/\*======================================================================

\*

\* PPMSDialog.cpp : implementation file

\*

\* This is the main file of the application. All the interesting stuff is

\* going on in here.

\*

\* Content:

\* - Help dialog

\* - FileExistsDialog

\* - CAboutDlg

\* - CPPMSDialog dialog, the main dialog

\*=======================================================================

\*/

#include "stdafx.h"

#include "afxdialogex.h"

#include "PPMSApp.h"

#include "PPMSDialog.h"

#include "CalcGap.h"

#include "constants.h"

#include "ppmsuser.h"

#include <cmath>

#include <iomanip>

#include <iterator>

#include <algorithm>

#include <iostream>

#include <fstream>

#include <sstream>

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

using namespace std;

/\*===========================================================================

\* Help dialog

\*===========================================================================

\*/

IMPLEMENT\_DYNAMIC(CHelpDialog, CDialogEx)

CHelpDialog::CHelpDialog(CString strHelpText, CWnd\* pParent /\*=NULL\*/)

: CDialogEx(CHelpDialog::IDD, pParent)

, m\_HelpText( strHelpText )

{

}

CHelpDialog::~CHelpDialog()

{

}

void CHelpDialog::DoDataExchange(CDataExchange\* pDX)

{

CDialogEx::DoDataExchange(pDX);

DDX\_Text(pDX, IDC\_HELPBOX, m\_HelpText);

DDX\_Control(pDX, IDC\_HELPBOX, m\_HelpCEdit);

}

BEGIN\_MESSAGE\_MAP(CHelpDialog, CDialogEx)

END\_MESSAGE\_MAP()

// CHelpDialog message handlers

BOOL CHelpDialog::OnInitDialog()

{

CDialogEx::OnInitDialog();

// TODO: Add extra initialization here

m\_font.CreatePointFont(90, \_T("Courier New"));

m\_HelpCEdit.SetFont(&m\_font);

m\_HelpCEdit.SetFocus();

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

// EXCEPTION: OCX Property Pages should return FALSE

}

/\*===========================================================================

\* FileExistsDialog dialog

\*===========================================================================

\*/

IMPLEMENT\_DYNAMIC(FileExistsDialog, CDialog)

FileExistsDialog::FileExistsDialog(CWnd\* pParent /\*=NULL\*/)

: CDialog(FileExistsDialog::IDD, pParent)

{

}

FileExistsDialog::~FileExistsDialog()

{

}

void FileExistsDialog::DoDataExchange(CDataExchange\* pDX)

{

CDialog::DoDataExchange(pDX);

}

BEGIN\_MESSAGE\_MAP(FileExistsDialog, CDialog)

ON\_BN\_CLICKED(IDCANCEL, &FileExistsDialog::OnBnClickedCancel)

ON\_BN\_CLICKED(IDC\_APPEND, &FileExistsDialog::OnBnClickedAppend)

ON\_BN\_CLICKED(IDC\_OVERWRITE, &FileExistsDialog::OnBnClickedOverwrite)

END\_MESSAGE\_MAP()

// FileExistsDialog message handlers

void FileExistsDialog::OnBnClickedCancel()

{

CDialog::OnCancel();

}

void FileExistsDialog::OnBnClickedAppend()

{

EndDialog(OUTPUT\_APPEND);

}

void FileExistsDialog::OnBnClickedOverwrite()

{

EndDialog(OUTPUT\_OVERWRITE);

}

/\*===========================================================================

\* CAboutDlg dialog (unused)

\*===========================================================================

\*/

CAboutDlg::CAboutDlg() : CDialogEx(CAboutDlg::IDD)

{

}

void CAboutDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialogEx::DoDataExchange(pDX);

}

BEGIN\_MESSAGE\_MAP(CAboutDlg, CDialogEx)

END\_MESSAGE\_MAP()

/\*===========================================================================

\*

\* CPPMSDialog dialog, the main dialog

\*

\*===========================================================================

\*/

CPPMSDialog::CPPMSDialog(CWnd\* pParent /\*=NULL\*/)

: CDialogEx(CPPMSDialog::IDD, pParent)

, m\_SaveFile(\_T(""))

, m\_DisplayStatus(\_T("Please set output files.\r\n"))

{

m\_hIcon = AfxGetApp()->LoadIcon(IDR\_MAINFRAME);

}

void CPPMSDialog::DoDataExchange(CDataExchange\* pDX)

{

CDialogEx::DoDataExchange(pDX);

DDX\_Text(pDX, IDC\_SAVEFILE, m\_SaveFile);

DDX\_Text(pDX, IDC\_DISPLAYSTATUS, m\_DisplayStatus);

DDX\_Control(pDX, IDC\_CHECK1, m\_CernoxSensor);

}

BEGIN\_MESSAGE\_MAP(CPPMSDialog, CDialogEx)

ON\_WM\_SYSCOMMAND()

ON\_WM\_PAINT()

ON\_WM\_QUERYDRAGICON()

ON\_BN\_CLICKED(IDC\_SAVEFILE\_BTN, &CPPMSDialog::OnBnClickedSavefileBtn)

ON\_BN\_CLICKED(IDOK, &CPPMSDialog::OnOK)

ON\_BN\_CLICKED(IDC\_HELP\_BTN, &CPPMSDialog::OnBnClickedHelpBtn)

ON\_BN\_CLICKED(IDC\_BTN\_CALCULATIONS, &CPPMSDialog::OnBnClickedBtnCalculations)

END\_MESSAGE\_MAP()

// CPPMSDialog message handlers

BOOL CPPMSDialog::OnInitDialog()

{

CDialogEx::OnInitDialog();

nReceivedAdvisories = 0;

// Add "About..." menu item to system menu.

// IDM\_ABOUTBOX must be in the system command range.

ASSERT((IDM\_ABOUTBOX & 0xFFF0) == IDM\_ABOUTBOX);

ASSERT(IDM\_ABOUTBOX < 0xF000);

CMenu\* pSysMenu = GetSystemMenu(FALSE);

if (pSysMenu != NULL)

{

BOOL bNameValid;

CString strAboutMenu;

bNameValid = strAboutMenu.LoadString(IDS\_ABOUTBOX);

ASSERT(bNameValid);

if (!strAboutMenu.IsEmpty())

{

pSysMenu->AppendMenu(MF\_SEPARATOR);

pSysMenu->AppendMenu(MF\_STRING, IDM\_ABOUTBOX, strAboutMenu);

}

}

// Set the icon for this dialog. The framework does this automatically

// when the application's main window is not a dialog

SetIcon(m\_hIcon, TRUE); // Set big icon

SetIcon(m\_hIcon, FALSE); // Set small icon

// TODO: Add extra initialization here

/\* Set GPIBdev with GPIB number PRIMARY\_ADDR\_OF\_DMM.

Initialization has to be successfull because GPIBdev has to be defined/valid.

The Average Exponent is set to 5.

\*/

GPIBdev = ibdev(BDINDEX, PRIMARY\_ADDR\_OF\_DMM, NO\_SECONDARY\_ADDR,

TIMEOUT, EOTMODE, EOSMODE);

if ( Ibsta() & ERR )

{

m\_DisplayStatus = "Unable to open capacitance bridge.\r\n";

m\_DisplayStatus += "Initialisation cancelled.\r\n";

m\_DisplayStatus += "Fix problem and restart program.\r\n";

AfxMessageBox(\_T("Unable to open capacitance bridge. Initialisation cancelled."), MB\_OK|MB\_ICONSTOP);

UpdateData(FALSE);

return false;

}

else

{

m\_DisplayStatus += "Capacitance bridge initialized.\r\n";

ibclr(GPIBdev); // Sends clear command to the capacitance bridge. Is this really necessary???

if ( Ibsta() & ERR )

m\_DisplayStatus += "Unable to clear device, capacitance bridge.";

}

Sleep(100); // wait to give the capacitance bridge some time

/\* Set the Average Exponent to 5\*/

ibwrt(GPIBdev, "AV 5", 4 );

if ( Ibsta() & ERR )

{

m\_DisplayStatus += "Unable to set Average Exponent.\r\n";

ErrorHandler( ERRID\_WRITE );

}

else

m\_DisplayStatus += "Average Exponent set to 5.\r\n";

/\* TODO: check if PPMS responds? Is this necessary? \*/

m\_hAdvisory = RegisterForAdvisories( this->m\_hWnd, WM\_ADVISORY, 0 );

m\_DisplayStatus += "Program ready to receive advisories.\r\n";

CheckDlgButton(IDC\_CHECK1, BST\_CHECKED); // Cernox sensor check box

// Load Cernox calibration file and calculate the interpolation coefficients.

// Cubic interpolation after Akima H.

// The coefficients are stored in the arrays a0, a1, a2 and a3.

// Set bCernoxLoaded to false - if the file can not be read

// - if temperature and reistance column have not the same size

//

// Format of input\_file containing the Cernox calibration data: 3 header lines and 2 columns.

// The temperature in the first column and the resistance of the Cernox sensor in the second.

bCernoxLoaded = false;

ifstream fin;

fin.open(PATH\_CERNOX);

if ( !fin )

m\_DisplayStatus += "Unable to read from Cernox calibration file.\r\n";

else

{

int ii;

string line;

for (ii=0; ii<3; ii++) // skip the 3 header files

getline(fin, line);

double x;

vector<double> vT, d, derivT;

while (fin>>x)

{

vT.push\_back(x);

if (fin>>x)

vR.push\_back(x);

}

fin.close();

if ( vT.size() != vR.size() )

m\_DisplayStatus += "Unable to read from Cernox calibration file.\r\n";

else

{

int size = vT.size();

// Calculate the estimate of the slope at each data point: d.

// d[i-1] = d(i) i.e.: d[0] = d(-1) or d[2] = d(1)

d.push\_back(0); // will be calculated below

d.push\_back(0); // will be calculated below

for(ii=0; ii<size-1; ii++)

d.push\_back((vT[ii+1]-vT[ii])/(vR[ii+1]-vR[ii]));

// Compute the slopes of the first two and the last two segments.

d[0] = 3\*d[2] - 2\*d[3];

d[1] = 2\*d[2] - d[3];

int k = d.size();

d.push\_back(2\*d[k-1] - d[k-2]);

d.push\_back(3\*d[k-1] - 2\*d[k-2]);

// Calculate the estimates of the derivative of vT: derivT.

for(ii=0; ii<size; ii++)

derivT.push\_back((abs(d[ii+3]-d[ii+2])\*d[ii+1] + abs(d[ii+1]-d[ii])\*d[ii+2])/(abs(d[ii+3]-d[ii+2])+abs(d[ii+1]-d[ii])));

// Compute the Akima coefficients a0, a1, a2 and a3.

double deltaR;

for(ii=0; ii<size-1; ii++)

{

deltaR = vR[ii+1] - vR[ii];

a0.push\_back(vT[ii]);

a1.push\_back(derivT[ii]);

a2.push\_back((3.\*vT[ii+1] - deltaR\*derivT[ii+1] - 3.\*vT[ii] - 2.\*deltaR\*derivT[ii])/pow(deltaR,2));

a3.push\_back((-2.\*vT[ii+1] + deltaR\*derivT[ii+1] + 2.\*vT[ii] + deltaR\*derivT[ii])/pow(deltaR,3));

}

bCernoxLoaded = true;

m\_DisplayStatus += "Cernox calibration table loaded.\r\n";

}

}

/\* string line;

ifstream CernoxFile;

CernoxFile.open(PATH\_CERNOX);

if ( !CernoxFile )

m\_DisplayStatus += "Unable to read from Cernox calibration file.\r\n";

else

{

// Get rid of header lines starting with '#' or '{'

// Problem of the old solution: The first line with numbers is skipped as well.

// Ugly solution: Reset to beginning of file and read it again.

int jj=0;

getline(CernoxFile, line);

// count number of header lines

while (line[0] == '#' || line[0] == '{')

{

jj++;

getline(CernoxFile, line);

}

CernoxFile.seekg(0, CernoxFile.beg);

for (int ii=0; ii<jj; ii++)

getline(CernoxFile, line);

copy(istream\_iterator<double>(CernoxFile), istream\_iterator<double>(), back\_inserter(vCernox)); // this line reads the file

CernoxFile.close();

m\_DisplayStatus += "Cernox calibration table loaded.\r\n";

}\*/

UpdateData(FALSE);

return true; // return true unless you set the focus to a control

}

void CPPMSDialog::OnSysCommand(UINT nID, LPARAM lParam)

{

if ((nID & 0xFFF0) == IDM\_ABOUTBOX)

{

CAboutDlg dlgAbout;

dlgAbout.DoModal();

}

else

{

CDialogEx::OnSysCommand(nID, lParam);

}

}

// If you add a minimize button to your dialog, you will need the code below

// to draw the icon. For MFC applications using the document/view model,

// this is automatically done for you by the framework.

void CPPMSDialog::OnPaint()

{

if (IsIconic())

{

CPaintDC dc(this); // device context for painting

SendMessage(WM\_ICONERASEBKGND, reinterpret\_cast<WPARAM>(dc.GetSafeHdc()), 0);

// Center icon in client rectangle

int cxIcon = GetSystemMetrics(SM\_CXICON);

int cyIcon = GetSystemMetrics(SM\_CYICON);

CRect rect;

GetClientRect(&rect);

int x = (rect.Width() - cxIcon + 1) / 2;

int y = (rect.Height() - cyIcon + 1) / 2;

// Draw the icon

dc.DrawIcon(x, y, m\_hIcon);

}

else

{

CDialogEx::OnPaint();

}

}

// The system calls this function to obtain the cursor to display while the user drags

// the minimized window.

HCURSOR CPPMSDialog::OnQueryDragIcon()

{

return static\_cast<HCURSOR>(m\_hIcon);

}

LRESULT CPPMSDialog::WindowProc(UINT message, WPARAM wParam, LPARAM lParam)

/\* Catches advisories from the PPMS \*/

{

if (message == (UINT)WM\_ADVISORY)

{

nReceivedAdvisories++;

ProcessAdvisory(message, wParam, lParam);

}

return CDialogEx::WindowProc(message, wParam, lParam);

}

void CPPMSDialog::ProcessAdvisory(UINT message, WPARAM wParam, LPARAM lParam)

/\* This is the main part of the program. It defines how advisories are processed.

\* 'wParam' is the advisory number set in the PPMS MultiVu Sequence.

\*/

{

string strCapacitance = "";

string strLoss = "";

string strTimestamp, strTemperature, strField, strResistance, strPressure;

size\_t findPos;

int ii, nEnd;

double CernoxTemperature, CernoxResistance;

// Pause the PPMS sequence until done processing (timeout = 1440 min)

strcpy( command, "HOLDOFF 1 5" ); // 1440 min is maybe too much. YES IT IS YOU MORON!!! 5 min is more than enough.

bytesToSend = strlen( command );

GpibSend( device, command, bytesToSend, response, responseSize,

&bytesRead, &pPpmsError, errorStr, errorSize, swapBytes,

mutexTimeout );

/\* Display the number of advisories received so far and reset the advisory counter \*/

if (wParam == 2)

{

CString strRA;

strRA.Format(\_T("%i"), nReceivedAdvisories);

m\_DisplayStatus += strRA;

m\_DisplayStatus += " advisories received. \r\n";

nReceivedAdvisories = 0;

}

/\* Make a single capacitance measurement and write the result to the output file \*/

if ( wParam == 1 )

{

/\* Request temperature, magnetic field and resisitivity (of the Cernox sensor) from the PPMS.

bit 1: temperature 2^1=2

bit 2: magnetic field 2^2=4

bit 8: user bridge channel 3 (ohm) 2^8=256

bit 19: sample space pressure 2^19=524288

Examples:

- Without sample space pressure:

262 = 2+4+256 -> Temperature(K), Magnetic Field(Oe), User Bridge Channel 3(Ohm)

- With sample space pressure:

524550 = 2+4+256+524288 -> Temperature(K), Magnetic Field(Oe), User Bridge Channel 3(Ohm), Sample Space Pressure (Volts/Unit)

- Without Cernox sensor (i.e. without User Bridge Channel 3)

524294 = 2+4+524288 -> Temperature(K), Magnetic Field(Oe), Sample Space Pressure \*/

if ( m\_CernoxSensor.GetCheck() == BST\_CHECKED )

strcpy(command, "GETDAT? 524550");

else

strcpy(command, "GETDAT? 524294");

bytesToSend = strlen( command );

GpibSend( device, command, bytesToSend, response, responseSize,

&bytesRead, &pPpmsError, errorStr, errorSize, swapBytes,

mutexTimeout );

/\* Get Capacitance and Loss from the AH2550A. \*/

ibwrt(GPIBdev, "SI", 2 ); // SI ... make a single capacitance measurement

if ( Ibsta() & ERR )

{

ErrorHandler( ERRID\_WRITE );

// resume the M6000 sequence

strcpy( command, "HOLDOFF -1" );

bytesToSend = strlen( command );

GpibSend( device, command, bytesToSend, response, responseSize,

&bytesRead, &pPpmsError, errorStr, errorSize, swapBytes,

mutexTimeout );

return;

}

ibrd(GPIBdev, ReadBuffer, ARRAYSIZEAH2550); // read response

if ( (Ibsta() & ERR) )

{

ErrorHandler( ERRID\_READ );

// resume the M6000 sequence

strcpy( command, "HOLDOFF -1" );

bytesToSend = strlen( command );

GpibSend( device, command, bytesToSend, response, responseSize,

&bytesRead, &pPpmsError, errorStr, errorSize, swapBytes,

mutexTimeout );

return;

}

/\* The processing of the return string of the AH2550A depends on its format.

Here I assumed the default format as given in the manual.

I also assume that the Loss is given in nanosiemens. \*/

ReadBuffer[Ibcnt()+1] = '\0';

//if ( ibcnt() >= 33 )

//{

string strCapacitance (ReadBuffer, 2, 13); // Gets 13 characters of ReadBuffer starting from position 2

string strLoss (ReadBuffer, 20, 13); // Gets 13 characters of ReadBuffer starting from position 20

//}

//else

// ErrorHandler( ERRID\_AH2550RESPONSE );

/\* Process response of the PPMS.

The format of GETDAT? is: DataFlag, TimeStamp, DataElement1, DataElement2, ... \*/

response[ bytesRead ] = 0;

string strResponse = response;

findPos = strResponse.find(',');

if ( findPos != string::npos )

{

strResponse.erase ( 0 , findPos + 1 );

findPos = strResponse.find(',');

if ( findPos != string::npos )

{

strTimestamp = strResponse.substr(0, findPos);

strResponse.erase ( 0 , findPos+ 1 );

}

else

{

strTimestamp = "";

strResponse.clear();

}

}

else

strResponse.clear();

if ( strResponse.size() > 0 )

{

/\* without sample space pressure \*/

//findPos = strResponse.find(','); // start counting at 1

//strTemperature = strResponse.substr( 0 , findPos ); //starts counting at 0

//strResponse.erase ( 0 , findPos+1 ); // also delete the ,

//findPos = strResponse.find(',');

//strField = strResponse.substr(0, findPos );

//strResponse.erase ( 0 , findPos+1 );

//findPos = strResponse.find(';');

//strResistance = strResponse.substr(0, findPos );

/\* with sample space pressure \*/

findPos = strResponse.find(','); // start counting at 1

strTemperature = strResponse.substr( 0 , findPos ); //starts counting at 0

strResponse.erase ( 0 , findPos+1 ); // also delete the ','

findPos = strResponse.find(',');

strField = strResponse.substr(0, findPos );

strResponse.erase ( 0 , findPos+1 );

// Interpolate the temperature of Cernox sensor using the coefficients a0, a1, a2 and a3.

// Requires a0, a1, a2, a3 and vR --> bCernoxLoaded needs to be true

// If the Cernox resitance is out of range CernoxTemperature is set to 0.

if ( m\_CernoxSensor.GetCheck() == BST\_CHECKED && bCernoxLoaded)

{

findPos = strResponse.find(',');

strResistance = strResponse.substr(0, findPos );

strResponse.erase ( 0 , findPos+1 );

CernoxResistance = atof(strResistance.c\_str());

bool InRange = true;

ii=0;

while( vR[ii] > CernoxResistance )

{

ii++;

if( ii==vR.size() )

InRange = false; // resistance out of range

}

if (ii == 0)

InRange = false; // resistance out of range

if ( InRange )

{

ii = ii-1;

double deltaX = (CernoxResistance-vR[ii]);

CernoxTemperature = a0[ii] + a1[ii]\*deltaX + a2[ii]\*pow(deltaX,2) + a3[ii]\*pow(deltaX,3);

}

else

CernoxTemperature = 0;

/\* Old method using a linear interpolation:

// Interpolate temperature of Cernox sensor

// Format of vCernox: T1 R1 T2 R2 ... starting at low T and high R

ii=1;

nEnd = vCernox.size()/2\*2;

while (vCernox[ii] > CernoxResistance && ii<nEnd )

ii+=2;

if ( ii >= nEnd )

{

m\_DisplayStatus += "Cernox resistance out of range. R=";

m\_DisplayStatus += strResistance.c\_str();

m\_DisplayStatus += "\r\n";

// add line to log file ?

}

CernoxTemperature = vCernox[ii-1] + (vCernox[ii-1]-vCernox[ii-3])/(vCernox[ii]-vCernox[ii-2]) \* (CernoxResistance - vCernox[ii]);

\*/

}

else

{

strResistance = "";

CernoxTemperature = 0;

}

findPos = strResponse.find(';');

strPressure = strResponse.substr(0, findPos );

}

else

{

ErrorHandler( ERRID\_PPMSRESPONSE );

strTemperature = ""; //whatever

strField = "";

strResistance = "";

CernoxTemperature = 0;

strPressure = "";

}

/\* Write the results to the output file. \*/

std::ofstream OutputFile( m\_SaveFile, ios\_base::app ); // maybe faster to create it right away??

if ( OutputFile.is\_open() )

{

//If this is changed make sure to also change the corresponding header in OnBnClickedSavefileBtn()

OutputFile << strField << ", " << CernoxTemperature << ", " << strCapacitance << ", " << strLoss << ", ";

OutputFile << strTemperature << ", " << strResistance << ", " << strTimestamp << ", " << strPressure << endl;

OutputFile.close();

}

else

{

m\_DisplayStatus += "Unable to write to measurement file.\r\n";

ErrorHandler( ERRID\_OUTPUTFILE );

}

}

/\* resume the PPMS sequence \*/

strcpy( command, "HOLDOFF -1" );

bytesToSend = strlen( command );

GpibSend( device, command, bytesToSend, response, responseSize,

&bytesRead, &pPpmsError, errorStr, errorSize, swapBytes,

mutexTimeout );

UpdateData( FALSE );

}

void CPPMSDialog::OnOK()

/\* Called on clicking the Close button in the main dialog \*/

{

// TODO: Add your specialized code here and/or call the base class

ibonl(GPIBdev, 0);

if (Ibsta() & ERR)

ErrorHandler( ERRID\_IBONL );

if ( !m\_hAdvisory )

UnregisterFromAdvisories( m\_hAdvisory );

CDialog::OnOK(); // Default

}

void CPPMSDialog::OnBnClickedSavefileBtn()

{

string strFileHeader;

// If this file header is changed make shure it is still handled correctly by "calculate gap".

strFileHeader = "#Field(Oe), CernoxT(K), Capacitance(pF), Loss(nS), PPMST(K), CernoxR(Ohm), timestamp(s), Pressure(Units/Volts)\r\n";

CFileDialog SaveDlg(FALSE, \_T(".txt"), \_T("DilatometerDefault.txt"), OFN\_NOCHANGEDIR, \_T("Text Files (\*.txt)|\*.txt|") \_T("All Files|\*.\*||"));

if ( SaveDlg.DoModal() == IDOK )

{

m\_SaveFile = SaveDlg.GetPathName();

fstream OutputFile(m\_SaveFile, ios\_base::in); // try to open file

if ( OutputFile.is\_open() )

{

/\* The file already exists. Ask the user if he wants to overwrite or append the file.

On Cancel replace m\_SaveFile with an empty string. \*/

OutputFile.close();

UpdateData(FALSE); // display file name (m\_SaveFile) in edit box

FileExistsDialog dlgFED;

OutPutMode = dlgFED.DoModal(); // return values: IDCANCEL, OUTPUT\_OVERWRITE or OUTPUT\_APPEND

if (OutPutMode == IDCANCEL )

{

m\_SaveFile = "";

UpdateData(FALSE);

return;

}

else if ( OutPutMode == OUTPUT\_OVERWRITE)

OutputFile.open(m\_SaveFile, ios\_base::out | ios\_base::trunc );

else if ( OutPutMode == OUTPUT\_APPEND)

{

OutputFile.open(m\_SaveFile, ios\_base::out | ios\_base::app );

strFileHeader = "";

}

}

else // the file does not exist yet, create it

OutputFile.open(m\_SaveFile, ios\_base::out );

if ( OutputFile.is\_open() )

{

OutputFile << strFileHeader;

OutputFile.close();

m\_DisplayStatus += "Measurement file ready.\r\n";

}

else

m\_DisplayStatus += "Unable to write to measurement file.\r\n";

}

UpdateData(FALSE);

}

void CPPMSDialog::ErrorHandler(int intErrorID)

/\* Appends strErrorMessage according to intErrorID to the log file.

\* Return value: None

\*/

{

//fstream OutputLogFile(PATH\_LOGFILE, ios\_base::out | ios\_base::app );

ofstream OutputLogFile;

OutputLogFile.open(PATH\_LOGFILE, ios::app);

if ( !OutputLogFile )

{

m\_DisplayStatus += "Unable to write to log file.\r\n";

UpdateData(FALSE);

return;

}

string strErrorMessage;

switch ( intErrorID ) {

case ERRID\_IBDEV:

strErrorMessage = "Unable to open AH2550A. ";

strErrorMessage += ReturnGPIBError();

break;

case ERRID\_IBCLR:

strErrorMessage = "Unable to clear AH2550A. ";

strErrorMessage += ReturnGPIBError();

break;

case ERRID\_IBONL:

strErrorMessage = "Unable to close AH2550A. ";

strErrorMessage += ReturnGPIBError();

break;

case ERRID\_WRITE:

strErrorMessage = "Unable to write to AH2550A.";

strErrorMessage += ReturnGPIBError();

break;

case ERRID\_READ:

strErrorMessage = "Unable to read from AH2550A.";

strErrorMessage += ReturnGPIBError();

break;

case ERRID\_AH2550RESPONSE:

strErrorMessage = "Wrong format of the return string of the AH2550A.";

break;

case ERRID\_PPMSRESPONSE:

strErrorMessage = "Unknown PPMS response.";

break;

case ERRID\_OUTPUTFILE:

strErrorMessage = "Unable to write to output file.";

break;

default:

strErrorMessage = "Unknown ErrorID.";

}

CTime theTime = CTime::GetCurrentTime();

OutputLogFile << "Time: " << CStringA(theTime.Format( "%d/%b/%Y %X" )) << ", ";

OutputLogFile << "ErrID: " << intErrorID << ", ";

OutputLogFile << "ErrorMsg: " << strErrorMessage << endl;

OutputLogFile.close();

}

string CPPMSDialog::ReturnGPIBError()

/\* Generates one error string from all the GPIB error functions

\* Return value: Said combined error string

\*/

{

stringstream ss;

ss << " Ibsta = " << Ibsta() << " Iberr = " << Iberr() << " " << ErrorMnemonic[Iberr()] ;

return ss.str();

}

void CPPMSDialog::OnBnClickedHelpBtn()

/\* Reads ReadMe1.txt and displays its content in a message box \*/

{

// TODO: Add your control notification handler code here

CString HelpText2 = \_T(" ");

ifstream helpfile("ReadMe1.txt");

if ( helpfile.is\_open() )

{

string line;

while (getline(helpfile, line) )

{

HelpText2 += line.c\_str();

HelpText2 += "\r\n";

}

helpfile.close();

}

else

HelpText2 += "Unable to open 'ReadMe1.txt'";

CHelpDialog dlgHelp(HelpText2);

dlgHelp.DoModal();

}

void CPPMSDialog::OnBnClickedBtnCalculations()

{

// Start CCalcGap

CCalcGap Dlg2(m\_SaveFile);

Dlg2.DoModal();

}