtlColor);

        // sprawdf, czy jest to static z wartoŁciami pomiarśw

        if (nCtlColor == CTLCOLOR_STATIC)
          {
            for (int i=0; i < TOTAL_CHANNEL; i++)
              if (GetDlgItem (values[i]) -> m_hWnd == pWnd -> m_hWnd ||
                        GetDlgItem (values[i]+1) -> m_hWnd == pWnd -> m_hWnd ||
```

```
                    GetDlgItem (values[i]+2) -> m_hWnd == pWnd -> m_hWnd)

                {
                  if (m_bFocused[i])
                    pDC -> SetTextColor ( RGB (0xff, 0, 0));
                  break;
                }
          }
        return brush;
}
```

```
// spliter.cpp : impl
//

#include "stdafx.h"
#include "lonworks.h"
#include "spliter.h"
#include "chartvw.h"
#include "meastabl.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CSpliter

IMPLEMENT_DYNCREATE(CSpliter, CMDIChildWnd)

CSpliter::CSpliter()
{
}

CSpliter::~CSpliter()
{
}

BOOL CSpliter::OnCreateClient(LPCREATESTRUCT /*lpcs*/, CCreateContext* pContext)
{
/*
        // create a splitter with 2 rows, 1 column
        if (!m_wndSplitter.CreateStatic(this, 2, 1))
        {
                TRACE("Failed to CreateStaticSplitter\n");
                return FALSE;
        }

        // add the first splitter pane - the default view in row 0
    // default view is LonworksView
        if (!m_wndSplitter.CreateView(0, 0,
                pContext->m_pNewViewClass, CSize(600, 100), pContext))
        {
                TRACE("Failed to create first pane\n");
                return FALSE;
        }

        // add the second splitter pane - ChartView in row 1
        if (!m_wndSplitter.CreateView(1, 0,
                RUNTIME_CLASS(CChartView), CSize(600, 400), pContext))
        {
                TRACE("Failed to create second pane\n");
                return FALSE;
        }

        // activate the first view
        SetActiveView((CView*)m_wndSplitter.GetPane(0,0));

        return TRUE;
*/
        CRect rect;
        GetClientRect(&rect);

        CSize size = rect.Size();
        size.cy /= 2;

        BOOL bSuccess;
```

```
        [illegible] = m_wndSplitter.CreateStatic(this, 2, 1, WS_CHILD|WS_VISIBLE, AFX_IDW
PANE_FIRST);
        bSuccess &= m_wndSplitter.CreateView(0, 0, RUNTIME_CLASS(CMeasTable), size, pCon
text);
        bSuccess &= m_wndSplitter.CreateView(1, 0, RUNTIME_CLASS(CChartView), size, pCon
text);

        return TRUE;


}

BEGIN_MESSAGE_MAP(CSpliter, CMDIChildWnd)
        //{{AFX_MSG_MAP(CSpliter)
        ON_COMMAND(ID_FILE_PRINT, OnFilePrint)
        ON_COMMAND(ID_FILE_PRINT_PREVIEW, OnFilePrintPreview)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////////
// CSpliter message handlers

void CSpliter::OnFilePrint()
{
        ((CChartView*)(m_wndSplitter.GetPane(1, 0)))->Print();   // Tell Chart to print
}

void CSpliter::OnFilePrintPreview()
{
        ((CChartView*)(m_wndSplitter.GetPane(1, 0)))->Preview();   // Tell Chart to pre
iew
}
```

```
// stdafx.cpp : source file that includes just the standard incl
//  stdafx.pch will be the pre-compiled header
//  stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"
// validate.cpp  : implementation file

#include "stdafx.h"
#include "validate.h"
#include "resource.h"

// validete minimum string lenght
void AFXAPI DDV_MinChars(CDataExchange* pDX, CString const& value, int nChars)
{
    ASSERT(nChars >= 0);        // allow them something
    if (pDX->m_bSaveAndValidate && value.GetLength() < nChars)
    {
        char szT[32];
        wsprintf(szT, "%d", nChars);
        CString prompt;
        AfxFormatString1(prompt, AFX_MIN_STRING_SIZE, szT);
        AfxMessageBox(prompt, MB_ICONEXCLAMATION);
        prompt.Empty(); // exception prep
        pDX->Fail();
    }
}
```

```
// chancfg.h : header
//

/////////////////////////////////////////////////////////////////////////////
// CChannelCfgDlg dialog

class CChannelCfgDlg : public CDialog
{
// Construction
public:
        CChannelCfgDlg(CChannelCfg *pChannelCfg);

        CChannelCfg *m_pChannelCfg;

// Dialog Data
        //{{AFX_DATA(CChannelCfgDlg)
        enum { IDD = IDD_CHANNEL_DLG };
        CComboBox       m_cbUnit;
        CComboBox       m_cbKindConv;
        CComboBox       m_cbConv;
        double  m_dCoeff;
        double  m_dDisp;
        double  m_dHiAlarm;
        double  m_dLowAlarm;
        double  m_dUnit;
        CString m_sName;
        double  m_dNomin;
        //}}AFX_DATA
        int             m_nChannelNo;

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

        // Generated message map functions
        //{{AFX_MSG(CChannelCfgDlg)
        virtual BOOL OnInitDialog();
        afx_msg void OnNext();
        afx_msg void OnPrev();
        virtual void OnOK();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
```

```
// chanchoi.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CChanChoiceDlg dialog

class CChanChoiceDlg : public CDialog
{
// Construction
public:
        CChanChoiceDlg(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
        //{{AFX_DATA(CChanChoiceDlg)
        enum { IDD = IDD_CHAN_CHOICE };
        int             m_nChannelNo;
        BOOL    m_bPercent;
        int             m_nScale;
        //}}AFX_DATA

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

        // Generated message map functions
        //{{AFX_MSG(CChanChoiceDlg)
                // NOTE: the ClassWizard will add member functions here
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};


// chartvw.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CChartView view

class CChartView : public CScrollView
{
        DECLARE_DYNCREATE(CChartView)
protected:
        CChartView();                   // protected constructor used by dynamic creatio
n

// Attributes
public:
        int m_nChannelNo; // nr kana|u do wy£wietlenia
// Operations
public:
        void Print();
        void Preview();

// Implementation
protected:
        virtual ~CChartView();
        virtual void OnDraw(CDC* pDC);          // overridden to draw this view
        virtual void OnInitialUpdate();         // first time after construct
        virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);

        // Generated message map functions
        //{{AFX_MSG(CChartView)
        afx_msg LRESULT OnOpenFile (WPARAM, LPARAM);
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()

// chart support
```

```
        double GetMaxValue();
        int GetTickValue(double dMaxValue, int nNumDiv);
        int GetChartMetrics(CDC* pDC, int nHDiv, int nVDiv,
                CSize& sizeGraph, CSize& sizePixelsPerTick, CPoint& ptOrigin);
        void PlotCaptions(CDC* pDC, CPoint ptOrigin,
                        CSize sizePixelsPerTick, int nValuePerTick,
                        int nNumTicks, int nCharHeight, BOOL bVert, BOOL bAlpha);
        void PlotAxes(CDC* pDC, CPoint ptOrigin, CSize sizePixelsPerTick,
                        int nHorzTicks, int nVertTicks, int nDrawTicks);
        CRect GetDisplayRect(CDC* pDC);
};

#define CHART_LINE      0
#define CHART_VBAR      1
#define CHART_HBAR      2
#define CHART_3DVBAR    3
#define CHART_3DHBAR    4
#define CHART_VGANTT    5
#define CHART_HGANTT    6

#define TICKS_NONE  0
#define TICKS_VERT  1
#define TICKS_HORZ  2
#define TICKS_BOTH  3

////////////////////////////////////////////////////////////////////////////
```

```
/* DISPLAY.H -- include file for Gizmo-2 seven segment display
 * Copyright (c) 1993 by Echelon Corporation.
 * All Rights Reserved.
 *
 * Date last modified: 3/25/93
 */

// This NEURON C #include file contains code to drive the Motorola MC14489
// seven-segment display controller chip, interfaced to the NEURON CHIP
// using Neurowire output mode.

// Pin IO_8 is the Neurowire clock, pin IO_9 is the serial output data
// Pin IO_2 is the chip select (may be modified).

// Change the following #defines if there are fewer than 4 digits
// in the display.  The rightmost digit is numbered 0.

#define NUM_DIGITS     4
#define MAX_NUMBER  9999
#define MIN_NUMBER  -999

////////////////////// DECLARATIONS ////////////////////////////////

IO_8 neurowire master select( IO_2 ) IO_seven_seg;
IO_2 output bit IO_7_seg_select = 1;    // Initially unselected

unsigned char display_reg[ 3 ];         // 24 bits for 7-seg display reg
```

```
unsigned char display_reg_copy[ 3 ];
unsigned char config_reg_copy;


//////////////////// CONTROL DISPLAY HARDWARE ////////////////////

void shift_out( void ) {        // update device hardware registers
    memcpy( display_reg_copy, display_reg, 3 );
    config_reg_copy = config_reg;
    io_out( IO_seven_seg, &config_reg_copy, 8 );
    io_out( IO_seven_seg, display_reg_copy, 24 );









}

void clear_display( void ) {    // clear image of device registers
    display_reg[ 0 ] = 0x80;    // max brightness
    display_reg[ 1 ] = 0;       // blanks on all digits
    display_reg[ 2 ] = 0;
    config_reg = 0xFF;          // special decode on banks 1-5, normal mode
}

////////////////// SINGLE-CHARACTER DISPLAY FUNCTIONS //////////////

void display_decimal( int digit_number, int decimal ) {
        // display decimal number ( 0 - 9 ) on a digit

    display_reg[ 2 - digit_number / 2 ] |=
        ( digit_number & 1 ) ? ( decimal << 4 ) : decimal;
    config_reg &= ~( 1 << ( digit_number + 1 ) );
}

void display_char( int digit_number, int character, boolean is_numeric ) {
          // display a character from the Gizmo character set

    display_reg[ 2 - digit_number / 2 ] |=
        ( digit_number & 1 ) ? ( character << 4 ) : character;
    if( is_numeric ) config_reg &= ~( 1 << ( digit_number + 1 ) );
}

void display_DP( int digit_number ) {
                                        // display decimal point on a digit
    display_reg[ 0 ] |= ( digit_number + 1 ) << 4;
}

void display_minus( int digit_number ) {
                                       // display minus sign on a digit
    display_reg[ 2 - digit_number / 2 ] |=
        ( digit_number & 1 ) ? 0xD0 : 0x0D;
}

////////////////////////// Blank the display ////////////////////////

void blank_display (void) {
    clear_display();
    shift_out();
}
////////////////////////// DISPLAY DECIMAL NUMBER //////////////////

void display_number( long number, int dp_digit );

#define NO_DP   -1      /* display number without  decimal point  */
#define ALL_DPS -2      /* display number with all decimal points */

void display_number( long number, int dp_digit )
{
```

```
    clear_display( );       // clear image of display registers

    if( ( number > MAX_NUMBER ) || ( number < MIN_NUMBER ) ) {
        display_reg[ 1 ] = display_reg[ 2 ] = 0xDD;
        shift_out( );       // update hardware
        return;             // display "----" for overrange
    }
    if( number < 0 ) {
        display_minus( NUM_DIGITS - 1 );        // leading minus sign
        local_num = - number;




    } else local_num = number;

    for( digit = 0;
                local_num || ( digit <= max( dp_digit, 0 ) ); digit++ ) {
                // convert binary to decimal with leading zero suppress
        if( ( number < 0 ) && ( digit == NUM_DIGITS - 1 ) ) break;
        display_decimal( digit, ( int )( local_num % 10 ) );
        local_num /= 10;
    }
    display_DP( dp_digit );  // display decimal point
    shift_out( ) ;           // update hardware
}

////////////////////////////////////////////////////////////////////

typedef struct {
     unsigned character;
     boolean  is_numeric;
     char     ascii;
} char_table_t;                 // Gizmo character set table

const char_table_t char_table[ ] = {
    0xA, TRUE,      'A',
    0xC, TRUE,      'C',
    0xE, TRUE,      'E',
    0xF, TRUE,      'F',
    0x2, FALSE,     'H',
    0x1, TRUE,      'I',
    0x4, FALSE,     'J',
    0x5, FALSE,     'L',
    0x0, TRUE,      'O',
    0x8, FALSE,     'P',
    0x5, TRUE,      'S',
    0xA, FALSE,     'U',
    0xC, FALSE,     'Y',
    0x2, TRUE,      'Z',
    0xB, TRUE,      'b',
    0x1, FALSE,     'c',
    0xD, TRUE,      'd',
    0x3, FALSE,     'h',
    0x1, TRUE,      'l',
    0x6, FALSE,     'n',
    0x7, FALSE,     'o',
    0x9, FALSE,     'r',
    0xB, FALSE,     'u',
    0xD, FALSE,     '-',
    0xE, FALSE,     '=',
    0xF, FALSE,     '%',    // degrees
    0x0, FALSE,     ' ',
    0x0, TRUE,      '0',
    0x1, TRUE,      '1',
    0x2, TRUE,      '2',
    0x3, TRUE,      '3',
```

```
    0x6, TRUE,      '6',
    0x7, TRUE,      '7',
    0x8, TRUE,      '8',
    0x9, TRUE,      '9',
    0xFF };

//////////////////////////// DISPLAY ASCII STRING ///////////////////////

void display_string( const char * s, int dp_digit ) {
    // displays first 4 characters of a string with optional decimal point




    const char_table t * table_ptr;
    int digit;

    clear_display( );          // all non-displayable chars show as blank

    for( digit = NUM_DIGITS - 1; digit >= 0; digit--, s++ ) {

        for( table_ptr = char_table; table_ptr->character != 0xFF;
            table_ptr++ ) {

            if( * s == table_ptr->ascii ) {       // match char in table
                display_char( digit, table_ptr->character,
                    table_ptr->is_numeric );
                break;
            }
        }
    }
    display_DP( dp_digit );  // display decimal point
    shift_out( );            // update hardware
}
```

```
/* HAUIF.H -- Example host application user interface function prototypes.
 * Copyright (c) 1993 by Echelon Corporation.
 * All Rights Reserved.
 *
 * Date last modified: 3/25/93
 */

extern int get_integer( const char * prompt, int default_val );

extern boolean get_choice( const char * prompt, char true_choice,
        char false_choice );

extern boolean handle_error( NI_Code ni_error,
             ComplType completion,      // set to MSG_SUCCEEDS to omit check
             byte      response_code,   // set to NO_CHECK to omit check
             const char    * msg_name );
        // return TRUE if failure, FALSE if success

#define NO_CHECK 0x20   /* do not check response code */

extern void print_array( const void * data, int length, const char * format );

extern boolean report_flag, verbose_flag;
```

```
/* LDVINTFC.H -- LONWORKS standard network interface functions.
 * Copyright (c) 1993 by Echelon Corporation.
 * All Rights Reserved.
 *
 * Date last modified: 10/7/93
 *
 *
 ****************************************************************************
 * Network driver error codes.
 ****************************************************************************
 */

typedef enum {
    LDV_OK = 0,
    LDV_NOT_FOUND,
    LDV_ALREADY_OPEN,
    LDV_NOT_OPEN,
    LDV_DEVICE_ERR,
    LDV_INVALID_DEVICE_ID,
    LDV_NO_MSG_AVAIL,
    LDV_NO_BUFF_AVAIL,




    LDV_DEVICE_BUSY
} LDVCode;

/*
 ****************************************************************************
 * Standard network driver functions.
 ****************************************************************************
 */

typedef int LNI;          // device handle

LNI ldv_open( const char * device_name );
            /* Open network interface device - returns device handle */

LDVCode ldv_read( LNI handle, void * msg_p, unsigned length );
            /* Read a buffer */

LDVCode ldv_write( LNI handle, void * msg_p, unsigned length );
            /* Write a buffer */

LDVCode ldv_close( LNI handle );
           /* Close network interface device */
```

```
// londoc.h : interface of the CLonworksDoc
//
/////////////////////////////////////////////////////////////////////////////
#define MAX_CHANNEL_NAME 30
#define TOTAL_CHANNEL 5

typedef struct
{
  public:
        char    m_pszName[MAX_CHANNEL_NAME];
        int             m_nUnit;
        double  m_dNominal;
        int             m_nKindConv;
        int             m_nConv;
        double  m_dCoeff;
        double  m_dDisp;
        double  m_dHiAlarm;
        double  m_dLowAlarm;
        double  m_dUnit;
} CChannelCfg;

typedef struct
{
        DWORD   m_nTime;
        double  m_dTemp;
} CMonControl;

typedef struct
{

        double dValue;
        char nChannelNo;   //tg
} CChannelMeas;

class CMonMeasDlg;
class CMonCtrlDlg;

class CLonworksDoc : public CDocument
{
protected: // create from serialization only
        CLonworksDoc();
        DECLARE_DYNCREATE(CLonworksDoc)

// Attributes




public:
        static CChannelCfg channelCfg[TOTAL_CHANNEL+1];
        static CMonControl monControl[2]; // max. czas, temp. zadania
        static CMonControl monInfo[2];    // czas, temp.
        CMonMeasDlg *pMonMeasDlg;
        CMonCtrlDlg *pMonCtrlDlg;

private:
        double dMeasVal[TOTAL_CHANNEL][3]; // wartoŁci pomiarśw
        BOOL bFocused[TOTAL_CHANNEL];      // czy dany wiersz z wynikami ma byµ wyrśŜnio
ny (np. na czerwono)
        BOOL bEnable[2];                                    // czy dany piec jest w|ączo.
y
    CTime zegar_old[TOTAL_CHANNEL];
        BOOL bChannelInit[TOTAL_CHANNEL];  // czy czas dla kana|św zainicjowany
```

```
        double dMax[TOTAL_CHANNELS][3];     // wartoœci maksymalne dla poszczególnych ka
a³ów
                                                                        // dla bie¿¹c
ej godziny, dnia, miesi¹ca
        CTime timeLog;                                          // czas ostatniego logu
        CString m_sLogDir;                                      // katalog, w którym bêd¹ umi
eszczane logi
protected:
        void UpdateLog ();
        void WriteLog(BYTE nLogType, const CTime &currentTime);
// Operations
public:
        void UpdateMeas(CChannelMeas *pChanMeas);

// Implementation
public:
        virtual ~CLonworksDoc();
        virtual void Serialize(CArchive& ar);   // overridden for document i/o
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif

protected:
        virtual BOOL OnNewDocument();
        virtual BOOL CanCloseFrame( CFrameWnd* pFrame );

// Generated message map functions
protected:
        //{{AFX_MSG(CLonworksDoc)
        afx_msg void OnOptionsChannels();
        afx_msg void OnMonitorMonitorowanie();
        afx_msg void OnUpdateMonitorMonitorowanie(CCmdUI* pCmdUI);
        afx_msg void OnMonitorSterowanie();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
```

```
// lonview.h : interface of the CLonworksView class
//
/////////////////////////////////////////////////////////////////////////////
#define WM_UPDATE_MEAS (WM_USER + 140)
#define WM_ADD_LOG (WM_USER + 143)
#define WM_OPEN_FILE (WM_USER + 145)

class CLonworksDoc;

class CLonworksView : public CEditView
{




        BOOL m_bFull;
protected: // create from serialization only
        CLonworksView();
        DECLARE_DYNCREATE(CLonworksView)

// Attributes
public:
        CLonworksDoc* GetDocument();

// Operations
public:

// Implementation
public:
        virtual ~CLonworksView();
        virtual void OnDraw(CDC* pDC);  // overridden to draw this view
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif

protected:
        void OnInitialUpdate();

        // Printing support
        virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
        virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
        virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);

// Generated message map functions
protected:
        //{{AFX_MSG(CLonworksView)
                // NOTE - the ClassWizard will add and remove member functions here.
        afx_msg LONG OnUpdateMeas( UINT, LONG );
                //    DO NOT EDIT what you see in these blocks of generated code !
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
        LRESULT OnAddLog( WPARAM wParam, LPARAM lParam );

        int OnCreate(LPCREATESTRUCT lpCreateStruct);
        BOOL CopyToClipboard (char far * lpszTextToCopy);
  private:
        CFont m_font;
};

#ifndef _DEBUG  // debug version in lonview.cpp
inline CLonworksDoc* CLonworksView::GetDocument()
```

```
#endif

////////////////////////////////////////////////////////////////////////////////
```

```
// lonworks.h : main header file for the LONWORKS application
//

#ifndef __AFXWIN_H__
        #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols

/////////////////////////////////////////////////////////////////////////////
// CLonworksApp:
// See lonworks.cpp for the implementation of this class
//

class CLonworksApp : public CWinApp









{
public:
        CLonworksApp();

// Overrides
        virtual BOOL InitInstance();
        virtual int ExitInstance();
        virtual BOOL OnIdle( LONG lCount );

// Implementation
        virtual CDocument* OpenDocumentFile( LPCSTR lpszFileName );

        //{{AFX_MSG(CLonworksApp)
        afx_msg void OnAppAbout();
        afx_msg void OnHistoriaWykresyDzie();
        afx_msg void OnHistoriaWykresyMiesic();
        afx_msg void OnHistoriaWykresyRok();
        afx_msg void OnHistoriaWykresyTydzie();
        afx_msg void OnPrognozaWykresyDzie();
        afx_msg void OnPrognozaWykresyMiesic();
        afx_msg void OnPrognozaWykresyRok();
        afx_msg void OnPrognozaWykresyTydzie();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
// VB-Event extern declarations

//{{AFX_VBX_REGISTER()
//}}AFX_VBX_REGISTER

/////////////////////////////////////////////////////////////////////////////
```

```
// mainfrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////

class CMainFrame : public CMDIFrameWnd
{
        DECLARE_DYNAMIC(CMainFrame)
public:
        CMainFrame();

// Attributes
public:

// Operations
public:

// Implementation
public:
        virtual ~CMainFrame();

        virtual void OnParentNotify( UINT message, LPARAM lParam);
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif

protected:  // control bar embedded members
        CStatusBar  m_wndStatusBar;
        CToolBar    m_wndToolBar;
        BOOL            m_bLog;

// Generated message map functions
protected:




        //{{AFX_MSG(CMainFrame)
        afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
        afx_msg void OnEnterIdle( UINT nWhy, CWnd* pWho );
        afx_msg void OnKonfiguracjaEkrany();
        afx_msg void OnBilanseEnergia();
        afx_msg void OnBilanseKonfiguracja();
        afx_msg void OnParametryLimity();
        afx_msg void OnParametryTaryfy();
        afx_msg void OnKonfiguracjaWywietlanielogu();
        afx_msg void OnUpdateKonfiguracjaWywietlanielogu(CCmdUI* pCmdUI);
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
        LRESULT OnAddLog( WPARAM wParam, LPARAM lParam );

};

/////////////////////////////////////////////////////////////////////////////
```

```
// measdoc.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CMeasDoc document

class CChartView;

class CMeasDoc : public CDocument
{
        DECLARE_SERIAL(CMeasDoc)
protected:
        CMeasDoc();                     // protected constructor used by dynamic creatio
n

// Attributes
public:
        CMapWordToPtr m_mapDsc;                  // opis prśbek z wykresu
// dane i parametry wykresśw
        double* m_pdData;
        WORD m_nDataRows;       // kana|y
        WORD m_nDataCols;       // prśbki
        WORD m_nStartRow;
        WORD m_nStartCol;
        int  m_nChartType;
        WORD m_nChannelNo;
        BOOL m_bChartInPercent; // czy wykresy w procentach
        int  m_nScale;                  // dla wykresśw w procentach wart. max. skali
        double m_dNominal;

// Operations
public:
        void Serialize(LPCSTR lpszFileName);
// Implementation
protected:
        virtual ~CMeasDoc();
        virtual void Serialize(CArchive& ar);   // overridden for document i/o
        virtual BOOL OnNewDocument();

        // Generated message map functions
protected:
        //{{AFX_MSG(CMeasDoc)
        afx_msg void OnHistoriaKanal();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
friend CChartView;
};
```

```
// meastabl.h : header file



//

/////////////////////////////////////////////////////////////////////////////
// CMeasTable form view

#ifndef __AFXEXT_H__
#include <afxext.h>
#endif

class CMeasTable : public CFormView
{
        DECLARE_DYNCREATE(CMeasTable)
protected:
        CMeasTable();                   // protected constructor used by dynamic creati
n

// Form Data
public:
        //{{AFX_DATA(CMeasTable)
        enum { IDD = IDD_GRIDENTRY };
        CVBControl*     m_pGrid;
        //}}AFX_DATA

// Attributes
public:

// Operations
public:
        void OnInitialUpdate();     // first time after construct
        void SizeToFit();
        CPoint FindCellPosition(int nRow, int nCol);

// Implementation
protected:
        virtual ~CMeasTable();
        virtual void DoDataExchange(CDataExchange* pDX);        // DDX/DDV support
        // Generated message map functions
        //{{AFX_MSG(CMeasTable)
                // NOTE - the ClassWizard will add and remove member functions here.
        LRESULT OnOpenFile (WPARAM, LPARAM fileName);
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
```

```
// monctrld.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CMonCtrlDlg dialog

class CMonCtrlDlg : public CDialog
{
// Construction
public:
        CMonCtrlDlg(BOOL *pbEnable, CMonControl *pMonControl, CMonControl *pMonInfo);

        void PrepareData (void);

// Dialog Data
        //{{AFX_DATA(CMonCtrlDlg)
        enum { IDD = IDD_CONTROL };
        DWORD   m_nMaxTime1;
        DWORD   m_nMaxTime2;
        double  m_dTaskTemp1;
        double  m_dTaskTemp2;




        CString m_sTemp1;
        CString m_sTemp2;
        CString m_sTime1;
        CString m_sTime2;
        //}}AFX_DATA
private:
        BOOL *m_pbEnable;
        CMonControl *m_pMonControl;
        CMonControl *m_pMonInfo;
// Implementation
protected:
        virtual void DoDataExchange(CDataExchange* pDX);        // DDX/DDV support

        // Generated message map functions
        //{{AFX_MSG(CMonCtrlDlg)
        afx_msg HBRUSH OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor);
        virtual void OnOK();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
```

```
// monmeasd.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CMonMeasDlg dialog

class CMonMeasDlg : public CDialog
{
// Construction
public:
        CMonMeasDlg(CChannelCfg *pChannelCfg, double *pdMeasVal,
                                BOOL *bFocused, CMonMeasDlg **ppThis);

// Dialog Data
        //{{AFX_DATA(CMonMeasDlg)
        enum { IDD = IDD_MONITORING };
                // NOTE: the ClassWizard will add data members here
        //}}AFX_DATA
private:
        CChannelCfg *m_pChannelCfg;
        double *m_pdMeasVal;
        BOOL *m_bFocused;
        CMonMeasDlg **m_ppThis;

// Implementation
protected:
        virtual void DoDataExchange(CDataExchange* pDX);        // DDX/DDV support
        virtual void PostNcDestroy( );
        // Generated message map functions
        //{{AFX_MSG(CMonMeasDlg)
        virtual void OnCancel();
        afx_msg HBRUSH OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor);
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
```

```
/* NT MGMT.H -- Host application network management codes and messages.
 * Copyright (c) 1992, 1993 by Echelon Corporation.
 * All Rights Reserved.
 *
 * Date last modified: 6/11/93
 *
 * Network management codes and messages for an application node.
 * This file contains a subset of the LonTalk network management structures.
 * The types have been changed where necessary because of the difference
 * in representations between Neuron and MS-DOS objects.
 */




#define ID_STR_LEN 8            /* program ID length */

#define NULL_IDX  15            /* unused address table index */

typedef struct {
    bits    selector_hi : 6;
    bits    direction   : 1;
    bits    priority    : 1;
    bits    selector_lo : 8;
    bits    addr_index  : 4;
    bits    auth        : 1;
    bits    service     : 2;
    bits    turnaround  : 1;
} nv_struct;

// Note - Microsoft C will make nv_struct 4 bytes long because it does
// not allow odd-length bit-fields.  It is really 3 bytes long.
// Be careful when using sizeof() any structure that includes an nv_struct.

#define NM_update_nv_cnfg        0x6B
#define NM_update_nv_cnfg_fail   0x0B
#define NM_update_nv_cnfg_succ   0x2B

typedef struct {
    byte        code;
    nv_struct   nv_cnfg;
} NM_query_nv_cnfg_response;

#define NM_query_nv_cnfg        0x68
#define NM_query_nv_cnfg_fail   0x08
#define NM_query_nv_cnfg_succ   0x28

typedef enum {
    APPL_OFFLINE   = 0,             /* Soft offline state                   */
    APPL_ONLINE,
    APPL_RESET,
    CHANGE_STATE
} nm_node_mode;

typedef enum {
    APPL_UNCNFG    = 2,
    NO_APPL_UNCNFG = 3,
    CNFG_ONLINE    = 4,
    CNFG_OFFLINE   = 6,             /* Hard offline state                   */
    SOFT_OFFLINE   = 0xC
} nm_node_state;
```


71

```
typedef
    byte        code;
    byte        mode;               /* Interpret with 'nm_node_mode'        */
    byte        node_state;         /* Optional field if mode==CHANGE_STATE */
                                    /* Interpret with 'nm_node_state'       */
} NM_set_node_mode_request;

#define NM_set_node_mode 0x6C

typedef enum {
    ABSOLUTE           = 0,
    READ_ONLY_RELATIVE = 1,
    CONFIG_RELATIVE    = 2,
} nm_mem_mode;

typedef enum {
    NO_ACTION      = 0,
    BOTH_CS_RECALC = 1,

    CNFG_CS_RECALC = 4,
    ONLY_RESET     = 8,
    BOTH_CS_RECALC_RESET = 9,
    CNFG_CS_RECALC_RESET = 12,
} nm_mem_form;

typedef struct {
    byte    code;
    byte    mode;
    byte    offset_hi;
    byte    offset_lo;
    byte    count;
    byte    form;                   // followed by the data
} NM_write_memory_request;

#define NM_write_memory 0x6E

typedef struct {
    byte    length_hi;
    byte    length_lo;
    byte    num_netvars;
    byte    version;          // version 0 format
    byte    mtag_count;
} snvt_struct_v0;

typedef struct {
    byte    length_hi;
    byte    length_lo;
    byte    num_netvars_lo;
    byte    version;           // version 1 format
    byte    num_netvars_hi;
    byte    mtag_count;
} snvt_struct_v1;

// Partial list of SNVT type index values

typedef enum {
    SNVT_str_asc  = 36,
    SNVT_lev_cont = 21,
    SNVT_lev_disc = 22,
    SNVT_count_f  = 51,
} SNVT_t;

typedef struct {
    bits    nv_config_class     :1;
    bits    nv_auth_config      :1;
    bits    nv_priority_config  :1;
```

```
    bits    nv polled           :1;
    bits    nv sync             :1;
    bits    ext rec             :1;
    bits    snvt type index     :8;         // use enum SNVT_t
} snvt_desc_struct;

typedef struct {
#ifdef  _MSC_VER
    byte        mask;           // Microsoft C does not allow odd-length
#else                           // bit fields
    bits    unused :3;
    bits    nc      :1;     // array count
    bits    sd      :1;     // self-documentation
    bits    nm      :1;     // network variable name
    bits    re      :1;     // rate estimate
    bits    mre     :1;     // max rate estimate
#endif









} snvt_ext_rec_mask;

// Network management message codes

#define NM_query_SNVT           0x72
#define NM_query_SNVT_fail      0x12
#define NM_query_SNVT_succ      0x32

typedef struct {
    byte    code;
    byte    offset_hi;      // big-endian 16-bits
    byte    offset_lo;
    byte    count;
} NM_query_SNVT_request;

#define NM_wink 0x70

#define NM_NV_fetch         0x73
#define NM_NV_fetch_fail    0x13
#define NM_NV_fetch_succ    0x33

// Explicit application message codes

#define NA_appl_msg         0x00
#define NA_appl_offline     0x3F
#define NA_foreign_msg      0x40
#define NA_foreign_offline  0x4F

// Structure used by host application to store network variables

typedef enum { NV_IN = 0, NV_OUT = 1 } nv_direction;

typedef struct {                        // structure to define NVs
    int             size;               // number of bytes
    nv_direction    direction;          // input or output
    const           char * name;        // name of variable
    void            ( * print_func )( byte * ); // routine to print value
    void            ( * read_func )( byte * );  // routine to read value
    byte            data[ MAX_NETVAR_DATA ];    // actual storage for value
} network_variable;
```

```
                                                        definitions.
/*
 * Copyright (c) 1993 by Echelon Corporation.
 * All Rights Reserved.
 *
 * Date last modified: 6/11/93
 */

/*
 ************************************************************************
 * Application buffer structures for sending and receiving messages to and
 * from a network interface.  The 'ExpAppBuffer' and 'ImpAppBuffer'
 * structures define the application buffer structures with and without
 * explicit addressing.  These structures have up to four parts:
 *
 *   Network Interface Command (NI_Hdr)                    (2 bytes)
 *   Message Header (MsgHdr)                               (3 bytes)
 *   Network Address (ExplicitAddr)                        (11 bytes)
 *   Data (MsgData)                                        (varies)
 *
 * Network Interface Command (NI_Hdr):
 *
 *   The network interface command is always present.  It contains the
 *   network interface command and queue specifier.  This is the only
 *   field required for local network interface commands such as niRESET.
 *
 * Message Header (MsgHdr: union of NetVarHdr and ExpMsgHdr):
 *
 *   This field is present if the buffer is a data transfer or a completion
 *   event.  The message header describes the type of LONTALK message
 *   contained in the data field.
 *
 *   NetVarHdr is used if the message is a network variable message and
 *   network interface selection is enabled.
 *
 *   ExpMsgHdr is used if the message is an explicit message, or a network
 *   variable message and host selection is enabled (this is the default
 *   for the SLTA).
 *
 * Network Address (ExplicitAddr:  SendAddrDtl, RcvAddrDtl, or RespAddrDtl)
 *
 *   This field is present if the message is a data transfer or completion
 *   event, and explicit addressing is enabled.  The network address
 *   specifies the destination address for downlink application buffers,
 *   or the source address for uplink application buffers.  Explicit
 *   addressing is the default for the SLTA.
 *
 *   SendAddrDtl is used for outgoing messages or NV updates.
 *
 *   RcvAddrDtl is used  for incoming messages or unsolicited NV updates.
 *
 *   RespAddrDtl is used for incoming responses or NV updates solicited
 *   by a poll.
 *
 * Data (MsgData: union of UnprocessedNV, ProcessedNV, and ExplicitMsg)
 *
 *   This field is present if the message is a data transfer or completion
 *   event.
 *
```

```
 *   data are included.  This provides the NV index, the NV selector, or
 *   message code as appropriate.
 *
 *   UnprocessedNV is used if the message is a network variable update, and
 *   host selection is enabled. It consists of a two-byte header followed by
 *   the NV data.
 *
 *   ProcessedNV is used if the message is a network variable update, and
 *   network interface selection is enabled. It consists of a two-byte header
 *   followed by the NV data.
 *
 *   ExplicitMsg is used if the message is an explicit message.  It consists
 *   of a one-byte code field followed by the message data.
 *
 * Note - the fields defined here are for a little-endian (Intel-style)
 * host processor, such as the 80x86 processors used in PC compatibles.
 * Bit fields are allocated right-to-left within a byte.
 * For a big-endian (Motorola-style) host, bit fields are typically
 * allocated left-to-right.  For this type of processor, reverse
 * the bit fields within each byte.  Compare the NEURON C include files
 * ADDRDEFS.H and MSG_ADDR.H, which are defined for the big-endian NEURON
 * CHIP.
 ****************************************************************************
 */

/* Change the following declarations to port to a non-80x86  */

typedef unsigned char byte;      /* 8 bits */
typedef unsigned bits;           /* bit fields */
typedef unsigned int word;       /* 16 bits */

typedef enum { FALSE = 0, TRUE } boolean;




/*
 ****************************************************************************
 * Network Interface Command data structure.  This is the application-layer
 * header used for all messages to and from a LONWORKS network interface.
 ****************************************************************************
 */

/* Literals for the 'cmd.q.queue' nibble of NI_Hdr. */

typedef enum {
    niTQ          =  2,          /* Transaction queue                      */
    niTQ_P        =  3,          /* Priority transaction queue             */
    niNTQ         =  4,          /* Non-transaction queue                  */
    niNTQ_P       =  5,          /* Priority non-transaction queue         */
    niRESPONSE    =  6,          /* Response msg & completion event queue*/
    niINCOMING    =  8           /* Received message queue                 */
} NI_Queue;

/* Literals for the 'cmd.q.q_cmd' nibble of NI_Hdr. */

typedef enum {
    niCOMM          = 1,         /* Data transfer to/from network        */
    niNETMGMT       = 2          /* Local network management/diagnostics */
} NI_QueueCmd;

/* Literals for the 'cmd.noq' byte of NI_Hdr. */

typedef enum {
    niNULL          = 0x00,
    niTIMEOUT       = 0x30,      /* Not used                             */
    niCRC           = 0x40,      /* Not used                             */
    niRESET         = 0x50
```

```
    niFLUSH_CANCEL    = 0x60,       /* Downlink                          */
    niONLINE          = 0x70,
    niOFFLINE         = 0x80,
    niFLUSH           = 0x90,
    niFLUSH_IGN       = 0xA0,
    niSLEEP           = 0xB0,       /* SLTA only                         */
    niACK             = 0xC0,
    niNACK            = 0xC1,       /* SLTA only                         */
    niSSTATUS         = 0xE0,       /* SLTA only                         */
    niPUPXOFF         = 0xE1,
    niPUPXON          = 0xE2,
    niPTRHROTL        = 0xE4,       /* Not used                          */
    niDRV_CMD         = 0xF0,       /* Not used                          */
} NI_NoQueueCmd;

/*
 * Header for network interface messages.  The header is a union of
 * two command formats: the 'q' format is used for the niCOMM and
 * niNETMGMT commands that require a queue specification; the 'noq'
 * format is used for all other network interface commands.
 * Both formats have a length specification where:
 *
 *      length = header (3) + address field (11 if present) + data field
 *
 * WARNING:  The fields shown in this structure do NOT reflect the actual
 * structure required by the network interface.  Depending on the network
 * interface, the network driver may change the order of the data and add
 * additional fields to change the application-layer header to a link-layer
 * header.  See the description of the link-layer header in Chapter 2 of the
 * Host Application Programmer's Guide.
 */

typedef union {

    struct {
        bits queue  :4;         /* Network interface message queue       */
                                /* Use value of type 'NI_Queue'          */
        bits q_cmd  :4;         /* Network interface command with queue  */
                                /* Use value of type 'NI_QueueCmd'       */
        bits length :8;         /* Length of the buffer to follow        */
    } q;                        /* Queue option                          */
    struct {
        byte     cmd;           /* Network interface command w/o queue   */
                                /* Use value of type 'NI_NoQueueCmd'     */
        byte     length;        /* Length of the buffer to follow        */
    } noq;                      /* No queue option                       */
} NI_Hdr;

/*
 ***************************************************************************
 * Message Header structure for sending and receiving explicit
 * messages and network variables which are not processed by the
 * network interface (host selection enabled).
 ***************************************************************************
 */

/* Literals for 'st' fields of ExpMsgHdr and NetVarHdr. */

typedef enum {
   ACKD             = 0,
   UNACKD_RPT       = 1,
   UNACKD           = 2,
   REQUEST          = 3
} ServiceType;

/* Literals for 'cmp' code' fields of ExpMsgHdr and NetVarHdr. */
```

```
typedef enum {
    MSG_NOT_COMPL   = 0,            /* Not a completion event              */
    MSG_SUCCEEDS    = 1,            /* Successful completion event         */
    MSG_FAILS       = 2             /* Failed completion event             */
} ComplType;

/* Explicit message and Unprocessed NV Application Buffer. */

typedef struct {

    bits   tag        :4;           /* Message tag for implicit addressing  */
                                    /* Magic cookie for explicit addressing */
    bits   auth       :1;           /* 1 => Authenticated                   */
    bits   st         :2;           /* Service Type - see 'ServiceType'     */
    bits   msg_type   :1;           /* 0 => explicit message                */
                                    /*      or unprocessed NV               */
/*-------------------------------------------------------------------------*/
    bits   response   :1;           /* 1 => Response, 0 => Other            */
    bits   pool       :1;           /* 0 => Outgoing                        */
    bits   alt_path   :1;           /* 1 => Use path specified in 'path'    */
                                    /* 0 => Use default path                */
    bits   addr_mode  :1;           /* 1 => Explicit addressing,            */
                                    /* 0 => Implicit                        */
                                    /* Outgoing buffers only                */
    bits   cmpl_code  :2;           /* Completion Code - see 'ComplType'    */
    bits   path       :1;           /* 1 => Use alternate path,             */
                                    /* 0 => Use primary path                */
                                    /*      (if 'alt_path' is set)          */
    bits   priority   :1;           /* 1 => Priority message                */
/*-------------------------------------------------------------------------*/
    byte   length;                  /* Length of msg or NV to follow        */
                                    /* not including any explicit address   */
                                    /* field, includes code byte or         */




                                    /* selector bytes                       */
} ExpMsgHdr;

/*
 ***************************************************************************
 * Message Header structure for sending and receiving network variables
 * that are processed by the network interface (network interface
 * selection enabled).
 ***************************************************************************
 */

typedef struct {
    bits   tag        :4;           /* Magic cookie for correlating         */
                                    /* responses and completion events      */
    bits   rsvd0      :2;
    bits   poll       :1;           /* 1 => Poll, 0 => Other                */
    bits   msg_type   :1;           /* 1 => Processed network variable      */
/*-------------------------------------------------------------------------*/
    bits   response   :1;           /* 1 => Poll response, 0 => Other       */
    bits   pool       :1;           /* 0 => Outgoing                        */
    bits   trnarnd    :1;           /* 1 => Turnaround Poll, 0 => Other     */
    bits   addr_mode  :1;           /* 1 => Explicit addressing,            */
                                    /* 0 => Implicit addressing             */
    bits   cmpl_code  :2;           /* Completion Code - see above          */
    bits   path       :1;           /* 1 => Used alternate path             */
                                    /* 0 => Used primary path               */
                                    /*      (incoming only)                 */
    bits   priority   :1;           /* 1 => Priority msg (incoming only)    */
/*-------------------------------------------------------------------------*/
    byte   length;                  /* Length of network variable to follow */
                                    /* not including any explicit address   */
                                    /* ...                                  */
```

```
/* Union of all message headers. */

typedef union {
    ExpMsgHdr  exp;
    NetVarHdr  pnv;
} MsgHdr;

/*
 ****************************************************************************
 * Network Address structures for sending messages with explicit addressing
 * enabled.
 ****************************************************************************
 */

/* Literals for 'type' field of destination addresses for outgoing messages. */

typedef enum {
    UNASSIGNED      = 0,
    SUBNET_NODE     = 1,
    NEURON_ID       = 2,
    BROADCAST       = 3,
    IMPLICIT        = 126,    /* not a real destination type */
    LOCAL           = 127,    /* not a real destination type */
} AddrType;

/* Group address structure.  Use for multicast destination addresses. */

typedef struct {
    bits   size      :7;        /* Group size (0 => huge group)        */
    bits   type      :1;        /* 1 => Group                          */

    bits   rsvd0     :7;

    bits   domain    :1;        /* Domain index                        */

    bits   retry     :4;        /* Retry count                         */
    bits   rpt_timer :4;        /* Retry repeat timer                  */

    bits   tx_timer  :4;        /* Transmit timer index                */
    bits   rsvd1     :4;

    byte       group;           /* Group ID                            */
} SendGroup;

/* Subnet/node ID address.  Use for a unicast destination address. */

typedef struct {
    byte   type;                /* SUBNET_NODE                         */

    bits   node      :7;        /* Node number                         */
    bits   domain    :1;        /* Domain index                        */

    bits   retry     :4;        /* Retry count                         */
    bits   rpt_timer :4;        /* Retry repeat timer                  */

    bits   tx_timer  :4;        /* Transmit timer index                */
    bits   rsvd1     :4;

    bits   subnet    :8;        /* Subnet ID                           */
} SendSnode;

/* 48-bit NEURON ID destination address. */

#define NEURON_ID_LEN 6
```


48

```
    byte    type;                   /* NEURON_ID                          */

    bits    rsvd0       :7;
    bits    domain      :1;         /* Domain index                       */

    bits    retry       :4;         /* Retry count                        */
    bits    rpt_timer   :4;         /* Retry repeat timer                 */

    bits    tx_timer    :4;         /* Transmit timer index               */
    bits    rsvd2       :4;

    bits    subnet      :8;         /* Subnet ID, 0 => pass all routers   */
    byte    nid[ NEURON_ID_LEN ];   /* NEURON ID                          */
} SendNrnid;

/* Broadcast destination address. */

typedef struct {
    byte    type;                   /* BROADCAST                          */

    bits    backlog     :6;         /* Backlog                            */
    bits    rsvd0       :1;
    bits    domain      :1;         /* Domain index                       */

    bits    retry       :4;         /* Retry count                        */
    bits    rpt_timer   :4;         /* Retry repeat timer                 */

    bits    tx_timer    :4;         /* Transmit timer index               */
    bits    rsvd2       :4;

    bits    subnet      :8;         /* Subnet ID, 0 => domain-wide        */
} SendBcast;




/* Address formats for special host addresses.                            */

typedef struct {
    byte    type;                   /*  LOCAL                             */
} SendLocal;

typedef struct {
   byte     type;                   /* IMPLICIT */
   byte     msg_tag;                /* address table entry number */
} SendImplicit;

/* Union of all destination addresses. */

typedef union {
    SendGroup       gp;
    SendSnode       sn;
    SendBcast       bc;
    SendNrnid       id;
    SendLocal       lc;
    SendImplicit    im;
} SendAddrDtl;

/*
 ***************************************************************************
 * Network Address structures for receiving messages with explicit
 * addressing enabled.
 ***************************************************************************
 */

/* Received subnet/node ID destination address.  Used for unicast messages. */

typedef struct {
```

```
    bits        ???
    bits                :1;
} RcvSnode;

/* Received 48-bit NEURON ID destination address. */

typedef struct {
    byte   subnet;
    byte   nid[ NEURON_ID_LEN ];
} RcvNrnid;

/* Union of all received destination addresses. */

typedef union {
    byte       gp;                  /* Group ID for multicast destination   */
    RcvSnode   sn;                  /* Subnet/node ID for unicast           */
    RcvNrnid   id;                  /* 48-bit NEURON ID destination address */
    byte       subnet;              /* Subnet ID for broadcast destination  */
                                    /* 0 => domain-wide */
} RcvDestAddr;

/* Source address of received message.  Identifies */
/* network address of node sending the message.    */

typedef struct {
    bits   subnet  :8;
    bits   node    :7;
    bits           :1;
} RcvSrcAddr;

/* Literals for the 'format' field of RcvAddrDtl. */

typedef enum {




    ADDR_RCV_BCAST   = 0,
    ADDR_RCV_GROUP   = 1,
    ADDR_RCV_SNODE   = 2,
    ADDR_RCV_NRNID   = 3
} RcvDstAddrFormat;

/* Address field of incoming message. */

typedef struct {
#ifdef _MSC_VER
    byte    kludge;    /* Microsoft C does not allow odd-length bitfields */
#else
    bits    format      :6;     /* Destination address type             */
                                /* See 'RcvDstAddrFormat'               */
    bits    flex_domain :1;     /* 1 => broadcast to unconfigured node  */
    bits    domain      :1;     /* Domain table index                   */
#endif
    RcvSrcAddr  source;             /* Source address of incoming message   */
    RcvDestAddr dest;               /* Destination address of incoming msg  */
} RcvAddrDtl;

/*
 ****************************************************************************
 * Network Address structures for receiving responses with explicit
 * addressing enabled.
 ****************************************************************************
 */

/* Source address of response message. */

typedef struct {
    bits   subnet   :8;
```

```
    ???   is snode :1;          /* 0 => Group response,                  */
                                /* 1 => snode response                   */
} RespSrcAddr;

/* Destination of response to unicast request. */

typedef struct {
    bits   subnet   :8;
    bits   node     :7;
    bits            :1;
} RespSnode;

/* Destination of response to multicast request. */

typedef struct {
    bits   subnet   :8;
    bits   node     :7;
    bits            :1;
    bits   group    :8;
    bits   member   :6;
    bits            :2;
} RespGroup;

/* Union of all response destination addresses. */

typedef union {
    RespSnode  sn;
    RespGroup  gp;
} RespDestAddr;

/* Address field of incoming response. */

typedef struct {




#ifdef _MSC_VER
    byte     kludge;     /* Microsoft C does not allow odd-length bitfields */
#else
    bits                 :6;
    bits     flex_domain :1;    /* 1=> Broadcast to unconfigured node   */
    bits     domain      :1;    /* Domain table index                   */
#endif
    RespSrcAddr  source;            /* Source address of incoming response  */
    RespDestAddr dest;              /* Destination address of incoming resp */
} RespAddrDtl;

/* Explicit address field if explicit addressing is enabled. */

typedef union {
    RcvAddrDtl  rcv;
    SendAddrDtl snd;
    RespAddrDtl rsp;
} ExplicitAddr;

/*
 ****************************************************************************
 * Data field structures for explicit messages and network variables.
 ****************************************************************************
 */

/*
 * MAX_NETMSG_DATA specifies the maximum size of the data portion of an
 * application buffer.  MAX_NETVAR_DATA specifies the maximum size of the
 * data portion of a network variable update.  The values specified here
 * are the absolute maximums,based on the LONTALK protocol. Actual limits
 * are based on the buffer sizes defined on the attached NEURON CHIP.
 */
```

81

```
#define MAX_NETMSG_DATA 228
#define MAX_NETVAR_DATA 31

/* Data field for network variables (host selection enabled). */

typedef struct {
    bits   NV_selector_hi :6;
    bits   direction      :1;     /* 1 => output NV, 0 => input NV      */
    bits   must_be_one    :1;     /* Must be set to 1 for NV            */
    bits   NV_selector_lo :8;
    byte       data[ MAX_NETVAR_DATA ]; /* Network variable data          */
} UnprocessedNV;

/* Data field for network variables (network interface selection enabled).  */

typedef struct {
    byte       index;                /* Index into NV configuration table  */
    byte       rsvd0;
    byte       data[ MAX_NETVAR_DATA ]; /* Network variable data          */
} ProcessedNV;

/* Data field for explicit messages. */

typedef struct {
    byte       code;                 /* Message code                       */
    byte       data[ MAX_NETMSG_DATA ]; /* Message data                    */
} ExplicitMsg;

/* Union of all data fields. */

typedef union {
    UnprocessedNV unv;
    ProcessedNV   pnv;




    ExplicitMsg   exp;
} MsgData;

/*
 ****************************************************************************
 * Message buffer types.
 ****************************************************************************
 */

/* Application buffer when using explicit addressing. */

typedef struct {
    NI_Hdr       ni_hdr;          /* Network interface header           */
    MsgHdr       msg_hdr;         /* Message header                     */
    ExplicitAddr addr;            /* Network address                    */
    MsgData      data;            /* Message data                       */
} ExpAppBuffer;

/* Application buffer when not using explicit addressing. */

typedef struct {
    NI_Hdr       ni_hdr;          /* Network interface header           */
    MsgHdr       msg_hdr;         /* Message header                     */
    MsgData      data;            /* Message data                       */
} ImpAppBuffer;

/*
 ****************************************************************************
 * Network interface error codes.
 ****************************************************************************
 */
```

82

```
    NI_OK,
    NI_NO_DEVICE,
    NI_DRIVER_NOT_OPEN,
    NI_DRIVER_NOT_INIT,
    NI_DRIVER_NOT_RESET,
    NI_DRIVER_ERROR,
    NI_NO_RESPONSES,
    NI_RESET_FAILS,
    NI_TIMEOUT,
    NI_UPLINK_CMD,
    NI_INTERNAL_ERR,
} NI_Code;

/***************************** Externals *****************************/

NI_Code ni_init( const char * device_name);
                            /* Initialize network interface   */
void ni_close( void );      /* Close network interface */
NI_Code ni_reset( void );   /* Reset network interface        */

    /* Assumes only one network interface is open */

    /* Assume network interface is configured with explicit addressing ON,
       and network variable processing OFF */

NI_Code ni_send_msg_wait(
    ServiceType       service,          /* ACKD, UNACKD_RPT, UNACKD, REQUEST */
    const SendAddrDtl * out_addr,       /* address of outgoing message */
    const MsgData     * out_data,       /* data of outgoing message */
    int               out_length,       /* length of outgoing message */
    boolean           priority,         /* outgoing message priority */
    boolean           out_auth,         /* outgoing message authenticated */
    ComplType       * completion,       /* MSG_SUCCEEDS or MSG_FAILS */
    int             * num_responses,    /* number of received responses */
    RespAddrDtl     * in_addr,          /* address of first response */
    MsgData         * in_data,          /* data of first response */
    int             * in_length         /* length of first response */
);

NI_Code ni_get_next_response(         /* get subsequent responses here */
    RespAddrDtl     * in_addr,
    MsgData         * in_data,
    int             * in_length
);

NI_Code ni_receive_msg(
    ServiceType     * service,          /* ACKD, UNACKD_RPT, UNACKD, REQUEST */
    RcvAddrDtl      * in_addr,          /* address of incoming msg */
    MsgData         * in_data,          /* data of incoming msg */
    int             * in_length,        /* length of incoming msg */
    boolean         * in_auth           /* if incoming was authenticated */
);

NI_Code ni_send_response(           /* send response to last received request */
    MsgData       * out_data,       /* data for outgoing response */
    int             out_length      /* length of outgoing response */
);

NI_Code ni_send_immediate( NI_NoQueueCmd command );
          // send an immediate (no queue) command to network interface */

extern ExpAppBuffer  msg_out;           /* Outgoing message buffer       */
extern ExpAppBuffer  msg_in;            /* Incoming message buffer       */

extern void ni_msg_arrived( int msg_size, ServiceType service,
```

83

```
    int msg_tag );

void ni_error_display(const char * s, NI_Code ni_error);
                                    /* Display a network interface error*/

void ni_msg_display( ExpAppBuffer *msg_ptr );
```

```
//{{NO DEPENDENCIES}}
// App Studio generated include file.
// Used by LONWORKS.RC
//
#define IDR_MAINFRAME                   2
#define IDR_LONWORTYPE                  3
#define IDNEXT                          3
#define IDPREV                          4
#define IDR_CHARTTYPE                   4
#define IDD_ABOUTBOX                    100
#define IDD_CHANNEL_DLG                 102
#define IDD_MONITORING                  104
#define IDD_CONTROL                     105
#define IDD_GRIDENTRY                   106
#define IDD_CHAN_CHOICE                 108
#define IDC_CHAN_NAME                   1000
#define IDC_KIND_COMBO                  1001
#define IDC_UNIT_COMBO                  1002
#define IDC_CONV_COMBO                  1003
#define IDC_UNIT_VALUE                  1004
#define IDC_DISPLACEMENT                1005
#define IDC_LOW_ALARM                   1006
#define IDC_HIGH_ALARM2                 1007
#define IDC_COEFFICIENT                 1008
#define IDC_CHANNEL_NO                  1009



#define IDC_CHAN1_NAME                  1010
#define IDC_CHAN2_NAME                  1011
#define IDC_NOMIN_VALUE                 1011
#define IDC_CHAN3_NAME                  1012
#define IDC_CHAN4_NAME                  1013
#define IDC_CHAN5_NAME                  1014
#define IDC_CHAN1_VAL1                  1015
#define IDC_CHAN1_VAL2                  1016
#define IDC_CHAN1_VAL3                  1017
#define IDC_CHAN2_VAL1                  1018
#define IDC_CHAN2_VAL2                  1019
#define IDC_CHAN2_VAL3                  1020
#define IDC_CHAN3_VAL1                  1021
#define IDC_CHAN3_VAL2                  1022
#define IDC_CHAN3_VAL3                  1023
#define IDC_CHAN4_VAL1                  1024
#define IDC_CHAN4_VAL2                  1025
#define IDC_CHAN4_VAL3                  1026
#define IDC_CHAN5_VAL1                  1027
#define IDC_CHAN5_VAL2                  1028
#define IDC_CHAN5_VAL3                  1029
#define IDC_CHAN1_UNIT1                 1030
#define IDC_CHAN1_UNIT2                 1031
#define IDC_CHAN1_UNIT3                 1032
#define IDC_CHAN2_UNIT1                 1033
#define IDC_CHAN2_UNIT2                 1034
#define IDC_CHAN2_UNIT3                 1035
#define IDC_CHAN3_UNIT1                 1036
#define IDC_CHAN3_UNIT2                 1037
#define IDC_CHAN3_UNIT3                 1038
#define IDC_CHAN4_UNIT1                 1039
#define IDC_CHAN4_UNIT2                 1040
#define IDC_CHAN4_UNIT3                 1041
```

```
#define IDC_CHAN5_UNIT3                 1044
#define IDC_MAXTIME1                    1045
#define IDC_TASK_TEMP1                  1046
#define IDC_TIME1                       1047
#define IDC_TEMP1                       1048
#define IDC_GRID                        1050
#define IDC_CHANNEL1                    1051
#define IDC_RADIO2                      1052
#define IDC_RADIO3                      1053
#define IDC_RADIO4                      1054
#define IDC_RADIO5                      1055
#define IDC_PERCENT                     1056
#define IDC_MAXTIME2                    1057
#define IDC_SCALE1                      1057
#define IDC_TASK_TEMP2                  1058
#define IDC_SCALE2                      1058
#define IDC_TIME2                       1059
#define IDC_SCALE3                      1059
#define IDC_TEMP2                       1060
#define IDC_SCALE4                      1060
#define ID_OPTIONS_CHANNELS             32771
#define ID_MONITOR_MONITOROWANIEPOMIARW 32772
#define ID_MONITOR_RANDOM               32773
#define ID_MONITOR_STEROWANIEPOMIARAMI  32774
#define ID_HA_INIT                      32775
#define ID_HISTORIA_WYKRESY_DZIE        32778
#define ID_HISTORIA_WYKRESY_MIESIC      32779
#define ID_HISTORIA_WYKRESY_ROK         32780
#define ID_PROGNOZA_WYKRESY_DZIE        32782
#define ID_PROGNOZA_WYKRESY_MIESIC      32784
#define ID_PROGNOZA_WYKRESY_ROK         32785



#define ID_KONFIGURACJA_EKRANY          32786
#define ID_BILANSE_ENERGIA              32787
#define ID_BILANSE_KONFIGURACJA         32788
#define ID_PARAMETRY_TARYFY             32789
#define ID_PARAMETRY_LIMITY             32790
#define ID_HISTORIA_KANA                32793
#define ID_KONFIGURACJA_WYWIETLANIELOGU 32796
#define AFX_MIN_STRING_SIZE             61216

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS

#define _APS_NEXT_RESOURCE_VALUE        109
#define _APS_NEXT_COMMAND_VALUE         32797
#define _APS_NEXT_CONTROL_VALUE         1058
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

```
// spliter.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CSpliter frame with splitter

#ifndef __AFXEXT_H__
#include <afxext.h>
#endif

class CSpliter : public CMDIChildWnd
{
        DECLARE_DYNCREATE(CSpliter)
protected:
        CSpliter();                     // protected constructor used by dynamic creatio
n

// Attributes
protected:
        CSplitterWnd    m_wndSplitter;
public:

// Operations
public:

// Implementation
public:
        virtual ~CSpliter();
        virtual BOOL OnCreateClient(LPCREATESTRUCT lpcs, CCreateContext* pContext);

        // Generated message map functions
        //{{AFX_MSG(CSpliter)
        afx_msg void OnFilePrint();
        afx_msg void OnFilePrintPreview();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
```

```
// stdafx.h : include file for standard system include files,
//  or project specific include files that are used frequently, but
//      are changed infrequently
//

#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions (including VB)
```

```
// validate.h : header file

void AFXAPI DDV_MinChars(CDataExchange*, CString const&, int);
```