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

074 A

ZESPÓŁ AUTOMATYKI ELEKTRONICZNEJ
Nazwa ONB/ZNB

Główny wykonawca mgr inż. Tadeusz Goszczyński

Wykonawcy: mgr inż. Elżbieta Jachczyk
tech. Andrzej Kulik

Komputerowe stanowisko pomiarowe KAL-LEG do badań legalizacyjnych par czujników temperatury dla elektronicznych przeliczników ciepła.
Etap 5. Ocena prototypu , wprowadzenie niezbędnych zmian i uzupełnień.

DOKUMENT WZORCOWY

(Tytuł pracy, numer i tytuł etapu)

Zleceniodawca **KBN**

Praca własna PIAP

Kierownik Zespołu
doc.dr inż. J. Korytkowski

Z-ca Dyrektora
d/s Bad.-Rozwojowych
dr inż. Jan Jabłkowski

Pracę zakończono dnia **30.11.1998r.**

Nr arch. 7597 Nr zlecenia **1782C i 9660C**

Analiza deskryptorowa

BADANIA TECHNICZNE + CZUJNIKI TEMPERATURY

Abstrakt

Sprawozdanie zawiera :
ocenę prototypu , niezbędne zmiany i uzupełnienia

Tytuły poprzednich sprawozdań

Komputerowe stanowisko pomiarowe do badań legalizacyjnych par czujników temperatury dla elektronicznych przeliczników ciepła.

Etap 1. Opracowanie układów pomiarowych i sporządzenie dokumentacji konstrukcyjnej prototypu . - Nr arch. 7492
Etap 3.Wykonanie prototypu. Opracowanie oprogramowania. Opracowanie instrukcji badań prototypu. - Nr arch. 7569
Etap 4. Wykonanie prób i badań prototypu. - Nr arch. 7586

Rozdzielnik

Egz. 1. ..... OIN .....

Egz. 2. ..... ZAE-1 .....

Egz. 3. ..... ZAE-3 .....

Egz. 4 ZAE- 3

2

Spis treści:

1. Wprowadzenie.

Prace w etapie 5 obejmowały ocenę prototypu stanowiska KAL-LEG w porozumieniu z GUM, tak by przygotować stanowisko do badań mających na celu dopuszczenie stanowiska do użytkowania w kraju, co jest przedmiotem 6 etapu pracy.
Prace wykonywane były w następujących kierunkach:

- oceny funkcjonalnej jakości stanowiska na podstawie badań wykonanych w etapie 5;
- oceny dokładności całego stanowiska wraz z zakupionymi jego częściami: termostatami, czujnikami wzorcowymi i rezystorami wzorcowymi oraz multimetrem do pomiaru rezystancji;
- przygotowanie specjalnej wersji oprogramowania stanowiska umozliwiającego wykonanie przez GUM sprawdzenia poprawności obliczeń na podstawie posiadanych przez GUM danych pomiarowych z ich stanowiska.

2. Główne wnioski i uwagi.

Ocena funkcjonalnej jakości wykazała, że stanowisko działa poprawnie przy wszystkich badanych warunkach pracy i sposobach połączeń badanych par temperatury. Nie są wymagane żadne zmiany ani uzupełnienia stanowiska.
Ocena dokładności stanowiska zostanie przekazana do GUM w ramach zlecenia badań mających na celu dopuszczenie stanowiska do użytkowania w kraju. Porównanie do obowiązujących norm wykazuje wystarczającą dokładność stanowiska.
Zmiany i uzupełnienia stanowiska objęły opracowanie na życzenie GUM i wykonanie specjalnej wersji oprogramowania stanowiska umożliwiającego wykonanie sprawdzenia poprawności obliczeń na podstawie posiadanych przez GUM danych pomiarowych z ich stanowiska. W tym celu wprowadzono między innymi okna do wprowadzenia danych z klawiatury uruchomiane w odpowiednich momentach pracy stanowiska oraz przekazywanie danych o nastawionych wartościach temperatur w termostatach do programu obliczeniowego.

3. Ocena funkcjonalnej jakości stanowiska.

Ocena funkcjonalnej jakości wykazała, że stanowisko działa poprawnie przy wszystkich badanych warunkach pracy i sposobach połączeń badanych par temperatury. Nie są wymagane żadne zmiany ani uzupełnienia stanowiska.

4. Ocena dokładności stanowiska

Poprzedni etap pracy - Etap. 4. Wykonanie prób i badań prototypu - umożliwił dokonanie oceny prototypu stanowiska KAL-LEG. Zarówno próba pracy długotrwałej, badania funkcjonalne oraz badania bezpieczeństwa wypadły pozytywnie. Nie wystąpiły podczas nich zakłócenia w pracy stanowiska.
Badania powtarzalności pomiarów, w których zastosowano zestawy precyzyjnych oporników umieszczone w stałej temperaturze, symulujących pary czujników jak i oporowy termometr wzorcowy, umożliwiły wyznaczenie błędów wnoszonych wyłącznie przez sterownik stanowiska KAL-LEG.
Niepewność pomiaru stanowiska można określić na podstawie normy dotyczącej ciepłomierzy EN 1434-4, wg której niepewność wyposażenia testującego nie powinna przekraczać 1/5 maksymalnych dopuszczalnych błędów ciepłomierza lub jego części składowych. Z kolei EN 1434-1 określa maksymalny dopuszczalny błąd pojedynczego czujnika pary w odniesieniu do czujnika idealnego jako 2 K a maksymalny dopuszczalny błąd pary czujników jako zależność od minimalnej różnicy temperatur Δt_{min} pary czujników i zmierzonej różnicy temperatur Δt wyrażoną wzorem:

$$E_t = \pm(0{,}5 + 3\Delta t_{min}/\Delta t)$$

Z powyższego wynika dopuszczalna niepewność stanowiska w wyznaczaniu charakterystyki pojedynczego czujnika pary wynosząca < 0,4°C.
Dopuszczalna niepewność stanowiska w wyznaczaniu błędu pary czujników jest zmienna i wynosi w najgorszym przypadku 0,1%.
Na całkowitą niepewność wyznaczania błędu charakterystyki pojedynczego czujnika składają się: błąd rezystancyjnego termometru wzorcowego (Tinsley) wynoszący ±0,001°C, błąd od wahań temperatury w łaźni termostatu wynoszący ±0,01°C, błąd wnoszony przez multimetr KEITHLEY wynoszący 0,008°C dla pomiaru 100Ω oraz błąd wnoszony przez sterownik stanowiska KAL-LEG. W zał. 4 sprawozdania z poprzedniego etapu pracy znajduje się wyznaczona na podstawie badań powtarzalności wyników niepewność maksymalna sterownika wynosząca ±0,00025°C. Powyższe wartości dają całkowitą niepewność stanowiska wynoszącą:

d = ±0,02°C < 0,4°C

Na podstawie wyników badań powtarzalności pomiarów wyznaczono w Tabl. 1...3 błędy maksymalne wyznaczania charakterystyki pary czujników w odniesieniu do pary czujników idealnych i porównano z dopuszczalnym błędem pary czujników temperatury dla danej różnicy temperatur.

Δt [°C]	3	10	20	20	100	180	180
δ_{dop} pary. [%]	3,5	3,5	2	0,95	0,8	0,667	0,55
$\lvert\delta_{max}\rvert$ stanowiska [%]	0,027	0,013	0,0078	0,003	0,001	0,00023	0,0004
udział = $\lvert\delta_{max}\rvert / \delta_{dop}$	**0,008**	**0,004**	**0,004**	**0,0032**	**0,0012**	**0,0004**	**0,0007**

Tabela 1. Zestawienie dla Pt 100

Δt [°C]	3	5	10	10	10	20	20
δ_{dop} pary [%]	3,5	3,5	3,5	2	1,4	2	1,25
$\lvert\delta_{max}\rvert$ stanowiska [%]	0,0092	0,004	0,003	0,003	0,004	0,00245	0,00075
udział = $\lvert\delta_{max}\rvert / \delta_{dop}$	**0,0026**	**0,0028**	**0,004**	**0,0008**	**0,0028**	**0,0012**	**0,0006**

Δt [°C]	20	100	100	120	150	150	180
δ_{dop} pary [%]	0,95	0,8	0,65	0,625	0,7	0,56	0,55
$\lvert\delta_{max}\rvert$ stanowiska [%]	0,0017	0,0004	0,0002	0,00013	0,0004	0,00012	0,00014
udział = $\lvert\delta_{max}\rvert / \delta_{dop}$	**0,0018**	**0,0005**	**0,0003**	**0,0002**	**0,0006**	**0,0002**	**0,00025**

Tabela 2. Zestawienie dla Pt 500

Δt [°C]	3	5	10	20	20	180	190
δ_{dop} pary [%]	3,5	3,5	2	1,25	0,95	0,583	0,579
$\lvert\delta_{max}\rvert$ stanowiska [%]	0,005	0,014	0,0073	0,0038	0,0009	0,00014	0,00024
udział = $\lvert\delta_{max}\rvert / \delta_{dop}$	**0,0015**	**0,004**	**0,004**	**0,003**	**0,001**	**0,0002**	**0,0004**

Tabela 3. Zestawienie dla Pt 1000

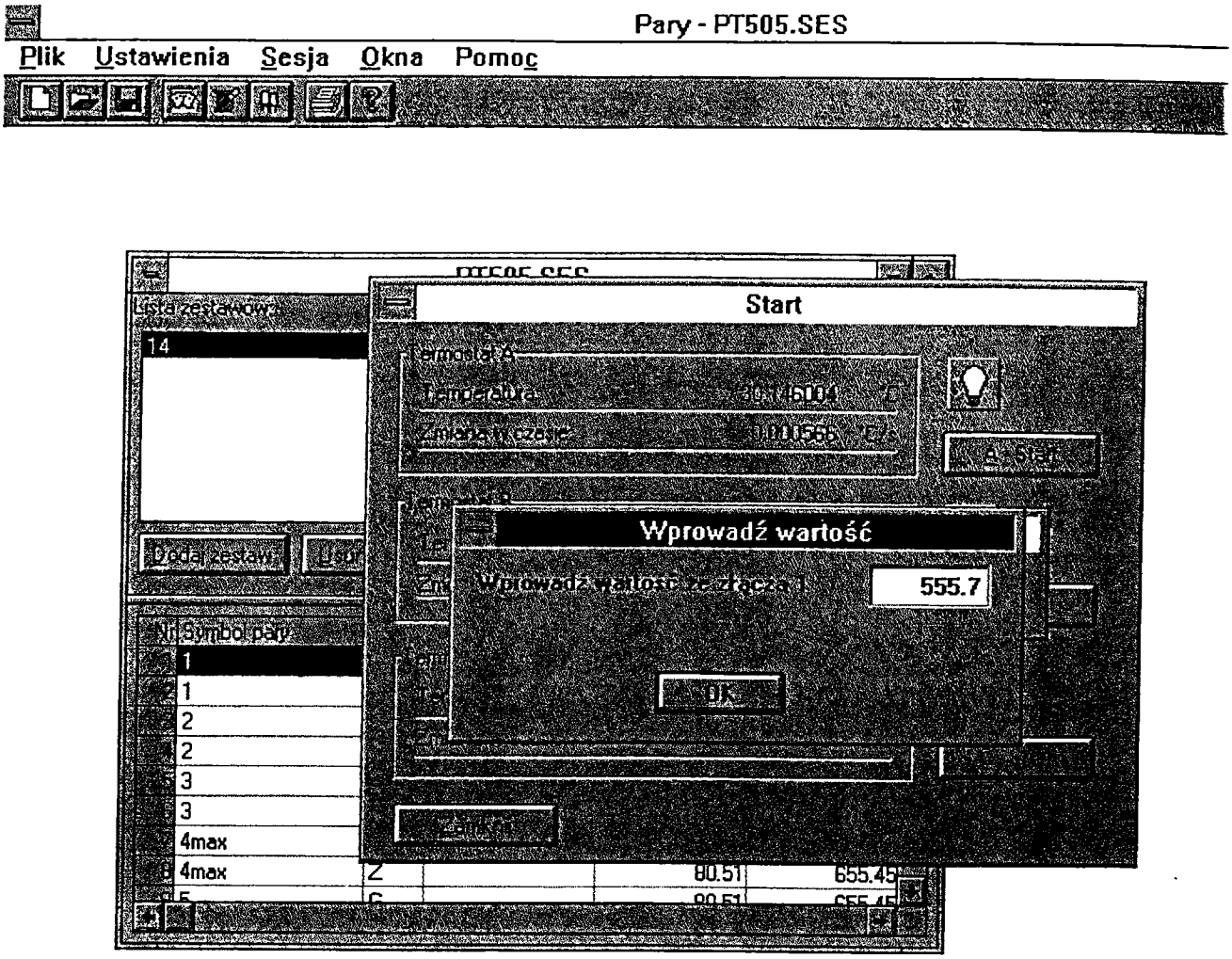
Ponieważ rozważania dotyczą pary czujników, nie odgrywają tutaj roli błędy wnoszone przez rezystancyjny termometr wzorcowy (Tinsley) ani przez multimetr. Błąd wnoszony przez wahania temperatury w termostacie jest pomijalny, ponieważ do wyznaczania charakterystyki pary brane są wartości rezystancji i temperatury, bądące wynikiem uśrednienia 3 kolejnych pomiarów.
Na podstawie Tabeli 1 określono maksymalny udział błędu stanowiska w dopuszczalnym błędzie pary czujników. Wynosi on:

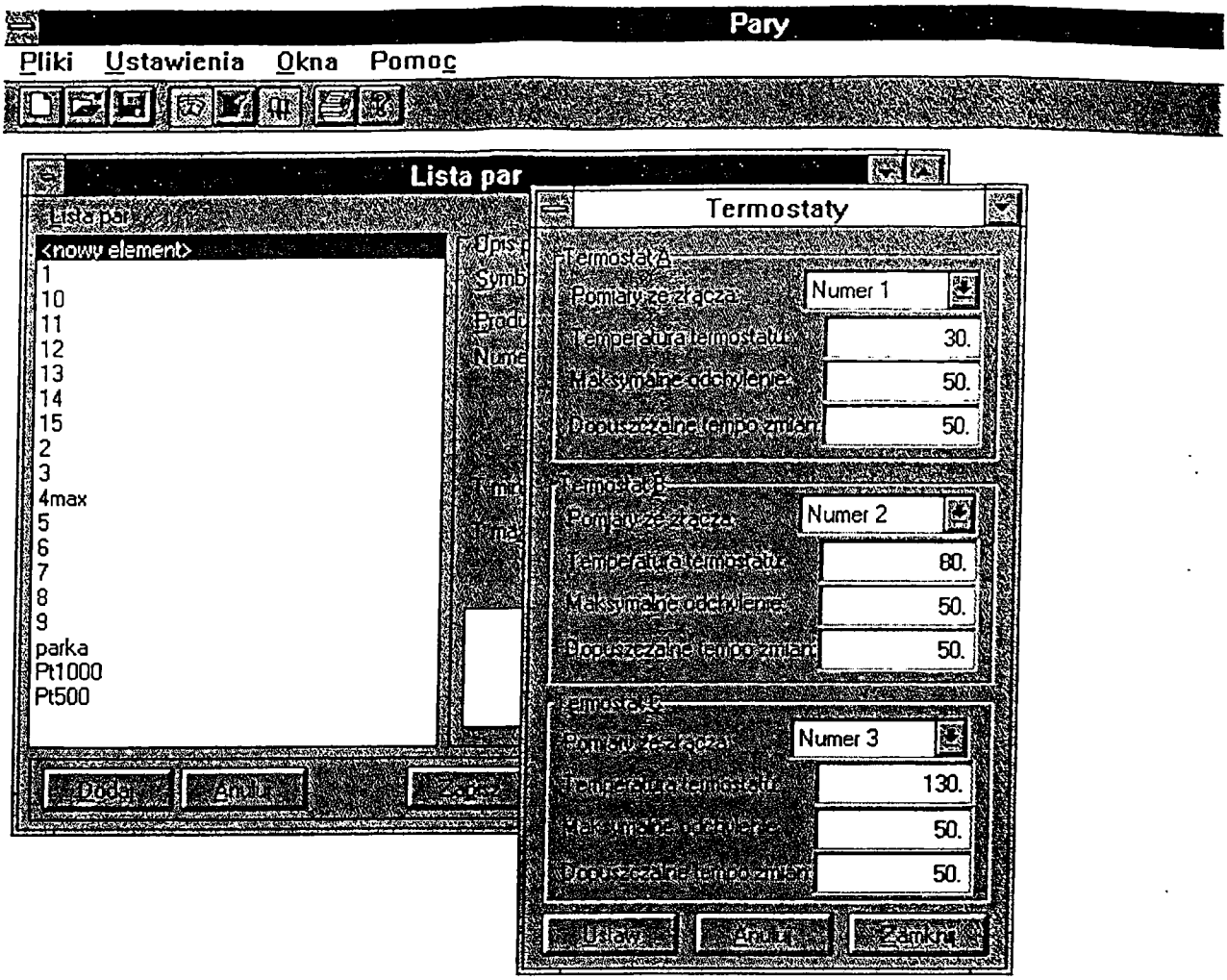
$$\frac{\lceil \delta_{max} \rceil}{\delta_{dop}} = 0{,}008 < 0{,}2$$

Wartość graniczna 0,2 wynika w zwymagań normy EN 1434-4 na której niepewność wyposażenia testującego.
Z powyższych rozważań wynika, że dokładność stanowiska KAL-LEG jest wystarczająca do badania charakterystyk par czujników temperatury i spełnia wymagania normy EN 1434.

5. Opis uzupełnienia oprogramowania stanowiska.

Przygotowano specjalną wersję oprogramowania stanowiska umożliwiającą wykonanie przez GUM sprawdzenia poprawności obliczeń na podstawie posiadanych przez GUM danych pomiarowych z ich stanowiska. ✓
Poniżej przedstawiono wygląd ekranów do wprowadzania danych z klawiatury oraz wydruki zbiorów zawierających te fragmenty oprogramowania.


Pary - PT505.SES
Plik Ustawienia Sesja Okna Pomoc
Start
14
Wprowadź wartość
555.7
OK
Nr Symbol pary
1
1
2
2
3
3
4max
4max


Pary
Pliki Ustawienia Okna Pomoc
Lista par
Lista par
<nowy element>
1
10
11
12
13
14
15
2
3
4max
5
6
7
8
9
parka
Pt1000
Pt500
Dodaj
Anuluj
Zapisz
Termostaty
Termostat A
Pomiary ze złącza:
Numer 1
Temperatura termostatu:
30.
Maksymalne odchylenie:
50.
Dopuszczalne tempo zmian:
50.
Termostat B
Pomiary ze złącza:
Numer 2
Temperatura termostatu:
80.
Maksymalne odchylenie:
50.
Dopuszczalne tempo zmian:
50.
Termostat C
Pomiary ze złącza:
Numer 3
Temperatura termostatu:
130.
Maksymalne odchylenie:
50.
Dopuszczalne tempo zmian:
50.
Ustaw
Anuluj
Zamknij

11

```
// //
#include "stdafx.h"
#include "pairs.h"
#include "cterm.h"
#include "statusdl.h"
#include "const.h"
#include "progress.h"
#include <math.h>


IMPLEMENT_SERIAL(CTermostat, CObject, 0);

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

#ifdef _DEBUG
static DWORD dwTicker[3];
static DWORD dwTickBase[3];
#endif

//////////////////////////////////////////////////////////////////////////////
//
//  IMPLEMENTACJA KLASY CTermostat - uogólniona komunikacja z aparatem
//
//////////////////////////////////////////////////////////////////////////////


WORD Nrczuj = 1;


//----------------------------------------------------------------------------
        CTermostat::CTermostat()
//----------------------------------------------------------------------------
{
    m_bIsActive = FALSE;;
    m_wPlug = 0;


    m_dT = 0.0;
    m_dDivergence = 0.0;
    m_dTdt = 0.0;

    m_dTnow = 0.0;
    m_dTdtnow = 0.0;
    m_bIsStable = FALSE;
}

//----------------------------------------------------------------------------
void CTermostat::SetParameters(WORD wPlug, double dT, double dDiv, double dTdt)
//----------------------------------------------------------------------------
{
    m_wPlug = wPlug;
    m_dT = dT;
    m_dDivergence = dDiv;
    m_dTdt = dTdt;
}

//----------------------------------------------------------------------------
void CTermostat::GetParameters(int& wPlug, double& dT, double& dDiv, double& dTdt)
//----------------------------------------------------------------------------
{
    wPlug = (int)m_wPlug;
    dT = m_dT;
    dDiv = m_dDivergence;
    dTdt = m_dTdt;
}


//----------------------------------------------------------------------------
void CTermostat::CheckTemperature(CDialog* pDlg)
//----------------------------------------------------------------------------
{

    //peek message
        MSG msg;                                          //| PM_NOYIELD
        while(PeekMessage((LPMSG)&msg,NULL,0,0,PM_REMOVE ))
        {
```

```
        while(PeekMessage((LPMSG)&msg,NULL,0,0,PM_REMOVE ))
        {
            if((!pDlg)||(!pDlg->IsDialogMessage(&msg)))
            {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
            }
        }

        // termostat przeprowadza pomiar
        // sygnalizacja
        if (pPA!=NULL)
        {
            m_pTermA->CheckTemperature(pDlg);
            pDlg->UpdateDisplay(ITEM_A, m_pTermA->m_dTnow, m_pTermA->m_dDelta, m_pTermA->IsStal
        }
        if (pPB!=NULL)
        {
            pDlg->UpdateDisplay(ITEM_B, m_pTermB->m_dTnow, m_pTermB->m_dDelta, m_pTermB->IsStal
            m_pTermB->CheckTemperature(pDlg);
        }
        if (pPC!=NULL)
        {
            pDlg->UpdateDisplay(ITEM_C, m_pTermC->m_dTnow, m_pTermC->m_dDelta, m_pTermC->IsStal
            m_pTermC->CheckTemperature(pDlg);
        }

        if ((pDlg) && (pDlg->m_bHasBeenCanceled))
            break;

        // jeśli któryś z przycisków Start, uruchamiamy następną operację
        if (pDlg->m_bStartA)
        {
            StartActivityOnChannel(m_pTermA, pPA);
            pDlg->m_bStartA = FALSE;
        }

        if (pDlg->m_bStartB)
        {
            StartActivityOnChannel(m_pTermB, pPB);
            pDlg->m_bStartB = FALSE;
        }

        if (pDlg->m_bStartC)
        {
            StartActivityOnChannel(m_pTermC, pPC);
            pDlg->m_bStartC = FALSE;
        }

    }


    GetApp()->m_pMainWnd->EnableWindow(TRUE);
    delete pDlg;
}
```

```
    CFileException fe;

    if (!file.Open(_szPathName, CFile::modeCreate |
      CFile::modeReadWrite | CFile::shareExclusive, &fe))
        AfxMessageBox(IDS_MSG_CANTSAVE, MB_ICONEXCLAMATION);

    CArchive saveArchive(&file, CArchive::store | CArchive::bNoFlushOnDelete);

    TRY
    {
        GetApp()->BeginWaitCursor();

        m_pTermA->Serialize(saveArchive);
        m_pTermB->Serialize(saveArchive);
        m_pTermC->Serialize(saveArchive);

        saveArchive.Close();
        file.Close();
    }
    CATCH_ALL(e)
    {
        file.Abort(); // will not throw an exception
        GetApp()->EndWaitCursor();

        AfxMessageBox(IDS_MSG_CANTSAVE, MB_ICONEXCLAMATION);
    }
    END_CATCH_ALL
    bRetVal = TRUE;
    GetApp()->EndWaitCursor();

    return bRetVal;
}


//--------------------------------------------------------------------------
void    CMeasuringDevice::OnSetTermParameters()
//--------------------------------------------------------------------------
{
    if (m_pSettingsDlg != NULL)
    {
        TRACE0 ("The m_pSettingsDlg window marked as already opened!\n");
        return;
    }

    CTermostatyDlg * pDlg = new CTermostatyDlg();
    ASSERT(pDlg);

    m_pTermA->GetParameters(pDlg->m_nPlugA, pDlg->m_dTempA, pDlg->m_dOdchA, pDlg->m_dTA);
    m_pTermB->GetParameters(pDlg->m_nPlugB, pDlg->m_dTempB, pDlg->m_dOdchB, pDlg->m_dTB);
    m_pTermC->GetParameters(pDlg->m_nPlugC, pDlg->m_dTempC, pDlg->m_dOdchC, pDlg->m_dTC);

    pDlg->Create(IDD_DLG_TERMOSTATY,GetApp()->m_pMainWnd);

    m_pSettingsDlg = pDlg;


}

//--------------------------------------------------------------------------
void CMeasuringDevice::SetTermParameters()
//--------------------------------------------------------------------------
{
    ASSERT(m_pSettingsDlg);
    if (m_pSettingsDlg!=NULL)
    {
        m_pTermA->SetParameters((WORD)  m_pSettingsDlg->m_nPlugA,
                                        m_pSettingsDlg->m_dTempA,
                                        m_pSettingsDlg->m_dOdchA,
                                        m_pSettingsDlg->m_dTA);

        m_pTermB->SetParameters((WORD)  m_pSettingsDlg->m_nPlugB,
                                        m_pSettingsDlg->m_dTempB,
                                        m_pSettingsDlg->m_dOdchB,
                                        m_pSettingsDlg->m_dTB);

        m_pTermC->SetParameters((WORD)  m_pSettingsDlg->m_nPlugC,
                                        m_pSettingsDlg->m_dTempC,
                                        m_pSettingsDlg->m_dOdchC,
                                        m_pSettingsDlg->m_dTC);
    }
```

```
}

static char BASED_CODE _szDispString[]="Proszę czekać... przeprowadzam pomiar na elemencie %d!

//-------------------------------------------------------------------------
void    CMeasuringDevice::StartActivityOnChannel(CTermostat * pT, CMsment * pPTrg)
//-------------------------------------------------------------------------
{
    ASSERT ((pT==m_pTermA) || (pT==m_pTermB) || (pT==m_pTermC));
    ASSERT(pPTrg);

    // funkcja ma pomierzyć wszystkie wartości na wskazanym termostacie
    ASSERT(pT->m_wPlug!=0);



    pPTrg->m_dTemp = pT->m_dT;
    pPTrg->m_wTerm = (pT==m_pTermA)?1:((pT==m_pTermB)?2:3);
    pPTrg->m_wPlug = pT->m_wPlug;
    pPTrg->m_ctWhen = CTime::GetCurrentTime();

    CProgressDlg * pDlg = new CProgressDlg;
    ASSERT(pDlg);

    char sBuff[255];
// Ela
//  NrCzuj=0;
    TRY // aby mozna było przerwać pomiar
        // np. w przypadku zwisu hardware'u
    {
        for ( int i=0; i<PRS_IN_AGG_CNT; i++ )
        {
        i++;
        --i;
            if (!(pPTrg->m_dwMask & (0x1 << i)))
                continue;

            sprintf(sBuff, _szDispString, i+1);
            pDlg->DisplayInfo(sBuff);
#ifdef _DEBUG
            DWORD dwTick = ::GetTickCount();
            while (GetTickCount()<dwTick+500) {}
#endif
            pPTrg->m_aRes[i]=pT->GetItemR(i);

        //POZNAN
            pPTrg->m_aTemp[i]=pT->GetItemT(i);
        //   pPTrg->m_aTemp[i]=pT->m_dT;


        }
    }
    CATCH_ALL(e)
    {
    }
    END_CATCH_ALL

    pDlg->DestroyWindow();
    delete pDlg;

    pPTrg->m_bDone = TRUE;

}

//-------------------------------------------------------------------------
void    CMeasuringDevice::EnterMeasureLoop(CMsment * pPA, CMsment * pPB, CMsment * pPC)
//-------------------------------------------------------------------------
{
    GetApp()->m_pMainWnd->EnableWindow(FALSE);

    CStatusDlg  * pDlg = new CStatusDlg(NULL);
    ASSERT(pDlg);

    m_pTermA->Reset();
    m_pTermB->Reset();
    m_pTermC->Reset();

    for (;;)
    {
        //peek message
        MSG msg;                                        //| PM_NOYIELD
```

```
            if((!pDlg)||(!pDlg->IsDialogMessage(&msg)))
            {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
            }
        }

//GG



    if (m_wPlug!=0)
    {
    WORD    wIt = 15;
    if (GetDevice()->SelectItem(wIt, m_wPlug))

            m_dTnow = GetDevice()->GetTemperature(m_wPlug, TRUE);
    }
    else
        m_dTnow = TEMP_NOMS;
}

//---------------------------------------------------------------------------
BOOL CTermostat::IsStable()
//---------------------------------------------------------------------------
{
    if (m_wPlug==0)
        return FALSE;

    if (m_dwTicker==0)
    {
        m_dwTicker = GetTickCount();
        m_dTlast = m_dTnow;
        return FALSE;
    }

    if ((GetTickCount()-m_dwTicker)<TEST_QUANT)
        return m_bIsStable;

//POZNAN
//  CheckTemperature(pDlg);
//        m_dTnow=  this->m_dT;

    DWORD dwCheckPoint = GetTickCount();
    m_dDelta = (fabs(m_dTlast - m_dTnow))/((dwCheckPoint - m_dwTicker)*0.001);

    if ((!(m_dDelta>m_dTdt)) && (fabs(m_dT-m_dTnow) < m_dDivergence))
        m_bIsStable = TRUE;
    else
        m_bIsStable = FALSE;
    m_dTlast = m_dTnow;
    m_dwTicker = dwCheckPoint;
    return m_bIsStable;
}

//---------------------------------------------------------------------------
double CTermostat::GetItemT(WORD wIt)
//---------------------------------------------------------------------------
{
    // zwraca temperaturę dla itemu wIt
//Ela   ustawia 15 w Pobierz wartosci
//  wIt = 14;
    if (GetDevice()->SelectItem(wIt, m_wPlug))
        return GetDevice()->GetTemperature(m_wPlug);
    else
        return TEMP_NOMS;
}

//---------------------------------------------------------------------------
double CTermostat::GetItemR(WORD wIt)
//---------------------------------------------------------------------------
{
    // zwraca temperaturę dla itemu wIt
    if (GetDevice()->SelectItem(wIt, m_wPlug))
        return GetDevice()->GetResistance(m_wPlug);
    else
        return TEMP_NOMS;
}
```

```
//--------------------------------------------------------------------------
void CTermostat::Reset()
//--------------------------------------------------------------------------
{
    m_dTlast = 0.0;
    m_dTnow = 0.0;
    m_dTdtnow = 0.0;
    m_bIsStable = FALSE;
    m_dwTicker = 0;

#ifdef _DEBUG
    if (m_wPlug!=0)
    {
        dwTicker[m_wPlug-1]=GetTickCount();
        dwTickBase[m_wPlug-1] = dwTicker[m_wPlug-1];
    }
#endif
}

//--------------------------------------------------------------------------
void CTermostat::Serialize( CArchive& ar )
//--------------------------------------------------------------------------
{
    CObject::Serialize(ar);
    if (ar.IsStoring())
    {
        ar << (DWORD)m_bIsActive;
        ar << m_wPlug;

        ar << m_dT;
        ar << m_dDivergence;
        ar << m_dTdt;
    }
    else // ar.IsLoading()
    {
        DWORD dwIsActive;
        ar >> dwIsActive;
        ar >> m_wPlug;

        ar >> m_dT;
        ar >> m_dDivergence;
        ar >> m_dTdt;
        m_bIsActive = (BOOL)dwIsActive;
    }

}

////////////////////////////////////////////////////////////////////////////
//
//  IMPLEMENTACJA KLASY CInterfaceIEC625 - podstawowe operacje na czujnikach
//
////////////////////////////////////////////////////////////////////////////


//--------------------------------------------------------------------------
double CInterfaceIEC625::GetTemperature(WORD wChannel, BOOL bWaitingForTemperatureOK)
//--------------------------------------------------------------------------
{
    BOOL bNoHardware = FALSE;
    BOOL bDemo = FALSE;
    BOOL bFromKeyb = FALSE;

#ifdef NO_HARDWARE
    bNoHardware = TRUE;
#endif

#ifdef DEMO
    bDemo = TRUE;
#endif

#ifdef FROM_KEYB
    bFromKeyb = TRUE;
#endif

    if(!bNoHardware && !(bFromKeyb && bWaitingForTemperatureOK))
    {
        double dR, dT;
//Ela
        WORD temper=999;
        pomiar.PobierzWartosci(wChannel,temper, &dT, &dR);
```

Nr arch.7597
Str. 15

17

```
        if(bFromKeyb)
            dT = GetTargetTemperature(wChannel);

        return dT;
    }
//TG

    if(bDemo || bFromKeyb)
    {
#ifdef _DEBUG
        dwTicker[wChannel] = GetTickCount()+1; // na wypadek wypadku
        double dDif = (dwTicker[wChannel]-dwTickBase[wChannel])/1000 + 0.1;
#endif
        double dTarget = GetTargetTemperature(wChannel);

        return ((log((double)dDif)+dTarget)/dDif) + dTarget;
    //   return dTarget;
    }
    else
        return 0;

}

//---------------------------------------------------------------------------
double CInterfaceIEC625::GetResistance(WORD wChannel)
//---------------------------------------------------------------------------
{

 # define rezyst(t,R0,A,B) (R0*(1+A*t+B*t*t))
  double r0 = 100.0; double a=0.003909667; double b = -5.93333333333343e-7;


#ifndef NO_HARDWARE
    double dR, dT;
    pomiar.PobierzWartosci(wChannel,Nrczuj, &dT, &dR);

    ++Nrczuj;

    return dR;
#endif

#ifdef _DEBUG
    dwTicker[wChannel] = GetTickCount()+1; // na wypadek wypadku
    double dDif = (dwTicker[wChannel]-dwTickBase[wChannel])/1000 + 0.1;

    double dTarget = GetTargetTemperature(wChannel);

// TGDEMO
    double rez = rezyst(dTarget,r0,a,b);
    return  rez;

//  return ((log((double)dDif)+dTarget)/dDif) + dTarget * 1.5;
#else
    return 0;
#endif
}

//---------------------------------------------------------------------------
double CInterfaceIEC625::GetTargetTemperature(WORD wChannel)
//---------------------------------------------------------------------------
{
    double dTarget = 0;

    if(wChannel == GetVirtDevice()->m_pTermA->m_wPlug)
        dTarget = GetVirtDevice()->m_pTermA->m_dT;
    else if(wChannel == GetVirtDevice()->m_pTermB->m_wPlug)
        dTarget = GetVirtDevice()->m_pTermB->m_dT;
    else if(wChannel == GetVirtDevice()->m_pTermC->m_wPlug)
        dTarget = GetVirtDevice()->m_pTermC->m_dT;
    else
        ASSERT(FALSE);

    return dTarget;
}

//---------------------------------------------------------------------------
BOOL CInterfaceIEC625::SelectItem(WORD wItem, WORD wChannel)
//---------------------------------------------------------------------------
{
    if ((wChannel>0) && (wChannel<4))
    {
```

```
            pomiar.PrzelaczNaCzujnik(wChannel, wItem);
            return TRUE;
        }
        else
            return FALSE;
    }

    ////////////////////////////////////////////////////////////////////////////
    //
    //  IMPLEMENTACJA KLASY CMeasuringDevice - obejmuje całość operacji pomiarowych
    //
    ////////////////////////////////////////////////////////////////////////////

    //---------------------------------------------------------------------------
        CMeasuringDevice::CMeasuringDevice()
    //---------------------------------------------------------------------------
    {
        // contstructor
        m_bBusy = FALSE;
        m_pSettingsDlg = NULL;
        m_pTermA = new CTermostat;
        m_pTermB = new CTermostat;
        m_pTermC = new CTermostat;
    }

    //---------------------------------------------------------------------------
        CMeasuringDevice::~CMeasuringDevice()
    //---------------------------------------------------------------------------
    {
        // contstructor
        delete m_pTermA;
        delete m_pTermB;
        delete m_pTermC;
    }


    static char BASED_CODE _szPathName[]="termdat.dat";
    //---------------------------------------------------------------------------
    BOOL    CMeasuringDevice::OnInitialize()
    //---------------------------------------------------------------------------
    {
        BOOL bRetVal = FALSE;
        CFile file;
        CFileException fe;


        if (!file.Open(_szPathName, CFile::modeRead | CFile::shareDenyWrite, &fe))
            AfxMessageBox(IDS_MSG_CANTLOAD, MB_ICONEXCLAMATION);
        else
        {
            CArchive loadArchive(&file, CArchive::load | CArchive::bNoFlushOnDelete);
            TRY
            {
                GetApp()->BeginWaitCursor();
                m_pTermA->Serialize(loadArchive);
                m_pTermB->Serialize(loadArchive);
                m_pTermC->Serialize(loadArchive);

                loadArchive.Close();
                file.Close();
            }
            CATCH_ALL(e)
            {
                file.Abort(); // will not throw an exception
                GetApp()->EndWaitCursor();
            }
            END_CATCH_ALL

            GetApp()->EndWaitCursor();
            bRetVal = TRUE;
        }

        return bRetVal;
    }

    //---------------------------------------------------------------------------
    BOOL    CMeasuringDevice::OnDeactivate()
    //---------------------------------------------------------------------------
    {
        BOOL bRetVal = FALSE;
        CFile file;
```

```
    CFileException fe;

    if (!file.Open(_szPathName, CFile::modeCreate |
      CFile::modeReadWrite | CFile::shareExclusive, &fe))
        AfxMessageBox(IDS_MSG_CANTSAVE, MB_ICONEXCLAMATION);

    CArchive saveArchive(&file, CArchive::store | CArchive::bNoFlushOnDelete);

    TRY
    {
        GetApp()->BeginWaitCursor();

        m_pTermA->Serialize(saveArchive);
        m_pTermB->Serialize(saveArchive);
        m_pTermC->Serialize(saveArchive);

        saveArchive.Close();
        file.Close();
    }
    CATCH_ALL(e)
    {
        file.Abort(); // will not throw an exception
        GetApp()->EndWaitCursor();

        AfxMessageBox(IDS_MSG_CANTSAVE, MB_ICONEXCLAMATION);
    }
    END_CATCH_ALL
    bRetVal = TRUE;
    GetApp()->EndWaitCursor();

    return bRetVal;
}


//-----------------------------------------------------------------------------
void    CMeasuringDevice::OnSetTermParameters()
//-----------------------------------------------------------------------------
{
    if (m_pSettingsDlg != NULL)
    {
        TRACE0 ("The m_pSettingsDlg window marked as already opened!\n");
        return;
    }

    CTermostatyDlg * pDlg = new CTermostatyDlg();
    ASSERT(pDlg);

    m_pTermA->GetParameters(pDlg->m_nPlugA, pDlg->m_dTempA, pDlg->m_dOdchA, pDlg->m_dTA);
    m_pTermB->GetParameters(pDlg->m_nPlugB, pDlg->m_dTempB, pDlg->m_dOdchB, pDlg->m_dTB);
    m_pTermC->GetParameters(pDlg->m_nPlugC, pDlg->m_dTempC, pDlg->m_dOdchC, pDlg->m_dTC);

    pDlg->Create(IDD_DLG_TERMOSTATY,GetApp()->m_pMainWnd);

       SettingsDlg = pDlg;



}

//-----------------------------------------------------------------------------
void CMeasuringDevice::SetTermParameters()
//-----------------------------------------------------------------------------
{
    ASSERT(m_pSettingsDlg);
    if (m_pSettingsDlg!=NULL)
    {
        m_pTermA->SetParameters((WORD)  m_pSettingsDlg->m_nPlugA,
                                        m_pSettingsDlg->m_dTempA,
                                        m_pSettingsDlg->m_dOdchA,
                                        m_pSettingsDlg->m_dTA);

        m_pTermB->SetParameters((WORD)  m_pSettingsDlg->m_nPlugB,
                                        m_pSettingsDlg->m_dTempB,
                                        m_pSettingsDlg->m_dOdchB,
                                        m_pSettingsDlg->m_dTB);

        m_pTermC->SetParameters((WORD)  m_pSettingsDlg->m_nPlugC,
                                        m_pSettingsDlg->m_dTempC,
                                        m_pSettingsDlg->m_dOdchC,
                                        m_pSettingsDlg->m_dTC);
    }
```

```
}

static char BASED_CODE _szDispString[]="Proszę czekać... przeprowadzam pomiar na elemencie %d!";

//-----------------------------------------------------------------------------
void    CMeasuringDevice::StartActivityOnChannel(CTermostat * pT, CMsment * pPTrg)
//-----------------------------------------------------------------------------
{
    ASSERT ((pT==m_pTermA) || (pT==m_pTermB) || (pT==m_pTermC));
    ASSERT(pPTrg);

    // funkcja ma pomierzyć wszystkie wartości na wskazanym termostacie
    ASSERT(pT->m_wPlug!=0);



    pPTrg->m_dTemp = pT->m_dT;
    pPTrg->m_wTerm = (pT==m_pTermA)?1:((pT==m_pTermB)?2:3);
    pPTrg->m_wPlug = pT->m_wPlug;
    pPTrg->m_ctWhen = CTime::GetCurrentTime();

    CProgressDlg * pDlg = new CProgressDlg;
    ASSERT(pDlg);

    char sBuff[255];
// Ela
//  NrCzuj=0;
    TRY // aby mozna było przerwać pomiar
        // np. w przypadku zwisu hardware'u

        for ( int i=0; i<PRS_IN_AGG_CNT; i++ )
        {
        i++;
        --i;
            if (!(pPTrg->m_dwMask & (0x1 << i)))
                continue;

            sprintf(sBuff, _szDispString, i+1);
            pDlg->DisplayInfo(sBuff);
    #ifdef _DEBUG
            DWORD dwTick = ::GetTickCount();
            while (GetTickCount()<dwTick+500) {}
    #endif
            pPTrg->m_aRes[i]=pT->GetItemR(i);

        //POZNAN
            pPTrg->m_aTemp[i]=pT->GetItemT(i);
        //   pPTrg->m_aTemp[i]=pT->m_dT;


        }
    }
    CATCH_ALL(e)


    END_CATCH_ALL

    pDlg->DestroyWindow();
    delete pDlg;

    pPTrg->m_bDone = TRUE;

}

//-----------------------------------------------------------------------------
void    CMeasuringDevice::EnterMeasureLoop(CMsment * pPA, CMsment * pPB, CMsment * pPC)
//-----------------------------------------------------------------------------
{
    GetApp()->m_pMainWnd->EnableWindow(FALSE);

    CStatusDlg  * pDlg = new CStatusDlg(NULL);
    ASSERT(pDlg);

    m_pTermA->Reset();
    m_pTermB->Reset();
    m_pTermC->Reset();

    for (;;)
    {
        //peek message
        MSG msg;                                        //| PM_NOYIELD
```

```
            if((!pDlg)||(!pDlg->IsDialogMessage(&msg)))
            {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
            }
        }

//GG



    if (m_wPlug!=0)
    {
    WORD    wIt = 15;
    if (GetDevice()->SelectItem(wIt, m_wPlug))

            m_dTnow = GetDevice()->GetTemperature(m_wPlug, TRUE);
    }
    else
        m_dTnow = TEMP_NOMS;
}

//---------------------------------------------------------------------------
BOOL CTermostat::IsStable()
//---------------------------------------------------------------------------
{
    if (m_wPlug==0)
        return FALSE;

       (m_dwTicker==0)
    {
        m_dwTicker = GetTickCount();
        m_dTlast = m_dTnow;
        return FALSE;
    }

    if ((GetTickCount()-m_dwTicker)<TEST_QUANT)
        return m_bIsStable;

//POZNAN
//  CheckTemperature(pDlg);
//        m_dTnow=  this->m_dT;

    DWORD dwCheckPoint = GetTickCount();
    m_dDelta = (fabs(m_dTlast - m_dTnow))/((dwCheckPoint - m_dwTicker)*0.001);

    if ((!(m_dDelta>m_dTdt)) && (fabs(m_dT-m_dTnow) < m_dDivergence))
        m_bIsStable = TRUE;
    else
        m_bIsStable = FALSE;
    m_dTlast = m_dTnow;
    m_dwTicker = dwCheckPoint;
       rn m_bIsStable;
}

//---------------------------------------------------------------------------
double CTermostat::GetItemT(WORD wIt)
//---------------------------------------------------------------------------
{
    // zwraca temperaturę dla itemu wIt
//Ela   ustawia 15 w Pobierz wartosci
//  wIt = 14;
    if (GetDevice()->SelectItem(wIt, m_wPlug))
        return GetDevice()->GetTemperature(m_wPlug);
    else
        return TEMP_NOMS;
}

//---------------------------------------------------------------------------
double CTermostat::GetItemR(WORD wIt)
//---------------------------------------------------------------------------
{
    // zwraca temperaturę dla itemu wIt
    if (GetDevice()->SelectItem(wIt, m_wPlug))
        return GetDevice()->GetResistance(m_wPlug);
    else
        return TEMP_NOMS;
}
```

```
//--------------------------------------------------------------------------
void CTermostat::Reset()
//--------------------------------------------------------------------------
{
    m_dTlast = 0.0;
    m_dTnow = 0.0;
    m_dTdtnow = 0.0;
    m_bIsStable = FALSE;
    m_dwTicker = 0;

#ifdef _DEBUG
    if (m_wPlug!=0)
    {
        dwTicker[m_wPlug-1]=GetTickCount();
        dwTickBase[m_wPlug-1] = dwTicker[m_wPlug-1];
    }
#endif
}

//--------------------------------------------------------------------------
void CTermostat::Serialize( CArchive& ar )
//--------------------------------------------------------------------------
{
    CObject::Serialize(ar);
    if (ar.IsStoring())
    {
        ar << (DWORD)m_bIsActive;
        ar << m_wPlug;

        ar << m_dT;
        ar << m_dDivergence;
        ar << m_dTdt;
    }
    else // ar.IsLoading()
    {
        DWORD dwIsActive;
        ar >> dwIsActive;
        ar >> m_wPlug;

        ar >> m_dT;
        ar >> m_dDivergence;
        ar >> m_dTdt;
        m_bIsActive = (BOOL)dwIsActive;
    }

}

////////////////////////////////////////////////////////////////////////////
//
//  IMPLEMENTACJA KLASY CInterfaceIEC625 - podstawowe operacje na czujnikach
//
////////////////////////////////////////////////////////////////////////////


//--  ------------------------------------------------------------------------
double CInterfaceIEC625::GetTemperature(WORD wChannel, BOOL bWaitingForTemperatureOK)
//--------------------------------------------------------------------------
{
    BOOL bNoHardware = FALSE;
    BOOL bDemo = FALSE;
    BOOL bFromKeyb = FALSE;

#ifdef NO_HARDWARE
    bNoHardware = TRUE;
#endif

#ifdef DEMO
    bDemo = TRUE;
#endif

#ifdef FROM_KEYB
    bFromKeyb = TRUE;
#endif

    if(!bNoHardware && !(bFromKeyb && bWaitingForTemperatureOK))
    {
        double dR, dT;
//Ela
        WORD temper=999;
        pomiar.PobierzWartosci(wChannel,temper, &dT, &dR);
```

```
        if(bFromKeyb)
            dT = GetTargetTemperature(wChannel);

        return dT;
    }
//TG

    if(bDemo || bFromKeyb)
    {
#ifdef _DEBUG
        dwTicker[wChannel] = GetTickCount()+1; // na wypadek wypadku
        double dDif = (dwTicker[wChannel]-dwTickBase[wChannel])/1000 + 0.1;
#endif
        double dTarget = GetTargetTemperature(wChannel);

        return ((log((double)dDif)+dTarget)/dDif) + dTarget;
    //   return dTarget;
    }
    else
        return 0;

}

//---------------------------------------------------------------------------
double CInterfaceIEC625::GetResistance(WORD wChannel)
//---------------------------------------------------------------------------
{

 # define rezyst(t,R0,A,B) (R0*(1+A*t+B*t*t))
  double r0 = 100.0; double a=0.003909667; double b = -5.93333333333343e-7;


#ifndef NO_HARDWARE
    double dR, dT;
    pomiar.PobierzWartosci(wChannel,Nrczuj, &dT, &dR);

    ++Nrczuj;

    return dR;
#endif

#ifdef _DEBUG
    dwTicker[wChannel] = GetTickCount()+1; // na wypadek wypadku
    double dDif = (dwTicker[wChannel]-dwTickBase[wChannel])/1000 + 0.1;

    double dTarget = GetTargetTemperature(wChannel);

// TGDEMO
    double rez = rezyst(dTarget,r0,a,b);
    return  rez;

//  return ((log((double)dDif)+dTarget)/dDif) + dTarget * 1.5;
#else
    return 0;
#endif
}

//---------------------------------------------------------------------------
double CInterfaceIEC625::GetTargetTemperature(WORD wChannel)
//---------------------------------------------------------------------------
{
    double dTarget = 0;

    if(wChannel == GetVirtDevice()->m_pTermA->m_wPlug)
        dTarget = GetVirtDevice()->m_pTermA->m_dT;
    else if(wChannel == GetVirtDevice()->m_pTermB->m_wPlug)
        dTarget = GetVirtDevice()->m_pTermB->m_dT;
    else if(wChannel == GetVirtDevice()->m_pTermC->m_wPlug)
        dTarget = GetVirtDevice()->m_pTermC->m_dT;
    else
        ASSERT(FALSE);

    return dTarget;
}

//---------------------------------------------------------------------------
BOOL CInterfaceIEC625::SelectItem(WORD wItem, WORD wChannel)
//---------------------------------------------------------------------------
{
    if ((wChannel>0) && (wChannel<4))
    {
```

```
        pomiar.PrzelaczNaCzujnik(wChannel, wItem);
        return TRUE;
    }
    else
        return FALSE;
}

////////////////////////////////////////////////////////////////////////////
//
//  IMPLEMENTACJA KLASY CMeasuringDevice - obejmuje całość operacji pomiarowych
//
////////////////////////////////////////////////////////////////////////////

//--------------------------------------------------------------------------
    CMeasuringDevice::CMeasuringDevice()
//--------------------------------------------------------------------------
{
    // contstructor
    m_bBusy = FALSE;
    m_pSettingsDlg = NULL;
    m_pTermA = new CTermostat;
    m_pTermB = new CTermostat;
    m_pTermC = new CTermostat;
}

//--------------------------------------------------------------------------
    CMeasuringDevice::~CMeasuringDevice()
//--------------------------------------------------------------------------
{
        :ontstructor
    ~ :ete m_pTermA;
    delete m_pTermB;
    delete m_pTermC;
}


static char BASED_CODE _szPathName[]="termdat.dat";
//--------------------------------------------------------------------------
BOOL    CMeasuringDevice::OnInitialize()
//--------------------------------------------------------------------------
{
    BOOL bRetVal = FALSE;
    CFile file;
    CFileException fe;


    if (!file.Open(_szPathName, CFile::modeRead | CFile::shareDenyWrite, &fe))
        AfxMessageBox(IDS_MSG_CANTLOAD, MB_ICONEXCLAMATION);
    else
    {
        CArchive loadArchive(&file, CArchive::load | CArchive::bNoFlushOnDelete);
        TRY
        {
            GetApp()->BeginWaitCursor();
            m_pTermA->Serialize(loadArchive);
            m_pTermB->Serialize(loadArchive);
            m_pTermC->Serialize(loadArchive);

            loadArchive.Close();
            file.Close();
        }
        CATCH_ALL(e)
        {
            file.Abort(); // will not throw an exception
            GetApp()->EndWaitCursor();
        }
        END_CATCH_ALL

        GetApp()->EndWaitCursor();
        bRetVal = TRUE;
    }

    return bRetVal;
}

//--------------------------------------------------------------------------
BOOL    CMeasuringDevice::OnDeactivate()
//--------------------------------------------------------------------------
{
    BOOL bRetVal = FALSE;
    CFile file;
```

```
/* pomiar.cpp */
////////////////////////////////////////////////////

#include "stdafx.h"
#include "pomiar.h"
#include "math.h"
#include "conio.h"
#include "dlgfromk.h"

/*
extern "C" int ieseg(int);
extern "C" void ieinit(int,int,int);
extern "C" int ieoutput(int,char*,int);
extern "C" int ieabort(void);
extern "C" int ieenter(int,char*,int);
*/
#include "iewinlib.h"


#define bn    -5.775e-7
#define an    0.0039083

#define d0    439.932854
#define d1    472.418020
#define d2    37.684494
#define d3    7.472018
#define d4    2.920828
#define d5    0.005184
#define d6    -0.963864
#define d7    -0.188732
#define d8    0.191203
#define d9    0.049025


/*************************************************************/
void ini_karty(void)
{
double ss;
int err,a;
for(a=0; a<65; a++)
    ss= 234,67/3.32;
    ieseg(0xD000);
    ieseg(0xD000);
        ieseg(0xD000);
    ieseg(0xD000);
        ieseg(0xD000);
    ieseg(0xD000);
for(a=0; a<650; a++)
    ss= 234,67/3.32;
  ieinit(0x2b8,21,0);
  ieinit(0x2b8,21,0);
  ie  t(0x2b8,21,0);
  i   t(0x2b8,21,0);
  ieinit(0x2b8,21,0);
  ieinit(0x2b8,21,0);
for(a=0; a<65; a++)
    ss= 234,67/3.32;
//  ieinit(0x2b8,21,0x00);
    ieabort();
    ieabort();
    ieabort();
    ieabort();
for(a=0; a<65; a++)
    ss= 234,67/3.32;

}

////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
void CPomiar::Delay(int czas)
{
    CTime poczatek = CTime::GetCurrentTime();
    //CParyDoc::printf("\r\n Czekaj %d sek. *",czas);
    while(1)
    {
        CTime teraz= CTime::GetCurrentTime();
        CTimeSpan roznica = teraz - poczatek;
                int  roz =(int) roznica.GetTotalSeconds();
        if( czas<= roz )
        break;
```

```
        while(1)
        {
        CTime sek= CTime::GetCurrentTime();
        roznica = sek - teraz;
        if( 1 <= (int) roznica.GetTotalSeconds() )
        ; //CParyDoc::printf("*");
        break;
        }
    }
}

/*************************************************************/

#define rezyst(t,R0,A,B) (R0*(1+A*t+B*t*t))

    struct
    {
    double temp_g;
    double temp_z;
    double bl_dop;
    } oblpar[6]= {  33.0,  30.0,  0.105,
                    45.0,  40.0,  0.165,
                    80.0,  70.0,  0.25,
                    104.0,  85.0, 0.475,
                    150.0, 110.0, 0.5,
                    170.0,  70.0, 1.25
                };


       uct D2 {
    ~ ble t;
    double r;
    } m_d2g[3];
//  struct D2 m_d2g[3];
    struct D2 m_d2z[3];


// double te;


void CPomiar::ObliczRAB(struct D2 *s ,double* R0, double* A, double* B)
{
    double w1=(s[0].r*s[1].t-s[1].r*s[0].t)*(s[1].r*s[2].t*s[2].t-s[2].r*s[1].t*s[1].t)-(s[0].r*s[1].t*s[1
].t-s[1].r*s[0].t*s[0].t)*(s[1].r*s[2].t-s[2].r*s[1].t);//
    double w2=(s[1].t-s[0].t)*(s[2].t-s[1].t)*(s[2].t-s[0].t); //warunek na trzy rozne punkty t
    if (w1*w2==0)
    return;
    double AA=(s[1].r-s[0].r)*(s[1].r*s[2].t*s[2].t-s[2].r*s[1].t*s[1].t)-(s[0].r*s[1].t*s[1].t-s[1].r*s[0
].t*s[0].t)*(s[2].r-s[1].r);
    AA=AA/w1;
    double BB=(s[0].r*s[1].t-s[1].r*s[0].t)*(s[2].r-s[1].r)-(s[1].r*s[2].t-s[2].r*s[1].t)*(s[1].r-s[0].r);
    BB=BB/w1;
    d ble RR=s[0].r/(1+AA*s[0].t+BB*s[0].t*s[0].t);
    *  = AA;
    *  = BB;
    *R0= RR;
}




/*   STARA na komputerze Pentium
//////////////////////////////////////
void CPomiar::SprawdzCzuj1()
{
    CString wyniki_new;
    wyniki = wyniki_new;
    sprintf(ok,"   ZLY   ");
    m_nOKey =1;
    double temp_char, rez, tn, rn,dr, bl, blad_dp;

    ObliczRAB( m_d2g, &m_dRg,&m_dAg,&m_dBg);


    sprintf(wyn,"\r\n   TABELA WYNIKOW CZUJNIKA WODY GORACEJ \r\n");
    wyniki += wyn;
    sprintf(wyn,"\r\n|------|-------|-------|--------|------------|---------|---------|\r\n");
    wyniki += wyn;
    sprintf(wyn,  "|  Lp. |   t   |  dR   | t czuj |     Elt    |   Edop  |  Nrearch.7597\n");
    wyniki += wyn;
```

```
    sprintf(wyn,  "|------|-------|-------|--------|------------|---------|---------|\r\n");wyniki += wyn;
    sprintf(wyn,  "|  -   |  °C   |  om   |   °C   |     °C     |   °C    |         |\r\n");wyniki += wyn;
    sprintf(wyn,  "|------|-------|-------|--------|------------|---------|---------|\r\n");wyniki += wyn;



    for(int i=0; i<6; i++)
    {
        temp_char= tmin + i * ((tmax-tmin)/5);
        rez = rezyst(temp_char,m_dRg,m_dAg,m_dBg);
        TempNorma( rez, &tn);
        rn=RezNorma(temp_char);
        dr=rez-rn;
        bl = tn - temp_char;
        blad_dp = 2.0;
// (0.30+0.005*temp_char)*r0n*(an+2*bn*temp_char);
        if(fabs(bl) - blad_dp >0)
        {
            sprintf(ok,"   ZLY   ");
            m_n0Key = 0;
        }
        else
        sprintf(ok, "  DOBRY  ");
        sprintf(wyn,"|  %-4d|  %-4.01f |%-6.21f | %-6.21f |",i+1,temp_char,dr,tn);
        wyniki += wyn;
        if(fabs(bl)<1.0E3)  {
            sprintf(wyn,"   %-9.31f|",bl);
            wyniki += wyn;
        }
        else  {
            sprintf(wyn,"   error    |");
            wyniki += wyn;
        }
        sprintf(wyn,"  %-6.21f |%s|\r\n",blad_dp,ok);
        wyniki += wyn;
        sprintf(wyn,  "|------|-------|-------|--------|------------|---------|---------|\r\n");
        wyniki += wyn;
    }

}
////////////////////////////////////
void CPomiar::SprawdzCzuj2()
{
    double temp_char, rez, tn, rn,dr, bl, blad_dp;
    ObliczRAB(m_d2z, &m_dRz,&m_dAz,&m_dBz);
    sprintf(wyn,"\r\n   TABELA WYNIKOW CZUJNIKA WODY ZIMNEJ  \r\n");
    wyniki += wyn;
    sprintf(wyn,"\r\n|------|-------|-------|-------|------------|---------|---------|\r\n");
    wyniki += wyn;
    sprintf(wyn,  "|  Lp. |   t   | R obl | R nor |     Elt    |   Edop  |  ocena  |\r\n");
    wyniki += wyn;
    sprintf(wyn,  "|------|-------|-------|-------|------------|---------|---------|\r\n");wyniki += wyn;
    sprintf(wyn,  "|  -   |  °C   |  om   |  om   |     °C     |   °C    |         |\r\n");wyniki += wyn;
    sprintf(wyn,  "|------|-------|-------|-------|------------|---------|---------|\r\n");wyniki += wyn;

    for(int i=0; i<6; i++)
    {
        temp_char= tmin + i * ((tmax-tmin)/5);
        rez = rezyst(temp_char,m_dRz,m_dAz,m_dBz);

        TempNorma( rez, &tn);
        rn=RezNorma(temp_char);
        dr=rez-rn;
        bl = tn - temp_char;
        blad_dp = 2.0;
// wersja blad w omach
//      tn=temp_char ;
//      rn=RezNorma(temp_char);
//      bl = rez - rn;
//      blad_dp =  0.7;
//
//      (0.30+0.005*temp_char)*r0n*(an+2*bn*temp_char);

        if(fabs(bl) - blad_dp >0)
        {
            sprintf(ok,"   ZLY   ");
            m_n0Key = 0;
        }
        else
        sprintf(ok, "  DOBRY  ");
        sprintf(wyn,"|  %-4d|  %-4.01f |%-5.21f |%-5.21f |",i+1,tn,rez,rn);
```

```
        wyniki += wyn;
        if(fabs(bl)<1.0E3)  {
            sprintf(wyn,"   %-9.3lf|",bl);
            wyniki += wyn;
        }
        else  {
            sprintf(wyn,"   error    |");
            wyniki += wyn;
        }
        sprintf(wyn,"  %-6.2lf |%s|\r\n",blad_dp,ok);
        wyniki += wyn;
        sprintf(wyn,  "|------|-------|-------|-------|------------|---------|---------|\r\n");
        wyniki += wyn;
    }

}

*/

// DOGRANA z komput   TG


void CPomiar::SprawdzCzuj1()
{
    CString wyniki_new;
    wyniki = wyniki_new;
    sprintf(ok,"   ZLY   ");
    m_nOKey =1;
    double temp_char, rez, tn, rn,dr, bl, blad_dp;

    &  liczRAB( m_d2g, &m_dRg,&m_dAg,&m_dBg);


    sprintf(wyn,"\r\n                    TABELA WYNIKOW CZUJNIKA WODY GORACEJ ");
    wyniki += wyn;
    sprintf(wyn,"\r\n                |------|-------|--------|--------|----------|-----------|---------|---
------|\r\n");
    wyniki += wyn;
    sprintf(wyn,  "                | Lp. |   t   | Robl   | Rnorm  |    dR    |    Et     |   Edop  | oce
na  |\r\n");
    wyniki += wyn;
    sprintf(wyn,  "                |------|-------|--------|--------|----------|-----------|---------|-----
----|\r\n");wyniki += wyn;
    sprintf(wyn,  "                |  -   |  °C   |  om    |  om    |    om    |    °C     |   °C    |
    |\r\n");wyniki += wyn;
    sprintf(wyn,  "                |------|-------|--------|--------|----------|-----------|---------|-----
----|\r\n");wyniki += wyn;



    for(int i=0; i<=6; i++)
    {

        temp_char= tmin + i * ((tmax-tmin)/5);

        if (i==6)   {

          double    ra0 = r_n*an;
          double    ra1= m_dRg*m_dAg;
          double rb1=m_dRg*m_dBg;
          double rb0=r_n*bn;
          double      rb=rb1 - rb0;
          double      ra=ra0 - ra1;
          temp_char= (ra) / (2 *(rb));
        }
    if(temp_char <= tmax && temp_char >= tmin) {
        rez = rezyst(temp_char,m_dRg,m_dAg,m_dBg);
        TempNorma( rez, &tn);
        rn=RezNorma(temp_char);
        dr=rez-rn;
        bl = tn - temp_char;
        blad_dp = 2.0;
        // (0.30+0.005*temp_char)*r0n*(an+2*bn*temp_char);
        if(fabs(bl) - blad_dp >0)
        {
            sprintf(ok,"   ZLY   ");
            m_nOKey = 0;
        }
        else
        sprintf(ok, "  DOBRY  ");
        sprintf(wyn,"                |  %-4d|  %-4.0lf |%-8.3lf|%-8.3lf|%+8.2e|",i+1,temp_char,rez,rn,dr);
```

```
        wyniki += wyn;
        if(fabs(bl)<1.0E3)  { /* 1.0E3 */
            sprintf(wyn,"%+11.2e|",bl);
            wyniki += wyn;
        }
        else  {
            sprintf(wyn,"   error    |");
            wyniki += wyn;
        }
        sprintf(wyn,"  %-6.2lf |%s|\r\n",blad_dp,ok);
        wyniki += wyn;
        sprintf(wyn,  "              |------|-------|--------|--------|----------|-----------|---------|-
--------|\r\n");
        wyniki += wyn;
    }
   }

}
////////////////////////////////////
void CPomiar::SprawdzCzuj2()
{

    double temp_char, rez, tn, rn,dr, bl, blad_dp;

    ObliczRAB( m_d2z, &m_dRz,&m_dAz,&m_dBz);


    sprintf(wyn,"\r\n                  TABELA WYNIKOW CZUJNIKA WODY ZIMNEJ ");
    wyniki += wyn;
        ntf(wyn,"\r\n              |------|-------|--------|--------|----------|-----------|---------|---
-----  ;\r\n");
    wyniki += wyn;
    sprintf(wyn,  "               |  Lp. |   t    | Robl   | Rnorm  |    dR    |    Et     |   Edop  |  oce
na  |\r\n");
    wyniki += wyn;
    sprintf(wyn,  "               |------|-------|--------|--------|----------|-----------|---------|-----
----|\r\n");wyniki += wyn;
    sprintf(wyn,  "               |  -   |  °C   |  om    |  om    |    om    |    °C     |   °C    |
    |\r\n");wyniki += wyn;
    sprintf(wyn,  "               |------|-------|--------|--------|----------|-----------|---------|-----
----|\r\n");wyniki += wyn;


        for(int i=0; i<=6; i++)
    {

        temp_char= tmin + i * ((tmax-tmin)/5);

        if (i==6)   {

            double  ra0 = r_n*an;
            double  ra1= m_dRz*m_dAz;
            double rb1=m_dRz*m_dBz;
            double rb0=r_n*bn;
               double      rb=rb1 - rb0;
               double      ra=ra0 - ra1;
            temp_char= (ra) / (2 *(rb));

        }

      if(temp_char <= tmax && temp_char >= tmin) {

        rez = rezyst(temp_char,m_dRz,m_dAz,m_dBz);
        TempNorma( rez, &tn);
        rn=RezNorma(temp_char);
        dr=rez-rn;
        bl = tn - temp_char;
        blad_dp = 2.0;
// (0.30+0.005*temp_char)*r0n*(an+2*bn*temp_char);
        if(fabs(bl) - blad_dp >0)
        {
            sprintf(ok,"   ZLY   ");
            m_nOKey = 0;
        }
        else
        sprintf(ok, "  DOBRY  ");
        sprintf(wyn,"               |  %-4d|  %-4.0lf |%-8.3lf|%-8.3lf|%+8.2e|",i+1,temp_char,rez,rn,dr);
        wyniki += wyn;
        if(fabs(bl)<1.0E3)  { /* 1.0E3 */
            sprintf(wyn,"%+11.2e|",bl);
            wyniki += wyn;
```

```
        }
        else  {
            sprintf(wyn,"   error    |");
            wyniki += wyn;
        }
        sprintf(wyn,"  %-6.2lf |%s|\r\n",blad_dp,ok);
        wyniki += wyn;
        sprintf(wyn,  "               |------|-------|--------|--------|----------|----------|---------|-
--------|\r\n");
        wyniki += wyn;
     } //if
    }  // for

}


//////////////////////////////////////////
//////////////////////////////////////////
void CPomiar::ObliczPary()
{


double temp_char;
int poz = 0;

    double tt1, tt2, dt, bl, blad_dp;
    double rez, tg, tz;
    sprintf(wyn,"\r\n                  TABELA WYNIKOW PARY CZUJNIKOW      ");
    wyniki += wyn;
    sprintf(wyn,"\r\n              |------|-------|-------|-------|------------|---------|---------|\r\n"
);
    wyniki += wyn;
    sprintf(wyn,  "               |  Lp. |  t1   |   t2  | t1-t2 |     Elt    |   Edop  |  ocena  |\r\n");
    wyniki += wyn;
    sprintf(wyn,  "               |------|-------|-------|-------|------------|---------|---------|\r\n");
wyniki += wyn;
    sprintf(wyn,  "               |      |  °C   |  °C   |  °C   |      %%    |    %%   |         |\r\n"
);wyniki += wyn;
    sprintf(wyn,  "               |------|-------|-------|-------|------------|---------|---------|\r\n");
wyniki += wyn;


//////////////////////////////////////////
    for(int i=0; i<6; i++)
    {
        temp_char= tmin + i * (((80-dt_min)-tmin)/5);
        tt1 = temp_char + dt_min;
        rez=  rezyst(tt1,m_dRg,m_dAg,m_dBg);
        TempNorma( rez, &tg);
//////////////
        tt2= temp_char ;
        rez = rezyst(tt2,m_dRz,m_dAz,m_dBz);
        TempNorma( rez, &tz);
        dt=tt1-tt2;
        blad_dp= (0.5 + 3*(dt_min/dt));
        bl =100*(dt - (tg-tz))/(tg-tz);
        if(fabs(bl) - blad_dp >0)
        {
            sprintf(ok,"   ZLY   ");
            m_nOKey = 0;
        }
        else
        sprintf(ok, "  DOBRY  ");
        ++poz;
        sprintf(wyn,"               |  %-4d|   %-4.0lf|   %-4.0lf|   %-4.0lf|",poz,tt1,tt2,dt);
        wyniki += wyn;
        if(fabs(bl)<1.0E3)  { /* 1.0E3 */
            sprintf(wyn,"   %-9.5lf|",bl);
            wyniki += wyn;
        }
        else  {
            sprintf(wyn,"   error    |");
            wyniki += wyn;
        }
        sprintf(wyn,"  %-6.3lf |%s|\r\n",blad_dp,ok);
        wyniki += wyn;
        sprintf(wyn,  "               |------|-------|-------|-------|------------|---------|---------|\r\
n");
        wyniki += wyn;
```

```
//////////////////////////////////////////
    for( i=0; i<6; i++)
    {

        temp_char= tmin + i * ((tmax-10-tmin)/5);
        tt1 = temp_char + 10;
        rez=  rezyst(tt1,m_dRg,m_dAg,m_dBg);
        TempNorma( rez, &tg);
///////////////
        tt2= temp_char ;
        rez = rezyst(tt2,m_dRz,m_dAz,m_dBz);
        TempNorma( rez, &tz);
        dt=tt1-tt2;
        blad_dp= (0.5 + 3*(dt_min/dt));
        bl =100*(dt - (tg-tz))/(tg-tz);
        if(fabs(bl) - blad_dp >0)
        {
            sprintf(ok,"   ZLY   ");
            m_nOKey = 0;
        }
        else
        sprintf(ok, "  DOBRY  ");
        ++poz;
        sprintf(wyn,"               |  %-4d|   %-4.01f|   %-4.01f|   %-4.01f|",poz,tt1,tt2,dt);
        wyniki += wyn;
        if(fabs(bl)<1.0E3)  { /* 1.0E3 */
            sprintf(wyn,"   %-9.51f|",bl);
            wyniki += wyn;
        }
        else  {
            sprintf(wyn,"   error    |");
            wyniki += wyn;
        }
        sprintf(wyn,"  %-6.31f |%s|\r\n",blad_dp,ok);
        wyniki += wyn;
        sprintf(wyn,  "               |------|-------|-------|-------|------------|---------|---------|\r\
n");
        wyniki += wyn;

    }
// tylko dla win95   wyrzucic dla win3_1
//  sprintf(wyn,  "\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n               |------|-------|-------|-------|----
--------|---------|---------|\r\n");
//      wyniki += wyn;
////////////////////////////////////////////////
    for( i=0; i<6; i++)
    {
        temp_char= tmin + i * ((tmax-20-tmin)/5);
        tt1 = temp_char + 20;
        rez=  rezyst(tt1,m_dRg,m_dAg,m_dBg);
        TempNorma( rez, &tg);
///,     '///////
        tt2= temp_char ;
        rez = rezyst(tt2,m_dRz,m_dAz,m_dBz);
        TempNorma( rez, &tz);
        dt=tt1-tt2;
        blad_dp= (0.5 + 3*(dt_min/dt));
        bl =100*(dt - (tg-tz))/(tg-tz);

        if(fabs(bl) - blad_dp >0)
        {
            sprintf(ok,"   ZLY   ");
            m_nOKey = 0;
        }
        else
        sprintf(ok, "  DOBRY  ");
        ++poz;
        sprintf(wyn,"               |  %-4d|   %-4.01f|   %-4.01f|   %-4.01f|",poz,tt1,tt2,dt);
        wyniki += wyn;
        if(fabs(bl)<1.0E3)  { /* 1.0E3 */
            sprintf(wyn,"   %-9.51f|",bl);
            wyniki += wyn;
        }
        else  {
            sprintf(wyn,"   error    |");
            wyniki += wyn;
        }
        sprintf(wyn,"  %-6.31f |%s|\r\n",blad_dp,ok);
        wyniki += wyn;
```

```
        sprintf(wyn,  "               |------|-------|-------|-------|------------|---------|---------|\r\
n");
        wyniki += wyn;

    }
////////////////////////////
        tt1 = tmin + dt_max;
        rez=  rezyst(tt1,m_dRg,m_dAg,m_dBg);
        TempNorma( rez, &tg);
////////////////
        tt2= tmin ;
        rez = rezyst(tt2,m_dRz,m_dAz,m_dBz);
        TempNorma( rez, &tz);
        dt=tt1-tt2;
        blad_dp= (0.5 + 3*(dt_min/dt));
        bl =100*(dt - (tg-tz))/(tg-tz);

        if(fabs(bl) - blad_dp >0)
        {
            sprintf(ok,"   ZLY   ");
            m_nOKey = 0;
        }
        else
        sprintf(ok, "  DOBRY  ");
        ++poz;
        sprintf(wyn,"               |  %-4d|   %-4.01f|   %-4.01f|   %-4.01f|",poz,tt1,tt2,dt);
        wyniki += wyn;
        if(fabs(bl)<1.0E3)  { /* 1.0E3 */
            sprintf(wyn,"   %-9.51f|",bl);
            wyniki += wyn;
        }
        else  {
            sprintf(wyn,"   error    |");
            wyniki += wyn;
        }
        sprintf(wyn,"  %-6.31f |%s|\r\n",blad_dp,ok);
        wyniki += wyn;
        sprintf(wyn,  "               |------|-------|-------|-------|------------|---------|---------|\r\
n");
        wyniki += wyn;

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

        tt1 = tmax ;
        rez=  rezyst(tt1,m_dRg,m_dAg,m_dBg);
        TempNorma( rez, &tg);
////////////////
        tt2= tmax - dt_max;
        rez = rezyst(tt2,m_dRz,m_dAz,m_dBz);
        TempNorma( rez, &tz);
        dt=tt1-tt2;
        blad_dp= (0.5 + 3*(dt_min/dt));
        bl =100*(dt - (tg-tz))/(tg-tz);

        if(fabs(bl) - blad_dp >0)
        {
            sprintf(ok,"   ZLY   ");
            m_nOKey = 0;
        }
        else
        sprintf(ok, "  DOBRY  ");
        ++poz;
        sprintf(wyn,"               |  %-4d|   %-4.01f|   %-4.01f|   %-4.01f|",poz,tt1,tt2,dt);
        wyniki += wyn;
        if(fabs(bl)<1.0E3)  { /* 1.0E3 */
            sprintf(wyn,"   %-9.51f|",bl);
            wyniki += wyn;
        }
        else  {
            sprintf(wyn,"   error    |");
            wyniki += wyn;
        }
        sprintf(wyn,"  %-6.31f |%s|\r\n",blad_dp,ok);
        wyniki += wyn;
        sprintf(wyn,  "               |------|-------|-------|-------|------------|---------|---------|\r\
n");
        wyniki += wyn;

////////////////////////////////////////
//TGGGG    max dla par       UWAGA 1=g goracy
////////////////////////////       2=z zimny
```

```
    double  ra0 = r_n*an;
            double  ra2= m_dRz*m_dAz;
            double rb2=m_dRz*m_dBz;
            double  ra1= m_dRg*m_dAg;
            double rb1=m_dRg*m_dBg;
            double rb0=r_n*bn;
               double      rb20=rb2 - rb0;
               double      ra21=ra2 - ra1;
               double      rb10=rb1 - rb0;
               double      ra02=ra0 - ra2;
               double      ra10=ra1 - ra0;
            temp_char=tmin +( (ra02-2*tmin*rb20) / (2 *rb20));



        tt2 = temp_char ;
            rez = rezyst(tt2,m_dRz,m_dAz,m_dBz);

        TempNorma( rez, &tz);

////////////////
        double l_y = rb0*ra21 + rb1*ra02 + rb2*ra10;
        double m_y = (-2)*rb10*rb20;
        double t_y = l_y/m_y;
        tt1= temp_char + t_y;
//TG
    if(tt1<=200 &&  t_y>2)  {
        rez=  rezyst(tt1,m_dRg,m_dAg,m_dBg);
        TempNorma( rez, &tg);
        dt=tt1-tt2;
        blad_dp= (0.5 + 3*(dt_min/dt));
        bl =100*(dt - (tg-tz))/(tg-tz);

        if(fabs(bl) - blad_dp >0)
        {
            sprintf(ok,"   ZLY   ");
            m_nOKey = 0;
        }
        else
        sprintf(ok, "  DOBRY  ");
        ++poz;
        sprintf(wyn,"               |  %-4d|   %-4.01f|   %-4.01f|   %-4.01f|",poz,tt1,tt2,dt);
        wyniki += wyn;
        if(fabs(bl)<1.0E3)  {
            sprintf(wyn,"   %-9.51f|",bl);
            wyniki += wyn;
        }
        else  {
            sprintf(wyn,"   error    |");
            wyniki += wyn;
        }
        sprintf(wyn,"  %-6.3lf |%s|\r\n",blad_dp,ok);
        wyniki += wyn;
        sprintf(wyn,  "               |------|-------|-------|-------|------------|---------|---------|\r\
n");
        wyniki += wyn;
    }
////////////////////////////////////
//TGGGG    max dla par



    if(m_nOKey)  {
    sprintf(wyn,"\r\n                  WYNIK  BADANIA   JEST   POZYTYWNY ");
    wyniki += wyn;
    }
    else  {
    sprintf(wyn,"\r\n                  WYNIK  BADANIA   JEST   NEGATYWNY ");
    wyniki += wyn;
    }
//  CParyDoc::printf("%s",wyniki);

}


////////////////////////////////////////////////////
void CPomiar::TempNorma(double rezys, double* tem)
{                           //wsp. rownania r=r0*(1+a*t+b*t*t)
```

```
 r_n=100.0;
    if(rezys > 500)
    r_n=500.0;
    if(rezys > 1000)
    r_n=1000.0;



    double del=r_n*r_n*an*an-4*r_n*bn*(r_n-rezys);//wyroznik trojmianu
    *tem = (sqrt(del) - r_n*an)/(2 *r_n*bn);
}
//////////////////////////////////////////////////////////////

double CPomiar::RezNorma(double t)
{
 double rezystancja = r_n*(1+an*t+bn*t*t);
 return rezystancja;
}
//////////////////////////////////////////////////////////////////

void CPomiar::TempWzorc(int nNrZlacza, double rezyst, double* temp)
{         //ze swiadectwa czujnika
double awz, bwz, r001, typ=1;
     // TG tymczasowo dla czujnika 25om zastapianego opornikiem 100om


    if( nNrZlacza==1) {
        awz=-0.00012347;
        bwz=-0.00002591;
        r001= 24.95867;


#ifdef DEMO
*temp = 50.0;
#endif
    }
    if( nNrZlacza==2) {
        awz=-0.00012347;
        bwz=-0.00002591;
        r001= 24.95867;

#ifdef DEMO
*temp = 100.0;
#endif
    }
    if( nNrZlacza==3) {
        awz=-0.00012347;
        bwz=-0.00002591;
        r001= 24.95867;

#ifdef DEMO
*tem  = 150.0;
#en
    }


#ifdef WZORC100om
r001=100.0;
#endif


    double  Wt = rezyst/r001;
    double  Wrt = Wt- awz*(Wt-1) - bwz*(Wt-1)*(Wt-1);
    double  ul = (Wrt-2.64)/1.64;
#ifndef DEMO
    *temp = d0+d1*ul+d2*ul*ul+d3*pow(ul,3)+d4*pow(ul,4)+d5*pow(ul,5)+d6*pow(ul,6)+d7*pow(ul,7)+d8*pow(ul,8
)+d9*pow(ul,9);
#endif
 }
//////////////////////////////////////////////////////////////
void CPomiar::PomiarIEC(double * wynik)
{
////////////////////////////////////////////////
    static int te;
    char instr[100];
    int a,dlug,przep;
    char oustr[100];
#ifndef DEMO
    if(te!=7)
    {
```

```
inicjal:    ini_karty();
    te=7;
//  }
    strcpy(oustr,"*RST");
    dlug=strlen(oustr);
    a=ieoutput(16,oustr,dlug);
Delay(10);
    a=ieoutput(16,oustr,dlug);


Delay(10);
        a=ieoutput(16,oustr,dlug);
/*
Delay(8);
        a=ieoutput(16,oustr,dlug);
*/

}

//TG przenioslem powyzszy nawias z za te=7zeby raz resetowal                a=ieoutput(16,oustr,dlug);
/*Delay(10);
                    a=ieoutput(16,oustr,dlug);

Delay(10);
                            a=ieoutput(16,oustr,dlug);
                            a=ieoutput(16,oustr,dlug);
=/
    strcpy(oustr,"*RST");
    dlug=strlen(oustr);
Dela  ( );
    a=ieoutput(16,oustr,dlug);
    strcpy(oustr,":conf:fres");
    dlug=strlen(oustr);
//Delay(1);
    a=ieoutput(16,oustr,dlug);
//Delay(1);
    strcpy(oustr,":sens:fres:nplc 8; dig 7.5; ocom on; aver:coun 2; tcon rep; stat on");
//strcpy(oustr,":sens:fres:nplc 8; dig 7.5");
    dlug=strlen(oustr);
//Delay(1);
    a=ieoutput(16,oustr,dlug);
//Delay(1);
    strcpy(oustr,":form:elem read");
    dlug=strlen(oustr);
//Delay(1);
    a=ieoutput(16,oustr,dlug);
//Delay(1);
    strcpy(oustr,":stat:meas:PTR 1; NTR 0");
    dlug=strlen(oustr);
//Delay(1);
    a=ieoutput(16,oustr,dlug);
//Delay(1);
    strcpy(oustr,":read?");
    dlug=strlen(oustr);
//Delay(1);
    a=ieoutput(16,oustr,dlug);
//Delay(1);
    a=ieenter(16,instr,100);
//Delay(1);
    strcpy(oustr,":stat:meas:even?");
    dlug=strlen(oustr);
//Delay(1);
    a=ieoutput(16,oustr,dlug);
Delay(0.1);
    a=ieenter(16,oustr,100);
Delay(0.1);
    przep=atoi(oustr);
    if (przep&&1==1)
//printf("przep = %d\n",przep);
            ;
    else
           ;
//printf("\n instr = %s\n",instr);
//printf("\n %8f Ohms", atof(instr));

    double wyn = atof(instr);
    if (wyn<20)
    goto inicjal;

    *wynik =wyn;
//  MessageBeep(20);
```

```
//Delay(4);

#endif

#ifdef DEMO
double czujnika = 1;
 if(typ_cz==1)
   czujnika=5;
if(typ_cz==2)
   czujnika=10;

*wynik = (100 + 19.4*m_nNrZlacza)* czujnika;
#endif
}




//////////////////////////////////////////////////////////
void CPomiar::PrzelaczNaCzujnik(WORD nNrZlacza, WORD nNrCzujnika)
{
    int adres = 0x300;
#ifndef DEMO
    WORD multplex = (nNrCzujnika) + 16 * (nNrZlacza-1);
    multplex = ~multplex;
unsigned char a=(unsigned char)multplex;

    _outp(adres,a);
    _start = CTime::GetCurrentTime();
//TG
    Delay(1);
    WORD  stan=_inp(adres);
    WORD maska = 1<<(nNrZlacza-1);
    stan=stan&maska;
    if(stan==0){
        sprintf(ok,"ZLY NR ZLACZA  ");
 // CParyDoc::printf(" BLAD MULTIPLEXERA zlacze nr  %d", nNrZlacza );
    }
#endif
}

void CPomiar::PobierzWartoscFromKeyb(WORD NrZlacz, WORD  NrCzujnika, double* dValue)
{
    CDlgFromKeyb dlg;
    char szLabel[200];
    sprintf(szLabel, "Wprowadź wartość ze złącza %d", NrZlacz);
    dlg.m_strLabel = szLabel;
    dlg.DoModal();
    *dValue = dlg.m_dValue;
}

//////////////////////////////////////////////////////////
void CPomiar::PobierzWartosci(WORD NrZlacz,WORD  NrCzujnika, double* dTemperatura, double* dRezystancja)
{
    int wzorcowy = 15;
    double dWynik;
    double dTemperatur;
    m_nNrZlacza=NrZlacz;
    m_nNrCzujnika= NrCzujnika;

//Ela - nie mierzyc gdy tylko temppotrzebna
    if(m_nNrCzujnika !=999)
    {
//      PrzelaczNaCzujnik(m_nNrZlacza, m_nNrCzujnika-1);
        czas_start = CTime::GetCurrentTime();
        Delay(1);

#ifdef FROM_KEYB
        PobierzWartoscFromKeyb(m_nNrZlacza, m_nNrCzujnika, &dWynik);
#else
        PomiarIEC(& dWynik);
#endif
        *dRezystancja = dWynik;
    }

    PrzelaczNaCzujnik(m_nNrZlacza, wzorcowy);

#ifndef DEMO
    czas_start = CTime::GetCurrentTime();
    Delay(1);
```

```
#endif
    PomiarIEC( & dWynik);
    Delay(1);
#ifdef WZORC_ZMIENNY

    TempNorma( dWynik, &dTemperatur);
    *dTemperatura = dTemperatur;
#endif

#ifndef WZORC_ZMIENNY
    TempWzorc(m_nNrZlacza, dWynik,  &dTemperatur);
    *dTemperatura = dTemperatur;
#endif
}

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



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

const char* CPomiar::Oblicz(struct D2 *sG, struct D2 *sZ, double tmini, double tmaxi, double tdmini, doubl
e tdmaxi,WORD typ_czuj)
{
    typ_cz = typ_czuj;
    tmin=tmini;
    tmax=tmaxi;
    dt_min=tdmini;
    dt_max=tdmaxi;
    r_n=100.0;
    if(typ_czuj==1)
    r_n=500.0;
    if(typ_czuj==2)
    r_n=1000.0;
    for( int i=0; i<3; i++)
    {
    m_d2g[i] = sG[i];
    m_d2z[i] = sZ[i];
    }
    SprawdzCzuj1();
    SprawdzCzuj2();
    ObliczPary();
    return wyniki;
}
```

```
// dlgfromk.cpp : implementation file
//

#include "stdafx.h"
#include "dlgfromk.h"

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

/////////////////////////////////////////////////////////////////////////////
// CDlgFromKeyb dialog


CDlgFromKeyb::CDlgFromKeyb(CWnd* pParent /*=NULL*/)
    : CDialog(CDlgFromKeyb::IDD, pParent)
{
    //{{AFX_DATA_INIT(CDlgFromKeyb)
    m_strLabel = "";
    m_dValue = 0;
    //}}AFX_DATA_INIT
}

void CDlgFromKeyb::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CDlgFromKeyb)
    DDX_Text(pDX, IDC_LABEL, m_strLabel);
    DDX_Text(pDX, IDC_VALUE, m_dValue);
    DDV_MinMaxDouble(pDX, m_dValue, 0., 1.e+021);
    //}}AFX_DATA_MAP
}

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


/////////////////////////////////////////////////////////////////////////////
// CDlgFromKeyb message handlers
```