unit uRzdFrm;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, ExtCtrls, ComCtrls, DCL\_MYOWN, Buttons, Spin,

inifiles, posbase, Winpos\_ole\_TLB,

uWPProc, uRZDTareFrame, ucommonmath, uCommontypes, uwpopers, mathfunction,

pathutils, uPathMng, uEventList, UWPEvents, uMNK, u2DMath, uBtnListView,

uComponentServises,

udbobject,

uRZDBase,

ImgList,

VirtualTrees,

ActiveX,

uBaseObjService,

uVTServices,

Shellapi,

uEventTypes,

uLogFile,

uPointsList,

uAddRZDDataFrm, RpRender, RpRenderRTF, RpDefine,

pngimage, gifimg, jpeg,

RpRave, RvClass, RVCsStd,

ClipBrd, RpBase, RpFiler, RpRenderPDF, RpSystem, frxClass, frxRich, frxEditMemo,

frxExportRTF, frxExportPDF, frxExportHTML, frxExportODF;

type

cGist = class

public

start: integer;

shift: integer;

proc: boolean;

lvl: double;

noise: double;

end;

TVec4 = record

i1, i2, i3, i4: double;

end;

// структура описания создания вектора Ei

TEi = record

sname, // имя датчика по которому построен

fname, // имя силы по которому построен

// ссылка на узлы в дереве WP

nCloud, nPoly: iwpnode;

cloud, poly: iwpSignal;

fmax: double;

// значение компонента вектора f

ei: double;

polyC: array of double;

result: boolean;

end;

smatrix = record

mf: string;

vt, gt, s1, s2, s3, s4: string;

// интервал времени

t1t2: point2d;

end;

TRZDMatrix = class

public

date:string;

m\_active: boolean;

vtmax: array [0 .. 3] of double;

gtmax: array [0 .. 3] of double;

// 3 последних восстановленных сигнала используется при построении ф-ции сравнения

lastn1, lastn2, lastn3:string;

n1, n2, n3:iwpnode;

// сигналы по которым посчитана матрица

sensors: array [0 .. 3] of smatrix;

// если True то используются абсолютные имена датчиков иначе ищем датчики в замере по префиксам

useSNames: boolean;

// найденые имена по префиксам

fs1, fs2, fs3, fs4: string;

// имена датчиков для которых строится матрица предустановленные

ls1, ls2, ls3, ls4: string;

// участок для которого строится матрица

region: string;

// сечение для которго строится матрица

cut: integer;

// матрица коэфициентов

m: d2array;

// степень полинома

poly: integer;

public

// возвращает имя в ини файле

function GetIniName: string;

procedure UpdateSNames(mf: string); overload;

procedure UpdateSNames(src: csrc); overload;

function ApplyMatrix(src: csrc): boolean;

function getRow(i: integer): string;

function Infostr: string;

// возвращает приставку к имени сигнала вида \_1\_reg\_01

function GetResPostfix: string;

constructor create;

protected

function gets1: string;

function gets2: string;

function gets3: string;

function gets4: string;

function gets(index: integer): string;

// меняет ды1ююды4 (только для редактирования пользователем!!!)

procedure sets(index: integer; str: string);

public

// установка обновляет только имена по поиску

property s1: string read gets1 write fs1;

property s2: string read gets2 write fs2;

property s3: string read gets3 write fs3;

property s4: string read gets4 write fs4;

property s[index: integer]: string read gets write sets;

end;

TRZDFrm = class(TForm)

PathPanel: TPanel;

Splitter1: TSplitter;

BaseFolderPanel: TPanel;

BaseFolderLabel: TLabel;

BaseFolderEdit: TEdit;

BaseFolderBtn: TButton;

VertPanel: TPanel;

sectionCB: TComboBox;

RegionCB: TComboBox;

RegionLabel: TLabel;

sectionLabel: TLabel;

Splitter2: TSplitter;

TareNamePanel: TPanel;

LabelsPanel: TPanel;

VFLabel: TLabel;

HFLabel: TLabel;

S1Label: TLabel;

VCPathLabel: TLabel;

S2Label: TLabel;

S3Label: TLabel;

S4Label: TLabel;

MatrixPanel: TPanel;

MatrixGB: TGroupBox;

G11: TFloatEdit;

G21: TFloatEdit;

G31: TFloatEdit;

G41: TFloatEdit;

G42: TFloatEdit;

G32: TFloatEdit;

G22: TFloatEdit;

G12: TFloatEdit;

G43: TFloatEdit;

G33: TFloatEdit;

G23: TFloatEdit;

G13: TFloatEdit;

ExpMatrixBtn: TButton;

ImpMatrixBtn: TButton;

ApplyGB: TGroupBox;

ApplyBtn: TButton;

Hout\_D: TFloatEdit;

Hout\_DLabel: TLabel;

EvalGB: TGroupBox;

ExportMatrixPath: TEdit;

ImportMatrixPath: TComboBox;

SaveDialog1: TSaveDialog;

ExportMatrixBtn: TButton;

ImportMatrixBtn: TButton;

UseMFmatrix: TCheckBox;

useSignalPrefix: TCheckBox;

TareFramePageControl: TPageControl;

VC\_Page: TTabSheet;

H\_Page: TTabSheet;

Hout\_Page: TTabSheet;

Hin\_Page: TTabSheet;

LinkEvalWP: TCheckBox;

Splitter3: TSplitter;

DrawGraphBtn: TSpeedButton;

StayOnTopBtn: TSpeedButton;

CutSE: TSpinEdit;

RegionSE: TSpinEdit;

ChangeTestByCutCB: TCheckBox;

JournalPageControl: TPageControl;

JournalTS: TTabSheet;

MatrixTS: TTabSheet;

JournalLB: TListBox;

Panel1: TPanel;

Splitter4: TSplitter;

Matrix\_CutSE: TSpinEdit;

RegLabel: TLabel;

CutLabel: TLabel;

MatrixLV: TBtnListView;

mat\_UseNamesCB: TCheckBox;

MatrixRegSE: TEdit;

Mat\_S1CB: TComboBox;

Mat\_S2CB: TComboBox;

Mat\_S3CB: TComboBox;

Mat\_S4CB: TComboBox;

Mat\_S1label: TLabel;

mat\_S2label: TLabel;

mat\_s4label: TLabel;

mat\_s3Label: TLabel;

LoadBtnPanel: TPanel;

CreateMatrixBtn: TButton;

SelectIntervalCursorBtn: TButton;

SelectIntervalGraphBtn: TButton;

ImageList\_16: TImageList;

ImageList\_32: TImageList;

VTree1: TVTree;

Splitter5: TSplitter;

UpdatePathBtn: TSpeedButton;

Timer1: TTimer;

FmaxEdit: TFloatEdit;

FmaxLabel: TLabel;

VC\_Frame: TRZDTareFrame;

H\_Frame: TRZDTareFrame;

Hin\_Frame: TRZDTareFrame;

Hout\_Frame: TRZDTareFrame;

NullBtn: TButton;

TabSheet1: TTabSheet;

GistLV: TBtnListView;

Splitter6: TSplitter;

Panel2: TPanel;

GistStart: TSpinEdit;

GistShiftSE: TSpinEdit;

GistStartLabel: TLabel;

GistShiftLabel: TLabel;

AddGist: TSpeedButton;

DelGist: TSpeedButton;

GistDxFE: TFloatEdit;

GistDxLabel: TLabel;

Label2: TLabel;

GistTrigFE: TFloatEdit;

GistNoiseFE: TFloatEdit;

Label3: TLabel;

GistProcCB: TCheckBox;

NullCb: TCheckBox;

DelMatrixBtn: TSpeedButton;

GtmaxLabel: TLabel;

GtmaxEdit: TFloatEdit;

DelMatrixOnExpCB: TCheckBox;

frxReport1: TfrxReport;

frxRTFExport1: TfrxRTFExport;

PAlfaLabel: TLabel;

PAlfaEdit: TFloatEdit;

frxPDFExport1: TfrxPDFExport;

FmaxAbsLabel: TLabel;

GtmaxAbsLabel: TLabel;

FmaxAbsFE: TFloatEdit;

GtmaxAbsFE: TFloatEdit;

Panel3: TPanel;

TestLabel: TLabel;

PolyLabel: TLabel;

Hin\_EvalBtn: TSpeedButton;

Label1: TLabel;

TestsFolderLabel: TLabel;

TestPath: TComboBox;

PolySE: TSpinEdit;

TestPathBtn: TButton;

PolyCount: TIntEdit;

TestsFolderE: TEdit;

ReportCB: TCheckBox;

OpenReportCB: TCheckBox;

Reptype: TRadioGroup;

ExtImagesCB: TCheckBox;

frxHTMLExport1: TfrxHTMLExport;

FltPanel: TPanel;

TrendPortionILabel: TLabel;

TrendPortionFLabel: TLabel;

LPFOrderLabel: TLabel;

LPFfreqLabel: TLabel;

LPFCB: TCheckBox;

TrendPortionIE: TIntEdit;

TrendPortionFE: TFloatEdit;

TrendCB: TCheckBox;

LPFOrder: TIntEdit;

LPFfreq: TFloatEdit;

OpenDialog1vista: TFileOpenDialog;

OpenDialog1: TOpenDialog;

procedure BaseFolderBtnClick(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure BaseFolderEditChange(Sender: TObject);

procedure RegionCBChange(Sender: TObject);

procedure FormShow(Sender: TObject);

procedure sectionCBChange(Sender: TObject);

procedure useSignalPrefixClick(Sender: TObject);

procedure CreateMatrixBtnClick(Sender: TObject);

procedure TestPathBtnClick(Sender: TObject);

procedure Hin\_EvalBtnClick(Sender: TObject);

procedure DrawGraphBtnClick(Sender: TObject);

procedure StayOnTopBtnClick(Sender: TObject);

procedure CutSEChange(Sender: TObject);

procedure TrendPortionFEChange(Sender: TObject);

procedure TrendPortionIEChange(Sender: TObject);

procedure Hin\_FramePathChange(Sender: TObject);

procedure H\_FramePathChange(Sender: TObject);

procedure Hout\_FramePathChange(Sender: TObject);

procedure VC\_FramePathChange(Sender: TObject);

procedure ExpMatrixBtnClick(Sender: TObject);

procedure ImpMatrixBtnClick(Sender: TObject);

procedure ImportMatrixPathChange(Sender: TObject);

procedure ExportMatrixPathChange(Sender: TObject);

procedure MatrixLVChange(Sender: TObject; Item: TListItem;

Change: TItemChange);

procedure UseMFmatrixClick(Sender: TObject);

procedure TestPathChange(Sender: TObject);

procedure ApplyBtnClick(Sender: TObject);

procedure Hin\_FrameSelectIntervalCursorBtnClick(Sender: TObject);

procedure SelectIntervalCursorBtnClick(Sender: TObject);

procedure SelectIntervalGraphBtnClick(Sender: TObject);

procedure ExportMatrixBtnClick(Sender: TObject);

procedure RegionSEChange(Sender: TObject);

procedure TestsFolderEChange(Sender: TObject);

procedure VTree1DragOver(Sender: TBaseVirtualTree; Source: TObject;

shift: TShiftState; State: TDragState; Pt: TPoint; Mode: TDropMode;

var Effect: integer; var Accept: boolean);

procedure VTree1DragDrop(Sender: TBaseVirtualTree; Source: TObject;

DataObject: IDataObject; Formats: TFormatArray; shift: TShiftState;

Pt: TPoint; var Effect: integer; Mode: TDropMode);

procedure UpdatePathBtnClick(Sender: TObject);

procedure Timer1Timer(Sender: TObject);

procedure NullBtnClick(Sender: TObject);

procedure AddGistClick(Sender: TObject);

procedure DelGistClick(Sender: TObject);

procedure DelMatrixBtnClick(Sender: TObject);

procedure ReportCBClick(Sender: TObject);

public

m\_pageIndex:integer;

// log: cLogFile;

f\_updFolder: string;

ThreadCount: integer;

// префикс для D1, D2,D3,D4,Fh,Fv

m\_D1Pref, m\_D2Pref, m\_D3Pref, m\_D4Pref, m\_FhPref, m\_FvPref,

// префикс для участка

m\_regionPref,

// префикс для заезда

m\_TestPref,

// префикс для сечения

m\_sectionPref: string;

// текущий источник сигнала для обработки

m\_src,

// источники соответствующие нагружениям

vc\_src, h\_src, hout\_src, hin\_src: csrc;

// секция ини файла в которую сохраняются настройки и путь к ини файлу

m\_section, m\_cfg, m\_VCPref, m\_H\_Pref, m\_Hin\_Pref, m\_Hout\_Pref: string;

m\_wpMng: cwpobjmng;

// Графика для выбора участков времен по курсорам

graph: array [0 .. 3] of tgraphstruct;

// вектора для расчета матрицы силы

e1, e2, e3, e4: array [0 .. 3] of TEi;

// зависимость верт от гор тяги в втором эксперименте

m\_F2zFy, m\_F2zFy\_poly: iwpSignal;

m\_F2zFyValue, m\_F2zFyMaxX: double; // значение полинома Fz(Fy)

e1norm, e2norm, e3norm, e4norm: TVec4;

// матрица по векторам ei и результирующая матрица

E, G: d2array;

m\_matrix: TRZDMatrix;

MatrixList: tlist;

// база данных

m\_DB: cRZDbase;

f\_updDBCS: TRTLCriticalSection;

f\_needUpdateDB: boolean;

// m\_s1, m\_s2, m\_s3, m\_s4, m\_Vt, m\_Gt:iwpsignal;

protected

procedure cleargist;

procedure addgistproc(start, shift: integer; lvl, noise: double;

proc: boolean);

procedure CheckMatrix(m: TRZDMatrix; src: csrc);

function checkSignalInterval(s: iwpSignal; interval: point2d): boolean;

procedure UpdateTestsFolder;

procedure ShowG(p\_g: d2array);

procedure ImpMatrix(str: string);

procedure ExportMatrix(m: TRZDMatrix; ifile: string);

procedure FillMatrixSensorsCB;

procedure ClearMList;

procedure addMatrix(m: TRZDMatrix);

procedure ShowMatrixList;

// получить из ShowMatrixList статусы активности матриц

// procedure getActStatusForM;

// обновить размер порции для филдьтра "тренд"

procedure UpdateIPortion;

procedure UpdateFPortion;

// считывает максимальную частоту опроса среди всех датчиков из ини файла

function maxFS(s1, s2, s3, s4, vt, gt, mera: string): double;

function GetregionPath: string;

Procedure GetNumFromSectionCB;

// загрузка источника данных по выбранному пути

function LoadSrc(str: string): csrc;

procedure LoadVCSrc;

procedure LoadHSrc;

procedure LoadHinCSrc;

procedure LoadHoutSrc;

procedure SaveToIniTareFrame(ifile: tinifile; fr: TRZDTareFrame);

procedure LoadfromIniTareFrame(ifile: tinifile; fr: TRZDTareFrame);

// Заполняем комбобоксы доступными сигналами

procedure FillCB(V\_cb, H\_cb, D1\_cb, D2\_cb, D3\_cb, D4\_cb: TComboBox;

merafile: string);

procedure FillVCCB;

procedure FillHCB;

procedure FillHoutCB;

procedure FillHinCB;

// проверки каталогов

procedure CheckBaseFolder;

function checkImportFolder: boolean;

// считываем список участков

procedure ReadRegionFolders;

procedure ReadSegments(RegionPath: string);

// заполняем пути к сигналам калибровки

procedure ReadTests(segmentpath: string);

procedure createEvents;

procedure destroyEvents;

procedure OnAddLine(Sender: TObject);

procedure OnDestroySignal(Sender: TObject);

procedure OnDestroySrc(Sender: TObject);

function EvalEi(interval: point2d; s, V: iwpSignal; name: string): TEi;

procedure EvalFzFy(interval: point2d; Fz, Fy: iwpSignal; name: string);

procedure LinkEiSignals(E: array of TEi);

function GetAbsRelFmax(fmax, rel, p\_abs:double):double;

// построить сравннеие двух сигналов по формуле

// s1-s2/f(base,f);

function BuildCompare(s1, base: iwpSignal; fmax, rel: double; resfolder: string; var res1, res2:iwpsignal): iwpSignal;

// shift - задает первый импульс с которого начинается гистограмма, tab пропуск между импульсами

function BuildGistogram(s: iwpSignal; dx: double; folder: string): iwpSignal; overload;

function BuildGistogram(s: iwpSignal; dx, lvl, noise: double; start, shift: integer; folder, gistname: string): iwpSignal; overload;

procedure BuildReport(vt, gt: iwpSignal; m: TRZDMatrix);

procedure addRepPageFR(g:tgraphstruct; date:string; cut:string; testcaption:string;findcalibr:boolean; filepath:string;var pageIndex:integer);

procedure addEiJournal(p\_name: string; E: array of TEi; enorm: TVec4);

// построение рез матрицы

function buildG: d2array;

function getformmatrix: d2array;

procedure ApplyMatrix(s1, s2, s3, s4: iwpSignal; p\_g: d2array;

srcpath: string; sname: string; m:TRZDMatrix);

// возвращает СОЗДАННЫЙ ВНОВЬ стринг лист содержащий имена датчиков соотносящиеся с сечением

function GetSensorsNameWithPref(src: csrc; sect: integer): tstringlist;

overload;

function GetSensorsNameWithPref(mf: string; sect: integer): tstringlist;

overload;

// Загрузка данных для построения матрицы

function PrepareDataforMatrix: boolean;

// получить по индексу комбобокс для редактирования свойтсв матрицы

function GetSensCB(i: integer): TComboBox;

// отобразить исходные сигналы для выборки сигналов по курсору

// возвращает hpage на котором отображены графики

function ShowTimeGraphs(p: integer; fr: TRZDTareFrame; src: csrc;

graphname: string): tgraphstruct;

function CheckSrc: boolean;

// получить сигнал с учетом фильтров

function GetSignal(src: csrc; sname: string; calibr: boolean;

var findSignal: boolean): iwpSignal;

// получить путь с учетом базового каталога

function gettestpath: string;

procedure createDB;

procedure UpdateDB; overload;

procedure UpdateDB(str: string); overload;

// перечитать каталоги базы данных

// procedure UpdateDB;

procedure copyFile(str: string; p: crzdpars);

// обновляем базу

procedure doUpdateBase(Sender: TObject);

procedure doUpdateBaseMessage(Sender: TObject);

procedure UpdateHandler(var Message: TMessage); message wm\_UpdateFolder;

public

function GetCalibrFolder: string;

function GetTestsFolder: string;

// filename - путь к КАТАЛОГУ с мера файлом

function testDir(filename: string): boolean;

public

procedure Init(p\_section, p\_cfg: string; p\_wpMng: cwpobjmng);

procedure Save;

procedure Load;

end;

var

RZDFrm: TRZDFrm;

def\_h:double;

def\_init:boolean;

c\_CommentDx: double=9;

c\_CommentDy: double=7;

v\_GistAxXScale:boolean=true;

v\_GistAxXRate:double=0.2;

v\_MaxF:double=10;

v\_bitScale:double=0.8;

const

// число значащих цифр

c\_digits = 4;

implementation

const

c\_Percent = 1;

c\_abs = 0;

{$R \*.dfm}

{ TRZDFrm }

procedure setGistXScale(hline:integer);

var

G: tgraphstruct;

min, max, range: double;

gist:iwpsignal;

p2:point2d;

s1,s2:widestring;

color:integer;

opt:integer;

begin

G := GetGraphStructByHLine(hline);

gist:=IWPGraphs(WP.GraphAPI).GetSignal(hline) as iwpsignal;

if v\_GistAxXScale then

begin

min:=strtofloat(gist.GetProperty('GistMin'));

max:=strtofloat(gist.GetProperty('GistMax'));

range:=max-min;

p2.x:=min-range\*v\_GistAxXRate;

p2.y:=max+range\*v\_GistAxXRate;

IWPGraphs(WP.GraphAPI).SetXMinMax(g.hgraph, p2.x, p2.y);

IWPGraphs(WP.GraphAPI).GetAxisOpt(G.hgraph, 0, opt, min, max, s1, s2, color);

IWPGraphs(WP.GraphAPI).SetAxisOpt(g.hgraph, 0, AXOPT\_RANGE, $FFFFFFFF, p2.x, p2.y, '1','1', 0);

IWPGraphs(WP.GraphAPI).GetAxisOpt(G.hgraph, 0, opt, min, max, s1, s2, color);

end

else

IWPGraphs(WP.GraphAPI).SetYAxisMinMax(G.haxis, 0, max \* 1.2);

end;

procedure TRZDFrm.BaseFolderBtnClick(Sender: TObject);

begin

OpenDialog1vista.filename:=BaseFolderEdit.Text;

OpenDialog1vista.Options := [fdoPickFolders, fdoForceFileSystem];

if OpenDialog1vista.Execute() then

begin

BaseFolderEdit.Text := OpenDialog1vista.filename;

end;

end;

procedure TRZDFrm.Init(p\_section, p\_cfg: string; p\_wpMng: cwpobjmng);

var

i, j: integer;

m: d2array;

d: double;

begin

if AddRZDDataFrm = nil then

begin

AddRZDDataFrm := tAddRZDDataFrm.create(nil);

end;

m\_section := p\_section;

m\_cfg := p\_cfg;

m\_wpMng := p\_wpMng;

MatrixList := tlist.create;

createEvents;

setlength(E, 3, 4);

end;

procedure TRZDFrm.LoadfromIniTareFrame(ifile: tinifile; fr: TRZDTareFrame);

var

subStr: string;

i: integer;

begin

// Путь к замеру для тарировки вертикально центрального нагружения

for i := 1 to length(fr.name) - 1 do

begin

if fr.name[i] = '\_' then

begin

subStr := copy(fr.Name, 1, i - 1);

break;

end;

end;

fr.NullFE1.FloatNum := IniReadFloatEx(ifile,m\_section, subStr + '\_' + 'Null1', 0);

fr.NullFE2.FloatNum := IniReadFloatEx(ifile,m\_section, subStr + '\_' + 'Null2', 0);

fr.T1FE.FloatNum := IniReadFloatEx(ifile,m\_section, subStr + '\_' + 'T1', 0);

fr.T2FE.FloatNum := IniReadFloatEx(ifile,m\_section, subStr + '\_' + 'T2', 0);

fr.Path.Text := ifile.readString(m\_section, subStr + '\_' + 'Path', '');

fr.VFCbox.Text := ifile.readString(m\_section, subStr + '\_' + 'VF', '');

fr.HFCbox.Text := ifile.readString(m\_section, subStr + '\_' + 'HF', '');

fr.S1Cbox.Text := ifile.readString(m\_section, subStr + '\_' + 'S1', '');

fr.S2Cbox.Text := ifile.readString(m\_section, subStr + '\_' + 'S2', '');

fr.S3Cbox.Text := ifile.readString(m\_section, subStr + '\_' + 'S3', '');

fr.S4Cbox.Text := ifile.readString(m\_section, subStr + '\_' + 'S4', '');

end;

procedure TRZDFrm.Load;

var

ifile: tinifile;

str: string;

i, ind, count: integer;

start, shift: integer;

noise, lvl: double;

b: boolean;

begin

ifile := tinifile.create(m\_cfg);

// параметр по которому вычисляется приведенная погрешность

v\_MaxF:=IniReadFloatEx(ifile, m\_section,'MaxF', 10);

FMaxEdit.FloatNum:=IniReadFloatEx(ifile, m\_section,'VtMaxRel', 0.1);

FmaxAbsFE.FloatNum:=IniReadFloatEx(ifile, m\_section,'VtMaxAbs', 1);

GtMaxEdit.FloatNum:=IniReadFloatEx(ifile, m\_section,'GtMaxRel', 0.1);

GtmaxAbsFE.FloatNum:=IniReadFloatEx(ifile, m\_section,'GtMaxAbs', 1);

ExtImagesCB.Checked:=ifile.ReadBool(m\_section, 'ExtImages', false);

LinkEvalWP.Visible:=ifile.ReadBool(m\_section, 'MatrixBtn', false);

ApplyBtn.Visible:=ifile.ReadBool(m\_section, 'MatrixBtn', false);

FltPanel.Visible:=ifile.ReadBool(m\_section, 'FltPanel', false);

ReportCB.Checked:=ifile.ReadBool(m\_section, 'GenReport', false);

OpenReportCB.Checked:=ifile.ReadBool(m\_section, 'OpenReport', false);

NullCB.Checked:=ifile.ReadBool(m\_section, 'NullCB', false);

reptype.ItemIndex:=ifile.ReadInteger(m\_section, 'RepType', 0);

c\_CommentDx:=IniReadFloatEx(ifile, m\_section,'CommentDx', 9);

c\_CommentDy:=IniReadFloatEx(ifile, m\_section,'CommentDy', 7);

v\_GistAxXRate:=IniReadFloatEx(ifile, m\_section,'GistAxXRate', 0.1);

v\_GistAxXScale:=ifile.ReadBool(m\_section, 'GistAxXScale', true);

v\_bitScale:=IniReadFloatEx(ifile, m\_section,'BitScale', 0.8);

// загружаем префиксы

m\_D1Pref := ifile.readString(m\_section, 'D1Prefix', 'NV');

if m\_D1Pref = '' then

m\_D1Pref := 'VN';

m\_D2Pref := ifile.readString(m\_section, 'D2Prefix', 'VV');

if m\_D2Pref = '' then

m\_D2Pref := 'NN';

m\_D3Pref := ifile.readString(m\_section, 'D3Prefix', 'NN');

if m\_D3Pref = '' then

m\_D3Pref := 'VV';

m\_D4Pref := ifile.readString(m\_section, 'D4Prefix', 'VN');

if m\_D4Pref = '' then

m\_D4Pref := 'NV';

m\_FhPref := ifile.readString(m\_section, 'FhPrefix', 'Gt');

if m\_FhPref = '' then

m\_FhPref := 'GT';

m\_FvPref := ifile.readString(m\_section, 'FvPrefix', 'Vt');

if m\_FvPref = '' then

m\_FvPref := 'Vt';

m\_regionPref := ifile.readString(m\_section, 'RegionPrefix', 'Reg');

if m\_regionPref = '' then

m\_regionPref := 'Reg';

m\_TestPref := ifile.readString(m\_section, 'TestPrefix', 'Test');

if m\_TestPref = '' then

m\_TestPref := 'Test';

m\_sectionPref := ifile.readString(m\_section, 'SectionPrefix', 'Seg');

if m\_sectionPref = '' then

m\_sectionPref := 'Seg';

m\_VCPref := ifile.readString(m\_section, 'VCPrefix', 'VC\_');

if m\_VCPref = '' then

m\_VCPref := 'VC\_';

m\_H\_Pref := ifile.readString(m\_section, 'HPrefix', 'H\_');

if m\_H\_Pref = '' then

m\_H\_Pref := 'H\_';

m\_Hin\_Pref := ifile.readString(m\_section, 'HinPrefix', 'Hin\_');

if m\_Hin\_Pref = '' then

m\_Hin\_Pref := 'Hin\_';

m\_Hout\_Pref := ifile.readString(m\_section, 'HoutPrefix', 'Hout\_');

if m\_Hout\_Pref = '' then

m\_Hout\_Pref := 'Hout\_';

// Путь к замеру для тарировки вертикально центрального нагружения

LoadfromIniTareFrame(ifile, VC\_Frame);

LoadfromIniTareFrame(ifile, H\_Frame);

LoadfromIniTareFrame(ifile, Hin\_Frame);

LoadfromIniTareFrame(ifile, Hout\_Frame);

BaseFolderEdit.Text := ifile.readString(m\_section, 'BaseFolder', '');

TestsFolderE.Text := ifile.readString(m\_section, 'TestsFolder', '.\Tests');

// путь к заезду

TestPath.Text := ifile.readString(m\_section, 'TestPath', '');

TestPathChange(nil);

// участок

str := ifile.readString(m\_section, 'Region', '');

if str <> '' then

begin

RegionCB.Text := str;

RegionSE.Value := strtoint(getendnum(RegionCB.Text));

end;

// сечение

str := ifile.readString(m\_section, 'Section', '');

if str <> '' then

begin

sectionCB.Text := str;

GetNumFromSectionCB;

end;

Hout\_D.FloatNum := ifile.readFloat(m\_section, 'Dout', 20);

ExportMatrixPath.Text := ifile.readString(m\_section, 'ExportMatrixPath', '');

ImportMatrixPath.Text := ifile.readString(m\_section, 'ImportMatrixPath', '');

ExportMatrixPathChange(nil);

if checkImportFolder then

begin

ImpMatrixBtnClick(nil);

end;

PolySE.Value := ifile.ReadInteger(m\_section, 'Poly', 1);

PolyCount.IntNum := ifile.ReadInteger(m\_section, 'PolyCount', 100);

// использовать усреднение

TrendCB.Checked := ifile.ReadBool(m\_section, 'UseTrend', false);

TrendPortionIE.IntNum := ifile.ReadInteger(m\_section, 'TrendNumPoints', 0);

TrendPortionFE.FloatNum := IniReadFloatEx(ifile,m\_section, 'TrendPortion', 0);

// использовать lopassfilter

LPFCB.Checked := ifile.ReadBool(m\_section, 'UseLPF', false);

LPFOrder.IntNum := ifile.ReadInteger(m\_section, 'LPFOrder', 1);

LPFfreq.FloatNum := IniReadFloatEx(ifile,m\_section, 'LPFfreq', 0);

useSignalPrefix.Checked := ifile.ReadBool(m\_section, 'UsePrefix', true);

ChangeTestByCutCB.Checked := ifile.ReadBool(m\_section, 'ChangeTestByCut',

true);

cleargist;

count := ifile.ReadInteger(m\_section, 'GistCount', 0);

for i := 0 to count - 1 do

begin

str := ifile.readString(m\_section, 'gist\_' + inttostr(i), '');

start := strtoint(GetSubString(str, ';', 1, ind));

shift := strtoint(GetSubString(str, ';', ind + 1, ind));

b := strtobool(GetSubString(str, ';', ind + 1, ind));

lvl := strtofloat(GetSubString(str, ';', ind + 1, ind));

noise := strtofloat(GetSubString(str, ';', ind + 1, ind));

addgistproc(start, shift, lvl, noise, true);

end;

ifile.Destroy;

createDB;

ReportCBClick(nil);

end;

procedure TRZDFrm.SaveToIniTareFrame(ifile: tinifile; fr: TRZDTareFrame);

var

subStr: string;

i: integer;

begin

// Путь к замеру для тарировки вертикально центрального нагружения

for i := 1 to length(fr.name) - 1 do

begin

if fr.name[i] = '\_' then

begin

subStr := copy(fr.Name, 1, i - 1);

break;

end;

end;

ifile.writeFloat(m\_section, subStr + '\_' + 'Null1', fr.NullFE1.FloatNum);

ifile.writeFloat(m\_section, subStr + '\_' + 'Null2', fr.NullFE2.FloatNum);

ifile.writeFloat(m\_section, subStr + '\_' + 'T1', fr.T1FE.FloatNum);

ifile.writeFloat(m\_section, subStr + '\_' + 'T2', fr.T2FE.FloatNum);

ifile.writeString(m\_section, subStr + '\_' + 'Path', fr.Path.Text);

// верт. сила

ifile.writeString(m\_section, subStr + '\_' + 'VF', fr.VFCbox.Text);

// верт. сила

ifile.writeString(m\_section, subStr + '\_' + 'HF', fr.HFCbox.Text);

ifile.writeString(m\_section, subStr + '\_' + 'S1', fr.S1Cbox.Text);

ifile.writeString(m\_section, subStr + '\_' + 'S2', fr.S2Cbox.Text);

ifile.writeString(m\_section, subStr + '\_' + 'S3', fr.S3Cbox.Text);

ifile.writeString(m\_section, subStr + '\_' + 'S4', fr.S4Cbox.Text);

end;

procedure TRZDFrm.Save;

var

ifile: tinifile;

i: integer;

G: cGist;

str: string;

begin

ifile := tinifile.create(m\_cfg);

ifile.writeFloat(m\_section, 'VtMaxRel', FMaxEdit.FloatNum);

ifile.writeFloat(m\_section, 'VtMaxAbs', FmaxAbsFE.FloatNum);

ifile.writeFloat(m\_section, 'GtMaxRel', GtMaxEdit.FloatNum);

ifile.writeFloat(m\_section, 'GtMaxAbs', GtmaxAbsFE.FloatNum);

ifile.writeFloat(m\_section, 'MaxF', v\_MaxF);

ifile.writeInteger(m\_section, 'RepType', reptype.ItemIndex);

ifile.WriteBool(m\_section, 'MatrixBtn', ApplyBtn.Visible);

ifile.writeBool(m\_section, 'NullCB', NullCb.Checked);

ifile.writeBool(m\_section, 'ExtImages', ExtImagesCB.Checked);

ifile.writeBool(m\_section, 'GenReport', ReportCB.Checked);

ifile.writeBool(m\_section, 'OpenReport', OpenReportCB.Checked);

ifile.writeFloat(m\_section, 'CommentDx', c\_CommentDx);

ifile.writeFloat(m\_section, 'CommentDy', c\_CommentDy);

ifile.writeBool(m\_section, 'GistAxXScale', v\_GistAxXScale);

ifile.writeFloat(m\_section, 'GistAxXRate', v\_GistAxXRate);

ifile.writeFloat(m\_section, 'BitScale', v\_bitScale);

ifile.writeString(m\_section, 'BaseFolder', BaseFolderEdit.Text);

// каталог для списка заездов

ifile.writeString(m\_section, 'TestsFolder', TestsFolderE.Text);

// участок

ifile.writeString(m\_section, 'Region', RegionCB.Text);

// сечение

ifile.writeString(m\_section, 'Section', sectionCB.Text);

ifile.writeFloat(m\_section, 'Dout', Hout\_D.FloatNum);

SaveToIniTareFrame(ifile, VC\_Frame);

SaveToIniTareFrame(ifile, H\_Frame);

SaveToIniTareFrame(ifile, Hin\_Frame);

SaveToIniTareFrame(ifile, Hout\_Frame);

ifile.writeString(m\_section, 'ExportMatrixPath', ExportMatrixPath.Text);

ifile.writeString(m\_section, 'ImportMatrixPath', ImportMatrixPath.Text);

// путь к заезду

ifile.writeString(m\_section, 'TestPath', TestPath.Text);

ifile.writeString(m\_section, 'T1', '');

ifile.writeString(m\_section, 'T2', '');

ifile.writeInteger(m\_section, 'Poly', PolySE.Value);

ifile.writeInteger(m\_section, 'PolyCount', PolyCount.IntNum);

// использовать усреднение

ifile.writeBool(m\_section, 'UseTrend', TrendCB.Checked);

ifile.writeInteger(m\_section, 'TrendNumPoints', TrendPortionIE.IntNum);

ifile.writeFloat(m\_section, 'TrendPortion', TrendPortionFE.FloatNum);

// использовать lopassfilter

ifile.writeBool(m\_section, 'UseLPF', LPFCB.Checked);

ifile.writeInteger(m\_section, 'LPFOrder', LPFOrder.IntNum);

ifile.writeFloat(m\_section, 'LPFfreq', LPFfreq.FloatNum);

// сохраняем префиксы

ifile.writeString(m\_section, 'D1prefix', m\_D1Pref);

ifile.writeString(m\_section, 'D2prefix', m\_D2Pref);

ifile.writeString(m\_section, 'D3prefix', m\_D3Pref);

ifile.writeString(m\_section, 'D4prefix', m\_D4Pref);

ifile.writeString(m\_section, 'Fhprefix', m\_FhPref);

ifile.writeString(m\_section, 'Fvprefix', m\_FvPref);

ifile.writeString(m\_section, 'RegionPrefix', m\_regionPref);

ifile.writeString(m\_section, 'TestPrefix', m\_TestPref);

ifile.writeString(m\_section, 'SectionPrefix', m\_sectionPref);

ifile.writeString(m\_section, 'VCPrefix', m\_VCPref);

ifile.writeString(m\_section, 'HPrefix', m\_H\_Pref);

ifile.writeString(m\_section, 'HinPrefix', m\_Hin\_Pref);

ifile.writeString(m\_section, 'HoutPrefix', m\_Hout\_Pref);

ifile.writeBool(m\_section, 'UsePrefix', useSignalPrefix.Checked);

ifile.writeBool(m\_section, 'ChangeTestByCut', ChangeTestByCutCB.Checked);

ifile.writeInteger(m\_section, 'GistCount', GistLV.Items.count);

for i := 0 to GistLV.Items.count - 1 do

begin

G := cGist(GistLV.Items[i].data);

str := inttostr(G.start) + ';' + inttostr(G.shift) + ';' + booltostr

(G.proc, true) + ';' + floattostr(G.lvl) + ';' + floattostr(G.noise);

ifile.writeString(m\_section, 'gist\_' + inttostr(i), str);

end;

ifile.Destroy;

end;

function TRZDFrm.LoadSrc(str: string): csrc;

begin

result := m\_wpMng.LoadSrc(str);

end;

procedure TRZDFrm.LoadVCSrc;

begin

vc\_src := LoadSrc(VC\_Frame.Path.Text);

end;

procedure TRZDFrm.MatrixLVChange(Sender: TObject; Item: TListItem;

Change: TItemChange);

var

li: TListItem;

i, cut, usenames: integer;

reg: string;

m: TRZDMatrix;

j: integer;

cb: TComboBox;

begin

if MatrixLV.Selected = nil then

exit;

// обработка свойства сечение

cut := -1;

for i := 0 to MatrixLV.Items.count - 1 do

begin

if not MatrixLV.Items[i].Selected then

continue;

m := TRZDMatrix(MatrixLV.Items[i].data);

if cut = -1 then

begin

cut := m.cut;

end

else

begin

if cut <> m.cut then

begin

cut := -1;

break;

end;

end;

end;

if cut = -1 then

Matrix\_CutSE.Text := ''

else

Matrix\_CutSE.Value := cut;

// обработка свойства регион

reg := '-1';

for i := 0 to MatrixLV.Items.count - 1 do

begin

if not MatrixLV.Items[i].Selected then

continue;

m := TRZDMatrix(MatrixLV.Items[i].data);

if reg = '-1' then

begin

reg := m.region;

end

else

begin

if reg <> m.region then

begin

reg := '-1';

break;

end;

end;

end;

if reg = '-1' then

MatrixRegSE.Text := ''

else

MatrixRegSE.Text := reg;

// обработка свойства искать имена

usenames := 1;

for i := 0 to MatrixLV.Items.count - 1 do

begin

if not MatrixLV.Items[i].Selected then

continue;

m := TRZDMatrix(MatrixLV.Items[i].data);

SetMultiSelectComponentBool(mat\_UseNamesCB, m.useSNames);

end;

endMultiSelect(mat\_UseNamesCB);

// обработка свойства имен датчиков

for i := 0 to MatrixLV.Items.count - 1 do

begin

if MatrixLV.Items[i].Selected then

begin

m := TRZDMatrix(MatrixLV.Items[i].data);

for j := 0 to 3 do

begin

cb := GetSensCB(j);

if MatrixLV.Items[i] = MatrixLV.Selected then

cb.Text := m.s[j]

else

begin

if cb.Text <> '\_' then

begin

if cb.Text <> m.s[j] then

begin

cb.Text := '\_';

end;

end;

end;

end;

end;

end;

// обработка матрицы

if MatrixLV.SelCount = 1 then

begin

ShowG(m.m);

end;

if Change = ctState then

begin

if Item <> nil then

begin

if Item.data <> nil then

begin

TRZDMatrix(Item.data).m\_active := Item.Checked;

end;

end;

end;

end;

function TRZDFrm.GetSensCB(i: integer): TComboBox;

begin

case i of

0:

begin

result := Mat\_S1CB;

end;

1:

begin

result := Mat\_S2CB;

end;

2:

begin

result := Mat\_S3CB;

end;

3:

begin

result := Mat\_S4CB;

end;

end;

end;

procedure TRZDFrm.LoadHSrc;

begin

h\_src := LoadSrc(H\_Frame.Path.Text);

end;

procedure TRZDFrm.LoadHinCSrc;

begin

hin\_src := LoadSrc(Hin\_Frame.Path.Text);

end;

procedure TRZDFrm.LoadHoutSrc;

begin

hout\_src := LoadSrc(Hout\_Frame.Path.Text);

end;

procedure SetCBIndex(cb: TComboBox; str: string);

var

i: integer;

begin

for i := 0 to cb.Items.count - 1 do

begin

if cb.Items[i] = str then

begin

cb.ItemIndex := i;

exit;

end;

end;

end;

procedure TRZDFrm.FillCB(V\_cb, H\_cb, D1\_cb, D2\_cb, D3\_cb, D4\_cb: TComboBox;

merafile: string);

var

i: integer;

str, str1, sname, V\_cb\_str, H\_cb\_str, D1\_cb\_str, D2\_cb\_str, D3\_cb\_str,

D4\_cb\_str: string;

ifile: tinifile;

list: tstringlist;

begin

// загрузка имен датчиков

ifile := tinifile.create(merafile);

list := tstringlist.create;

ifile.ReadSections(list);

for i := 0 to list.count - 1 do

begin

if lowercase(list.Strings[i]) = 'mera' then

begin

list.Delete(i);

break;

end;

end;

ifile.Destroy;

V\_cb\_str := V\_cb.Text;

H\_cb\_str := H\_cb.Text;

D1\_cb\_str := D1\_cb.Text;

D2\_cb\_str := D2\_cb.Text;

D3\_cb\_str := D3\_cb.Text;

D4\_cb\_str := D4\_cb.Text;

V\_cb.clear;

H\_cb.clear;

D1\_cb.clear;

D2\_cb.clear;

D3\_cb.clear;

D4\_cb.clear;

for i := 0 to list.count - 1 do

begin

sname := list.Strings[i];

if useSignalPrefix.Checked then

begin

str := getendnum(sectionCB.Text);

str1 := DelNullCharsInNumStr(str);

if CheckPosSubstr(m\_FvPref, sname) then

begin

V\_cb.AddItem(sname, nil);

end;

if CheckPosSubstr(m\_FhPref, sname) then

begin

H\_cb.AddItem(sname, nil);

end;

if CheckPosSubstr(m\_D1Pref + str, sname) or CheckPosSubstr

(m\_D1Pref + str1, sname) then

begin

D1\_cb.AddItem(sname, nil);

end;

if CheckPosSubstr(m\_D2Pref + str, sname) or CheckPosSubstr

(m\_D2Pref + str1, sname) then

begin

D2\_cb.AddItem(sname, nil);

end;

if CheckPosSubstr(m\_D3Pref + str, sname) or CheckPosSubstr

(m\_D3Pref + str1, sname) then

begin

D3\_cb.AddItem(sname, nil);

end;

if CheckPosSubstr(m\_D4Pref + str, sname) or CheckPosSubstr

(m\_D4Pref + str1, sname) then

begin

D4\_cb.AddItem(sname, nil);

end;

end

else

begin

V\_cb.AddItem(sname, nil);

H\_cb.AddItem(sname, nil);

D1\_cb.AddItem(sname, nil);

D2\_cb.AddItem(sname, nil);

D3\_cb.AddItem(sname, nil);

D4\_cb.AddItem(sname, nil);

end;

end;

SetCBIndex(V\_cb, V\_cb\_str);

SetCBIndex(H\_cb, H\_cb\_str);

SetCBIndex(D1\_cb, D1\_cb\_str);

SetCBIndex(D2\_cb, D2\_cb\_str);

SetCBIndex(D3\_cb, D3\_cb\_str);

SetCBIndex(D4\_cb, D4\_cb\_str);

if (V\_cb.Items.count > 0) and (V\_cb.ItemIndex < 0) then

V\_cb.ItemIndex := 0;

if (H\_cb.Items.count > 0) and (H\_cb.ItemIndex < 0) then

H\_cb.ItemIndex := 0;

if (D1\_cb.Items.count > 0) and (D1\_cb.ItemIndex < 0) then

D1\_cb.ItemIndex := 0;

if (D2\_cb.Items.count > 0) and (D2\_cb.ItemIndex < 0) then

D2\_cb.ItemIndex := 0;

if (D3\_cb.Items.count > 0) and (D3\_cb.ItemIndex < 0) then

D3\_cb.ItemIndex := 0;

if (D4\_cb.Items.count > 0) and (D4\_cb.ItemIndex < 0) then

D4\_cb.ItemIndex := 0;

end;

procedure TRZDFrm.FillVCCB;

begin

FillCB(VC\_Frame.VFCbox, VC\_Frame.HFCbox, VC\_Frame.S1Cbox, VC\_Frame.S2Cbox,

VC\_Frame.S3Cbox, VC\_Frame.S4Cbox, VC\_Frame.Path.Text);

end;

procedure TRZDFrm.FillHCB;

begin

FillCB(H\_Frame.VFCbox, H\_Frame.HFCbox, H\_Frame.S1Cbox, H\_Frame.S2Cbox,

H\_Frame.S3Cbox, H\_Frame.S4Cbox, H\_Frame.Path.Text);

end;

procedure TRZDFrm.FillHoutCB;

begin

FillCB(Hout\_Frame.VFCbox, Hout\_Frame.HFCbox, Hout\_Frame.S1Cbox,

Hout\_Frame.S2Cbox, Hout\_Frame.S3Cbox, Hout\_Frame.S4Cbox,

Hout\_Frame.Path.Text);

end;

procedure TRZDFrm.FillHinCB;

begin

FillCB(Hin\_Frame.VFCbox, Hin\_Frame.HFCbox, Hin\_Frame.S1Cbox,

Hin\_Frame.S2Cbox, Hin\_Frame.S3Cbox, Hin\_Frame.S4Cbox, Hin\_Frame.Path.Text);

end;

procedure TRZDFrm.FormClose(Sender: TObject; var Action: TCloseAction);

begin

Save;

end;

procedure TRZDFrm.FormShow(Sender: TObject);

begin

Load;

end;

Function GetVtSignal(src: csrc; cut: integer): cwpsignal;

begin

end;

procedure TRZDFrm.BuildReport(vt, gt: iwpSignal; m: TRZDMatrix);

var

vt\_gist, gt\_gist, gt1, gt2, probabl1, probabl2, probabl3,s1, s2: iwpSignal;

trig, noise, mo, max: double;

n, color:integer;

folder, cutstr, str: string;

ifile:tinifile;

graph: tgraphstruct;

dbfld: cSegmentFolder;

l\_date:tdatetime;

l\_s, l\_sr, vc, h, hin, hout: csrc;

sig:cwpsignal;

calibr:boolean;

I: Integer;

begin

// BuildGistogram(s:iwpsignal; dx, lvl, noise:double;start, shift:integer;folder:string):iwpsignal;overload;

folder := GetSignalFolder(vt);

trig := GistTrigFE.FloatNum;

if GistProcCB.Checked then

trig := trig \* (vt.MaxY - vt.MinY);

noise := GistNoiseFE.FloatNum;

if GistProcCB.Checked then

noise := noise \* (vt.MaxY - vt.MinY);

cutstr := '\_Сеч.' + inttostr(m.cut);

vt\_gist := BuildGistogram(vt, GistDxFE.FloatNum, trig, noise, 0, 0, folder,

'Верт. сила' + cutstr);

trig := GistTrigFE.FloatNum;

if GistProcCB.Checked then

trig := trig \* (gt.MaxY - gt.MinY);

noise := GistNoiseFE.FloatNum;

if GistProcCB.Checked then

noise := noise \* (gt.MaxY - gt.MinY);

gt\_gist := BuildGistogram(gt, GistDxFE.FloatNum, trig, noise, 0, 0, folder,

'Гор. сила' + cutstr);

gt1 := BuildGistogram(gt, GistDxFE.FloatNum, trig, noise, 0, 1, folder,

'Осн. ось' + cutstr);

gt2 := BuildGistogram(gt, GistDxFE.FloatNum, trig, noise, 1, 1, folder,

'Набегающая ось' + cutstr);

probabl1 := BuildProbabilityDistribution(vt\_gist, palfaedit.FloatNum);

probabl1.sname := 'Распред.гор.силы' + cutstr;

WP.Link(folder, probabl1.sname, probabl1);

probabl2 := BuildProbabilityDistribution(gt1, palfaedit.FloatNum);

probabl2.sname := 'Распред.Gt\_ведущ.' + cutstr;

WP.Link(folder, probabl2.sname, probabl2);

probabl3 := BuildProbabilityDistribution(gt2, palfaedit.FloatNum);

probabl3.sname := 'Распред.Gt\_ведом.' + cutstr;

WP.Link(folder, probabl3.sname, probabl3);

// создаем графики калибровочной матрицы

dbfld := m\_DB.getCalibr(m.region, m.cut);

calibr:=false;

if dbfld <> nil then

begin

l\_date:=dbfld.GetDate;

vc:=dbfld.getvcres;

h:=dbfld.gethres;

hin:=dbfld.gethinres;

hout:=dbfld.gethoutres;

if (vc<>nil) and (h<>nil) and (hin<>nil) and (hout<>nil) then

begin

calibr:=true;

// вертикальная сила

l\_s := dbfld.getvc;

l\_sr := dbfld.getvcres;

s1 := l\_s.getSignalObj(m.sensors[0].vt).Signal;

ifile:=tinifile.Create(l\_sr.merafile.FileName);

str:=ifile.ReadString('RZDHexa','VT','');

ifile.Destroy;

if str='' then

str:='Vt' + m.GetResPostfix;

sig:=l\_sr.getSignalObj(str);

if sig<>nil then

s2 := sig.Signal

else

begin

JournalLB.AddItem('Не найден результат тарировки Vt'+' Сечение:' + inttostr(m.cut)+' Регион:' +m.region+ ' в сигнале '+ extractfilename(l\_sr.merafile.FileName), nil);

exit;

end;

graph := createline(s1);

IWPGraphs(WP.GraphAPI).SetPageDim(graph.hpage, PAGE\_DM\_TABLE, 4, 2);

graph := createline(s2, graph.hgraph, graph.haxis);

s1 := l\_s.getSignalObj(m.sensors[0].gt).Signal;

ifile:=tinifile.Create(l\_sr.merafile.FileName);

str:=ifile.ReadString('RZDHexa','GT','');

ifile.Destroy;

if str='' then

str:='Gt' + m.GetResPostfix;

sig:=l\_sr.getSignalObj(str);

s2 := sig.Signal;

graph := createline(s1, graph.hgraph, graph.haxis);

graph := createline(s2, graph.hgraph, graph.haxis);

str := 'Вертикально';

IWPGraphs(WP.GraphAPI).AddComment(graph.hgraph, str, 45, 10, c\_CommentDx\*4, c\_CommentDy);

IWPGraphs(WP.GraphAPI).SetGraphOpt(graph.hgraph, 0,

GROPT\_SHOWNAME + GROPT\_SHOWLEGEND);

// горизонтальная сила

l\_s := dbfld.geth;

l\_sr := dbfld.gethres;

s1 := l\_s.getSignalObj(m.sensors[1].vt).Signal;

ifile:=tinifile.Create(l\_sr.merafile.FileName);

str:=ifile.ReadString('RZDHexa','VT','');

ifile.Destroy;

if str='' then

str:='Vt' + m.GetResPostfix;

sig:=l\_sr.getSignalObj(str);

s2 := sig.Signal;

graph := createline(s1, graph.hpage);

graph := createline(s2, graph.hgraph, graph.haxis);

s1 := l\_s.getSignalObj(m.sensors[1].gt).Signal;

ifile:=tinifile.Create(l\_sr.merafile.FileName);

str:=ifile.ReadString('RZDHexa','GT','');

ifile.Destroy;

if str='' then

str:='Gt' + m.GetResPostfix;

sig:=l\_sr.getSignalObj(str);

s2 := sig.Signal;

graph := createline(s1, graph.hgraph, graph.haxis);

graph := createline(s2, graph.hgraph, graph.haxis);

str := 'Горизонтально';

IWPGraphs(WP.GraphAPI).AddComment(graph.hgraph, str, 45, 10, c\_CommentDx\*4, c\_CommentDy);

IWPGraphs(WP.GraphAPI).SetGraphOpt(graph.hgraph, 0,

GROPT\_SHOWNAME + GROPT\_SHOWLEGEND);

// горизонтальная наружу

l\_s := dbfld.gethout;

l\_sr := dbfld.gethoutres;

s1 := l\_s.getSignalObj(m.sensors[2].vt).Signal;

ifile:=tinifile.Create(l\_sr.merafile.FileName);

str:=ifile.ReadString('RZDHexa','VT','');

ifile.Destroy;

if str='' then

str:='Vt' + m.GetResPostfix;

sig:=l\_sr.getSignalObj(str);

s2 := sig.Signal;

graph := createline(s1, graph.hpage);

graph := createline(s2, graph.hgraph, graph.haxis);

s1 := l\_s.getSignalObj(m.sensors[2].gt).Signal;

ifile:=tinifile.Create(l\_sr.merafile.FileName);

str:=ifile.ReadString('RZDHexa','GT','');

ifile.Destroy;

if str='' then

str:='Gt' + m.GetResPostfix;

sig:=l\_sr.getSignalObj(str);

s2 := sig.Signal;

graph := createline(s1, graph.hgraph, graph.haxis);

graph := createline(s2, graph.hgraph, graph.haxis);

str := 'Вертикально наружу';

IWPGraphs(WP.GraphAPI).AddComment(graph.hgraph, str, 45, 10, c\_CommentDx\*5, c\_CommentDy);

IWPGraphs(WP.GraphAPI).SetGraphOpt(graph.hgraph, 0,

GROPT\_SHOWNAME + GROPT\_SHOWLEGEND);

// горизонтальная внутрь

l\_s := dbfld.gethin;

l\_sr := dbfld.gethinres;

s1 := l\_s.getSignalObj(m.sensors[3].vt).Signal;

ifile:=tinifile.Create(l\_sr.merafile.FileName);

str:=ifile.ReadString('RZDHexa','VT','');

ifile.Destroy;

if str='' then

str:='Vt' + m.GetResPostfix;

sig:=l\_sr.getSignalObj(str);

s2 := sig.Signal;

graph := createline(s1, graph.hpage);

graph := createline(s2, graph.hgraph, graph.haxis);

s1 := l\_s.getSignalObj(m.sensors[3].gt).Signal;

ifile:=tinifile.Create(l\_sr.merafile.FileName);

str:=ifile.ReadString('RZDHexa','GT','');

ifile.Destroy;

if str='' then

str:='Gt' + m.GetResPostfix;

sig:=l\_sr.getSignalObj(str);

s2 := sig.Signal;

graph := createline(s1, graph.hgraph, graph.haxis);

graph := createline(s2, graph.hgraph, graph.haxis);

str := 'Вертикально внутрь';

IWPGraphs(WP.GraphAPI).AddComment(graph.hgraph, str, 45, 10, c\_CommentDx\*5, c\_CommentDy);

IWPGraphs(WP.GraphAPI).SetGraphOpt(graph.hgraph, 0,

GROPT\_SHOWNAME + GROPT\_SHOWLEGEND);

end;

end;

// создаем гистограммы

if calibr then

graph := createline(vt\_gist, graph.hpage)

else

begin

graph := createline(gt\_gist);

IWPGraphs(WP.GraphAPI).SetPageDim(graph.hpage, PAGE\_DM\_TABLE, 4, 1);

end;

setGistXScale(graph.hline);

mo := strtofloat(vt\_gist.GetProperty('MO'));

max := strtofloat(vt\_gist.GetProperty('GistMax'));

n:=strtoint(vt\_gist.GetProperty('GistN'));

str := 'M=' + formatstr(mo, c\_digits) + #10 + 'Max=' + formatstr(max,c\_digits)+#10 + 'N=' + inttostr(n);

IWPGraphs(WP.GraphAPI).AddComment(graph.hgraph, str, 55, 10, c\_CommentDx\*4, c\_CommentDy\*2);

IWPGraphs(WP.GraphAPI).SetGraphOpt(graph.hgraph, 0,

GROPT\_SHOWNAME + GROPT\_SHOWLEGEND);

// Гистограмма суммарного распределения боковой силы

//graph := createline(gt\_gist, graph.hpage);

//setGistXScale(graph.hline);

//mo := strtofloat(gt\_gist.GetProperty('MO'));

//max := strtofloat(gt\_gist.GetProperty('GistMax'));

//n:=strtoint(gt\_gist.GetProperty('GistN'));

//str := 'M=' + formatstr(mo, c\_digits) + #10 + 'Max=' + formatstr(max,c\_digits)+#10 + 'N=' + inttostr(n);

//IWPGraphs(WP.GraphAPI).AddComment(graph.hgraph, str, 55, 10, c\_CommentDx\*4, c\_CommentDy\*2);

//IWPGraphs(WP.GraphAPI).SetGraphOpt(graph.hgraph, 0,

// GROPT\_SHOWNAME + GROPT\_SHOWLEGEND);

// две гистограммы

// Гистограмма ведущей оси

graph := createline(gt1, graph.hpage);

setGistXScale(graph.hline);

mo := strtofloat(gt1.GetProperty('MO'));

max := strtofloat(gt1.GetProperty('GistMax'));

n:=strtoint(gt1.GetProperty('GistN'));

str :='ведущ. ось' + #10+ 'M=' + formatstr(mo, c\_digits) + #10 + 'Max=' + formatstr(max,c\_digits)+#10 + 'N=' + inttostr(n);

IWPGraphs(WP.GraphAPI).AddComment(graph.hgraph, str, 53, 10, c\_CommentDx\*4.5, c\_CommentDy\*2.5);

IWPGraphs(WP.GraphAPI).SetGraphOpt(graph.hgraph, 0,

GROPT\_SHOWNAME + GROPT\_SHOWLEGEND);

color:=clGreen;

IWPGraphs(WP.GraphAPI).SetLineOpt(graph.hline, 0, LNOPT\_color, 0, color);

// Гистограмма ведомой оси

//graph := createline(gt2, graph.hgraph, graph.haxis);

graph := createline(gt2, graph.hpage);

setGistXScale(graph.hline);

mo := strtofloat(gt2.GetProperty('MO'));

max := strtofloat(gt2.GetProperty('GistMax'));

n:=strtoint(gt2.GetProperty('GistN'));

str :='Ведомая ось' + #10 + 'M=' + formatstr(mo, c\_digits) + #10 + 'Max=' + formatstr(max,c\_digits)+#10 + 'N=' + inttostr(n);

IWPGraphs(WP.GraphAPI).AddComment(graph.hgraph, str, 53, 40, c\_CommentDx\*4.5, c\_CommentDy\*2.5);

IWPGraphs(WP.GraphAPI).SetGraphOpt(graph.hgraph, 0,

GROPT\_SHOWNAME + GROPT\_SHOWLEGEND);

color:=clRed;

IWPGraphs(WP.GraphAPI).SetLineOpt(graph.hline, 0, LNOPT\_color, 0, color);

// График с 3-я комментами

graph.hgraph := IWPGraphs(wp.GraphApi).CreateGraph(graph.hpage);

graph.haxis := 0;

graph.hline := 0;

str := probabl1.GetProperty('Sigma');

if str <> '' then

begin

trig := strtofloat(str);

str := 'Vt Верт.' + #10 + 'Palfa=' + formatstr(trig, c\_digits);

IWPGraphs(WP.GraphAPI).AddComment(graph.hgraph, str, 20, 5, c\_CommentDx\*5, c\_CommentDy\*2);

end;

str := probabl2.GetProperty('Sigma');

if str <> '' then

begin

trig := strtofloat(str);

str := 'Gt ведущ.' + #10 + 'Palfa=' + formatstr(trig, c\_digits);

IWPGraphs(WP.GraphAPI).AddComment(graph.hgraph, str, 20, 20, c\_CommentDx\*5, c\_CommentDy\*2);

end;

str := probabl3.GetProperty('Sigma');

if str <> '' then

begin

trig := strtofloat(str);

str := 'Gt ведом.' + #10 + 'Palfa=' + formatstr(trig, c\_digits);

IWPGraphs(WP.GraphAPI).AddComment(graph.hgraph, str, 20, 35, c\_CommentDx\*5, c\_CommentDy\*2);

end;

IWPGraphs(WP.GraphAPI).SetGraphOpt(graph.hgraph, 0, GROPT\_SHOWNAME + GROPT\_SHOWLEGEND);

//str := probabl.GetProperty('Sigma');

//if str <> '' then

//begin

// trig := strtofloat(str);

// str := 'Распред. гор. силы' + #10 + 'Palfa=' + formatstr(trig, c\_digits);

// IWPGraphs(WP.GraphAPI).AddComment(graph.hgraph, str, 20, 30, c\_CommentDx\*5, c\_CommentDy\*2);

//end;

//IWPGraphs(WP.GraphAPI).SetGraphOpt(graph.hgraph, 0,

// GROPT\_SHOWNAME + GROPT\_SHOWLEGEND);

//addRepPage(graph, m.date, inttostr(m.cut), '');

// Сохранение отчета

for I := length(folder) downto 1 do

begin

if folder[i]='/' then

begin

folder:=copy(folder, 1, i-1);

break;

end;

end;

l\_s:=m\_wpMng.GetSrc(folder);

folder:=trimname(folder);

folder:=folder+' Дата:'+l\_s.merafile.Date;

//addRepPageFr(graph, datetostr(l\_date), inttostr(m.cut)+' '+m.region, folder, calibr, l\_s.merafile.FileName, m\_pageIndex);

addRepPageFr(graph, datetostr(l\_date), inttostr(m.cut), folder, calibr, l\_s.merafile.FileName, m\_pageIndex);

end;

Procedure ScaleBitmap(Const BMP: TBitmap; Const NewWidth: Integer; Const NewHeight: Integer);

Var

TMP: TBitmap;

Begin

If (BMP.Width=NewWidth) And (BMP.Height=NewHeight) Then

Exit;

TMP:=TBitmap.Create();

Try

tmp.Width:=NewWidth;

tmp.Height:=NewHeight;

// TransparentColor:=clBlack;

// cliprect - результрующие габариты, bmp - исходная картинка

//StretchDraw(ClipRect, BMP);

SetStretchBltMode(tmp.canvas.handle, ColorOnColor);

SetStretchBltMode(tmp.canvas.handle, HalfTone);

StretchBlt(tmp.canvas.Handle,0,0,NewWidth,NewHeight,

bmp.canvas.handle,0,0,bmp.Width,bmp.Height,SRCCOPY);

BMP.Assign(TMP);

Finally

TMP.Free();

End;

End;

procedure TRZDFrm.addRepPageFR(g:tgraphstruct; date:string; cut:string; testcaption:string;findcalibr:boolean; filepath:string; var pageIndex:integer);

var

txt: TfrxMemoView;

t:TfrxRichView;

bmp:TfrxPictureView;

page: TfrxReportPage;

png:TPNGImage;

Bit: TBitmap;

gif:TGIFImage;

res:boolean;

fpath:string;

begin

inc(pageIndex);

txt:=frxReport1.FindObject('From') as TfrxMemoView;

txt.Text:='Тарировка от '+date;

txt:=frxReport1.FindObject('Cut') as TfrxMemoView;

txt.Text:='Сечение:'+cut;

txt:=frxReport1.FindObject('TestMemo') as TfrxMemoView;

txt.Text:='Испытание '+testcaption;

// заполняем картинку

bmp:=frxReport1.FindObject('Bmp1') as TfrxPictureView;

if not def\_init then

begin

def\_h:=bmp.Height;

def\_init:=true;

end;

if not findcalibr then

begin

bmp.Height:=def\_h/1.6;

end

else

bmp.Height:=def\_h;

WP.SaveImage('', '');

if ExtImagesCB.Checked then

begin

// fastreport работает только с png

fpath:=extractfiledir(filepath)+'\graph'+inttostr(pageIndex)+'.png';

png:=TPngImage.Create;

png.CompressionLevel:=7;

png.LoadFromClipboardFormat(CF\_BITMAP,Clipboard.GetAsHandle(CF\_BITMAP), CF\_BITMAP);

png.SaveToFile(fpath);

png.free;

//WP.SaveImage(fpath, '');

//fpath:='G:\oburec\project2010\2011\wp\RZD\signals\Tests\sig02\1.png';

end;

if clipboard.HasFormat(CF\_BITMAP) or clipboard.HasFormat(CF\_PICTURE ) then

begin

//if ExtImagesCB.Checked then

//begin

// bmp.FileLink:=fpath;

// bmp.Stretched:=true;

//bmp.AutoSize:=true;

//bmp.Width:=800;

//bmp.Height:=600;

//end

//else

begin

// если не rtf с высоким качеством

if reptype.ItemIndex<>1 then

begin

bmp.Picture.Assign(Clipboard);

end

else

begin

Bit:=TBitmap.Create;

Bit.Assign(Clipboard);

Bit.PixelFormat:=pf8bit;

ScaleBitmap(BIT,ROUND(Bit.Width\*v\_bitScale),ROUND(Bit.Height\*v\_bitScale));

bmp.Picture.Assign(BIT);

end;

end;

end;

// создаем страницу отчета в памяти

frxReport1.PrepareReport(false);

frxReport1.PrintOptions.ShowDialog:=false;

end;

procedure TRZDFrm.Hin\_EvalBtnClick(Sender: TObject);

var

m: TRZDMatrix;

i: integer;

n: iwpnode;

dir, filename: string;

resSrc: csrc;

s, vt, gt: cwpsignal;

begin

// G\*S

m\_src := LoadSrc(gettestpath);

if m\_src = nil then

begin

CheckFolderComponent(TestPath, TestPath.Hint, false);

exit;

end;

for i := 0 to MatrixList.count - 1 do

begin

m := TRZDMatrix(MatrixList.Items[i]);

if m.m\_active then

m.ApplyMatrix(m\_src);

end;

n := WP.Get(m\_src.name + '/RZDForce') as iwpnode;

dir := extractfiledir(m\_src.merafile.filename);

WP.SaveUSML(n.absolutepath, dir + '\RZDforce.mera');

// сохраняем матрицу рядом с замером

for i := 0 to MatrixList.count - 1 do

begin

m := MatrixList.Items[i];

ExportMatrix(m, dir + '\matrix.rzd');

end;

resSrc := m\_wpMng.GetSrc(n.absolutepath);

// for I := 0 to MatrixList.Count - 1 do

// строим распределения для вертикальной и горизонтальной силы

i:=0;

while i<resSrc.childCount do

begin

s := resSrc.GetWPSignal(i);

if CheckPosSubstr('Vt', s.name) then

begin

if not s.virt then

begin

if not CheckPosSubstr('gist', s.name) then

begin

// строим сразу все гистограммы по списку

BuildGistogram(s.Signal, GistDxFE.FloatNum, m\_src.name + '/RZDForce');

// флаг что обработан

s.virt:=true;

i:=-1;

end;

end;

end;

inc(i);

end;

i:=0;

while i<resSrc.childCount do

begin

s := resSrc.GetWPSignal(i);

if CheckPosSubstr('Gt', s.name) then

begin

if not s.virt then

begin

if not CheckPosSubstr('gist', s.name) then

begin

// строим сразу все гистограммы по списку

BuildGistogram(s.Signal, GistDxFE.FloatNum, m\_src.name + '/RZDForce');

// флаг что обработан

s.virt:=true;

i:=-1;

end;

end;

end;

inc(i);

end;

// создание отчета

frxReport1.PreviewPages.Clear();

frxRTFExport1.OpenAfterExport:=OpenReportCB.Checked;

frxPDFExport1.OpenAfterExport:=OpenReportCB.Checked;

frxHTMLExport1.OpenAfterExport:=OpenReportCB.Checked;

m\_pageIndex:=0;

for i := 0 to MatrixList.count - 1 do

begin

m := MatrixList.Items[i];

if not m.m\_active then

continue;

vt := resSrc.getSignalObj('Vt\_' + inttostr(m.cut) + '\_' + m.region);

gt := resSrc.getSignalObj('Gt\_' + inttostr(m.cut) + '\_' + m.region);

if ReportCB.Checked then

begin

if (vt <> nil) and (gt <> nil) then

BuildReport(vt.Signal, gt.Signal, m);

end;

end;

case reptype.ItemIndex of

0:filename:=dir+'\RZDreport.rtf';

1:filename:=dir+'\RZDreport.rtf';

2:filename:=dir+'\RZDreport.pdf';

3:filename:=dir+'\RZDreport.html';

end;

if fileuse(filename) then exit;

case reptype.ItemIndex of

0: // экспорт в Rtf

begin

frxRTFExport1.filename:=filename;

frxRTFExport1.OverwritePrompt:=false;

frxRTFExport1.ShowDialog:=false;

frxRTFExport1.ExportEMF:=false;

frxReport1.Export(frxRTFExport1);

end;

1: // экспорт в Rtf с выс. качеством

begin

frxRTFExport1.filename:=filename;

frxRTFExport1.ExportEMF:=true;

frxRTFExport1.OverwritePrompt:=false;

frxRTFExport1.ShowDialog:=false;

frxReport1.Export(frxRTFExport1);

end;

2: // экспорт в ПДФ

begin

frxPDFExport1.filename:=filename;

frxPDFExport1.OverwritePrompt:=false;

frxPDFExport1.ShowDialog:=false;

frxReport1.Export(frxPDFExport1);

end;

3: // экспорт в HTML

begin

frxHtmlExport1.filename:=filename;

frxHtmlExport1.OverwritePrompt:=false;

frxHtmlExport1.ShowDialog:=false;

frxReport1.Export(frxHtmlExport1);

end;

end;

end;

procedure TRZDFrm.VC\_FramePathChange(Sender: TObject);

begin

vc\_src := m\_wpMng.GetSrc(VC\_Frame.Path.Text);

CheckSrc;

end;

procedure TRZDFrm.VTree1DragDrop(Sender: TBaseVirtualTree; Source: TObject;

DataObject: IDataObject; Formats: TFormatArray; shift: TShiftState;

Pt: TPoint; var Effect: integer; Mode: TDropMode);

var

FmtEtc: TFormatEtc;

Medium: TStgMedium;

FileCount: integer;

i: integer;

FileNameLength: integer;

filename: string;

FileList: tstringlist;

begin

FileList := tstringlist.create;

FmtEtc.cfFormat := CF\_HDROP;

FmtEtc.ptd := nil;

FmtEtc.dwAspect := DVASPECT\_CONTENT;

FmtEtc.lindex := -1;

FmtEtc.tymed := TYMED\_HGLOBAL;

if DataObject.GetData(FmtEtc, Medium) = S\_OK then

try

try

// второй параметр iFile - при ffffff - число файлов, иначе индекс файла

FileCount := DragQueryFile(Medium.hGlobal, $FFFFFFFF, nil, 0);

for i := 0 to FileCount - 1 do

begin

FileNameLength := DragQueryFile(Medium.hGlobal, i, nil, 0);

setlength(filename, FileNameLength);

DragQueryFile(Medium.hGlobal, i, PChar(filename), FileNameLength + 1);

if testDir(filename) then

FileList.Add(filename);

end;

finally

DragFinish(Medium.hGlobal);

end;

finally

ReleaseStgMedium(Medium);

end;

FormStyle := fsNormal;

AddRZDDataFrm.showsignals(FileList);

FileList.Destroy;

end;

// modes to determine drop position further

// TDropMode = ( dmNowhere, dmAbove, dmOnNode, dmBelow );

// TDragState = (dsDragEnter, dsDragLeave, dsDragMove);

procedure TRZDFrm.VTree1DragOver(Sender: TBaseVirtualTree; Source: TObject;

shift: TShiftState; State: TDragState; Pt: TPoint; Mode: TDropMode;

var Effect: integer; var Accept: boolean);

begin

Accept := false;

if Source = nil then

Accept := true;

end;

procedure TRZDFrm.Hin\_FramePathChange(Sender: TObject);

begin

hin\_src := m\_wpMng.GetSrc(Hin\_Frame.Path.Text);

CheckSrc;

end;

procedure TRZDFrm.Hin\_FrameSelectIntervalCursorBtnClick(Sender: TObject);

begin

Hin\_Frame.SelectIntervalCursorBtnClick(Sender);

end;

procedure TRZDFrm.Hout\_FramePathChange(Sender: TObject);

begin

hout\_src := m\_wpMng.GetSrc(Hout\_Frame.Path.Text);

CheckSrc;

end;

procedure TRZDFrm.H\_FramePathChange(Sender: TObject);

begin

h\_src := m\_wpMng.GetSrc(H\_Frame.Path.Text);

CheckSrc;

end;

procedure testmnk;

var

x, y, p: array of double;

poly: integer;

m1, m2, m3: d2array;

i, j, c: integer;

begin

setlength(m1, 3, 3);

setlength(m2, 4, 3);

c := 1;

for i := 0 to 3 - 1 do

begin

for j := 0 to 3 - 1 do

begin

m1[i, j] := c;

inc(c);

end;

end;

c := 1;

for i := 0 to 4 - 1 do

begin

for j := 0 to 3 - 1 do

begin

m2[i, j] := c;

inc(c);

end;

end;

MultMatrix(m1, m2, m3);

poly := 2;

setlength(x, 8);

setlength(y, 8);

setlength(p, poly + 1);

x[0] := 0.72;

y[0] := 0.53;

x[1] := 0.99;

y[1] := 1.10;

x[2] := 1.11;

y[2] := 1.22;

x[3] := 1.76;

y[3] := 3.22;

x[4] := 1.86;

y[4] := 3.91;

x[5] := 2.55;

y[5] := 6.23;

x[6] := 3.24;

y[6] := 9.06;

x[7] := 4.53;

y[7] := 2.32;

buildMNK(poly, x, y, p);

end;

procedure TRZDFrm.ExportMatrixPathChange(Sender: TObject);

var

str, dir: string;

begin

str := RelativePathToAbsolute(BaseFolderEdit.Text, ExportMatrixPath.Text);

// dir:=extractfiledir(str)+'\RZDmatrix\';

// str:=dir+ExtractFileName(STR);

ExportMatrixPath.Hint := 'Путь: ' + str;

end;

procedure TRZDFrm.ImportMatrixPathChange(Sender: TObject);

begin

checkImportFolder;

end;

procedure TRZDFrm.BaseFolderEditChange(Sender: TObject);

begin

// ищем папку

if CheckFolderComponent(BaseFolderEdit, true) then

BEGIN

ReadRegionFolders;

UpdateTestsFolder;

END;

end;

procedure TRZDFrm.UpdateTestsFolder;

var

str, fld, testfolder: string;

list: tstringlist;

i: integer;

begin

str := GetTestsFolder;

TestsFolderE.Hint := str;

testfolder := str;

if CheckFolderComponent(TestsFolderE, testfolder, true) then

begin

str := TestPath.Text;

end;

TestPath.clear;

TestPath.Hint := '';

list := tstringlist.create;

FindFileext('\*.mera', testfolder, 1, list);

FindFileext('\*.usm', testfolder, 1, list);

TestPath.clear;

for i := 0 to list.count - 1 do

begin

fld := list.Strings[i];

if fld[length(fld)] <> gettestpath then

TestPath.AddItem(fld, nil);

end;

TestPath.ItemIndex := -1;

for i := 0 to list.count - 1 do

begin

if TestPath.Items[i] = str then

begin

TestPath.ItemIndex := i;

break;

end;

end;

if TestPath.ItemIndex = -1 then

begin

TestPath.AddItem(str, nil);

end;

list.Destroy;

end;

function TRZDFrm.GetTestsFolder: string;

begin

result := RelativePathToAbsolute(BaseFolderEdit.Text, TestsFolderE.Text);

end;

function TRZDFrm.testDir(filename: string): boolean;

var

list: tstringlist;

begin

list := tstringlist.create;

ScanDir(filename, '\*.mera', list, 1);

if list.count > 0 then

result := true

else

begin

result := false;

end;

list.Destroy;

end;

function TRZDFrm.checkImportFolder: boolean;

var

str, dir: string;

begin

str := RelativePathToAbsolute(BaseFolderEdit.Text, ImportMatrixPath.Text);

dir := extractfiledir(str);

str := dir + '\' + ExtractFileName(str);

ImportMatrixPath.Hint := 'Путь: ' + str;

// ищем файл

result := CheckFolderComponent(ImportMatrixPath, str, false);

if result then

begin

if m\_DB <> nil then

crzdbasefolder(m\_DB.m\_BaseFolder).m\_MatrixFolder.Path :=

ImportMatrixPath.Text;

end;

end;

procedure TRZDFrm.CheckBaseFolder;

begin

if DirectoryExists(BaseFolderEdit.Text) then

begin

BaseFolderEdit.color := clWindow;

ReadRegionFolders;

end

else

begin

BaseFolderEdit.color := $008080FF;

end;

end;

procedure TRZDFrm.UseMFmatrixClick(Sender: TObject);

var

dir, rzdname, fname: string;

begin

if UseMFmatrix.Checked then

begin

dir := extractfiledir(gettestpath) + '\';

fname := FindFile('\*.rzd', dir, 1);

if fileexists(dir + fname) then

begin

ImpMatrix(dir + fname);

end;

end

else

begin

ImpMatrixBtnClick(nil);

end;

end;

procedure TRZDFrm.EvalFzFy(interval: point2d; Fz, Fy: iwpSignal; name: string);

var

// коэффициенты полинома

ci: array of double;

x, y: olevariant;

p, p2: point2d;

size: integer;

i: integer;

graph: tgraphstruct;

str: string;

begin

if interval.y > interval.x then

begin

Fz := GetIntervalSignal(interval, Fz);

Fy := GetIntervalSignal(interval, Fy);

end;

m\_F2zFy := BuildWPDependence(Fy, Fz);

m\_F2zFy.sname := name + '\_cloud';

m\_F2zFy.comment := 'Y:' + Fz.sname + '\_X:' + Fy.sname;

size := m\_F2zFy.size;

x := VarArrayCreate([0, size], varDouble);

y := VarArrayCreate([0, size], varDouble);

m\_F2zFy.GetArray(0, size, y, x, true);

setlength(ci, 1);

buildMNK(0, x, y, size, ci);

// добавляем запись в журнал

str := name + ' Poly:';

for i := 0 to length(ci) - 1 do

begin

str := str + ' ' + formatstr(ci[i], c\_digits);

end;

JournalLB.AddItem(str, nil);

// создаем апроксимирующий полином в дереве сигналов

m\_F2zFy\_poly := iwpSignal(WP.CreateSignalXY(VT\_R8, VT\_R8)) as iwpSignal;

m\_F2zFy\_poly.size := 2;

m\_F2zFy\_poly.sname := name + '\_poly';

m\_F2zFy\_poly.comment := 'Y:' + Fz.sname + '\_X:' + Fy.sname;

p.y := fi(PolySE.Value, ci, m\_F2zFy.MinX);

p.x := m\_F2zFy.MinX;

m\_F2zFy\_poly.SetX(0, p.x);

m\_F2zFy\_poly.SetY(0, p.y);

p2.y := fi(PolySE.Value, ci, m\_F2zFy.MaxX);

p2.x := m\_F2zFy.MaxX;

m\_F2zFy\_poly.SetX(1, p2.x);

m\_F2zFy\_poly.SetY(1, p2.y);

// уровень прямой на полиноме

m\_F2zFyValue := p2.y;

if abs(m\_F2zFy.MaxX) > abs(m\_F2zFy.MinX) then

m\_F2zFyMaxX := m\_F2zFy.MaxX

else

m\_F2zFyMaxX := m\_F2zFy.MinX;

JournalLB.AddItem('Fz(Fy)=' + formatstr(p2.y, c\_digits), nil);

// графика

if LinkEvalWP.Checked then

begin

WP.Link('Signals/MNK/', m\_F2zFy.sname, m\_F2zFy);

WP.Link('Signals/MNK/', m\_F2zFy\_poly.sname, m\_F2zFy\_poly);

// строим графику

graph := createline(m\_F2zFy);

IWPGraphs(WP.GraphAPI).SetLineOpt(graph.hline, LNOPT\_ONLYPOINTS,

LNOPT\_ONLYPOINTS, 0, $00D2D5);

graph := createline(m\_F2zFy\_poly, graph.hgraph, graph.haxis);

WP.refresh;

end;

end;

function TRZDFrm.EvalEi(interval: point2d; s, V: iwpSignal; name: string): TEi;

var

// коэффициенты полинома

ci: array of double;

x, y: olevariant;

p, p2: point2d;

size: integer;

str: string;

i: integer;

begin

result.result := true;

if interval.y > interval.x then

begin

s := GetIntervalSignal(interval, s);

V := GetIntervalSignal(interval, V);

end;

// строим облако точек

result.cloud := BuildWPDependence(V, s);

result.cloud.sname := name + '\_cloud';

result.cloud.comment := 'Y:' + s.sname + '\_X:' + V.sname;

size := result.cloud.size;

x := VarArrayCreate([0, size - 1], varDouble);

y := VarArrayCreate([0, size - 1], varDouble);

result.cloud.GetArray(0, size, y, x, true);

setlength(ci, PolySE.Value + 1);

buildMNK(PolySE.Value, x, y, size, ci);

setlength(result.polyC, PolySE.Value + 1);

// сохраняем коэфициенты

for i := 0 to PolySE.Value do

begin

result.polyC[i] := ci[i];

end;

// добавляем запись в журнал

str := name + ' Poly:';

for i := 0 to length(ci) - 1 do

begin

str := str + ' ' + formatstr(ci[i], c\_digits);

end;

JournalLB.AddItem(str, nil);

// создаем апроксимирующий полином в дереве сигналов

if PolySE.Value < 2 then

begin

result.poly := iwpSignal(WP.CreateSignalXY(VT\_R8, VT\_R8)) as iwpSignal;

result.poly.size := 2;

end

else

begin

result.poly := iwpSignal(WP.CreateSignal(VT\_R8)) as iwpSignal;

result.poly.size := PolyCount.IntNum;

end;

result.poly.sname := name + '\_poly';

result.poly.comment := 'Y:' + s.sname + '\_X:' + V.sname;

if PolySE.Value < 2 then

begin

p.y := fi(PolySE.Value, ci, result.cloud.MinX);

p.x := result.cloud.MinX;

result.poly.SetX(0, p.x);

result.poly.SetY(0, p.y);

p2.y := fi(PolySE.Value, ci, result.cloud.MaxX);

p2.x := result.cloud.MaxX;

result.poly.SetX(1, p2.x);

result.poly.SetY(1, p2.y);

end

else

begin

result.poly.StartX := result.cloud.MinX;

result.poly.DeltaX := (result.cloud.MaxX - result.cloud.MinX)

/ result.poly.size;

for i := 0 to result.poly.size - 1 do

begin

result.poly.SetY(i, fi(PolySE.Value, ci,

i \* result.poly.DeltaX + result.poly.StartX));

end;

end;

if abs(result.cloud.MaxX) > abs(result.cloud.MinX) then

begin

result.fmax := result.cloud.MaxX;

// result.ei := abs(p.y);

result.ei := fi(PolySE.Value, ci, result.cloud.MaxX);

end

else

begin

result.fmax := result.cloud.MinX;

// result.ei := abs(p2.y);

result.ei := fi(PolySE.Value, ci, result.cloud.MinX);

end;

end;

function TRZDFrm.ShowTimeGraphs(p: integer; fr: TRZDTareFrame; src: csrc;

graphname: string): tgraphstruct;

var

s1, s2, s3, s4, vt, gt: iwpSignal;

graph: tgraphstruct;

begin

s1 := src.GetSignal(fr.S1Cbox.Text);

s2 := src.GetSignal(fr.S2Cbox.Text);

s3 := src.GetSignal(fr.S3Cbox.Text);

s4 := src.GetSignal(fr.S4Cbox.Text);

vt := src.GetSignal(fr.VFCbox.Text);

gt := src.GetSignal(fr.HFCbox.Text);

// графика

graph.hpage := p;

// строим графику

if p = 0 then

graph := createline(s1)

else

graph := createline(s1, p);

graph := createline(s2, graph.hgraph, graph.haxis);

graph := createline(s3, graph.hgraph, graph.haxis);

graph := createline(s4, graph.hgraph, graph.haxis);

// в новую ось

// graph := createline(vt, graph.hgraph, graph.haxis);

// graph := createline(gt, graph.hgraph, graph.haxis);

graph := createlineNewAx(vt, graph.hgraph);

graph := createline(gt, graph.hgraph, graph.haxis);

normaliseGraph(graph.hgraph);

result := graph;

end;

procedure TRZDFrm.StayOnTopBtnClick(Sender: TObject);

begin

if FormStyle = fsNormal then

begin

FormStyle := fsStayOnTop;

StayOnTopBtn.Hint := 'Поверх окон';

StayOnTopBtn.Down := true;

end

else

begin

FormStyle := fsNormal;

StayOnTopBtn.Down := false;

StayOnTopBtn.Hint := 'Скрывать окно';

end;

end;

// Загрузка данных для построения матрицы

function TRZDFrm.PrepareDataforMatrix: boolean;

begin

// загружаем источники

LoadVCSrc;

LoadHSrc;

LoadHinCSrc;

LoadHoutSrc;

result := CheckSrc;

end;

function TRZDFrm.GetSignal(src: csrc; sname: string; calibr: boolean;

var findSignal: boolean): iwpSignal;

var

str:string;

s, smo: iwpSignal;

mo: double;

flt: boolean;

isig:iwpsignal;

nullInterval: point2d;

begin

result := nil;

str:=src.name + '/Filter/'+sname+'\_Flt';

// '/Signals/Sig75.mera/Filter/\_141Vt\_Flt'

isig:=getISignalByPath(str);

if isig<>nil then

begin

findSignal:=true;

result:=isig;

exit;

end;

s := src.GetSignal(sname);

if s <> nil then

findSignal := true

else

begin

findSignal := false;

exit;

end;

flt := false;

if TrendCB.Checked then

begin

s := TrendMO(s, TrendPortionIE.IntNum);

flt := true;

end;

if LPFCB.Checked then

begin

s := lpf(s, LPFOrder.IntNum, LPFfreq.FloatNum);

flt := true;

end;

if flt then

s.sname := sname + '\_Flt';

if src = vc\_src then

begin

nullInterval := VC\_Frame.GetNullInterval;

end;

if src = h\_src then

begin

nullInterval := H\_Frame.GetNullInterval;

end;

if src = hin\_src then

begin

nullInterval := Hin\_Frame.GetNullInterval;

end;

if src = hout\_src then

begin

nullInterval := Hout\_Frame.GetNullInterval;

end;

// балансировка нуля

if calibr then

begin

if nullInterval.x <> nullInterval.y then

begin

if flt = false then

begin

flt := true;

end;

smo := GetIntervalSignal(nullInterval, s);

mo := GetMO(smo);

s := addconstant(-1 \* mo, s);

s.sname := sname + '\_Flt';

end;

end;

result := s;

if flt then

begin

str:=src.name + '/Filter/'+s.sname;

isig:=getISignalByPath(str);

if isig=nil then

begin

WP.Link(src.name + '/Filter/', s.sname, s);

wp.refresh;

end;

end;

end;

function TRZDFrm.CheckSrc: boolean;

begin

result := true;

if (vc\_src = nil) or (h\_src = nil) or (hout\_src = nil) or (hin\_src = nil) then

begin

result := false;

CreateMatrixBtn.Caption := 'Загрузить';

LoadBtnPanel.color := clBtnFace;

DrawGraphBtn.Hint := 'Не загружены сигналы';

end

else

begin

CreateMatrixBtn.Caption := 'Создать';

LoadBtnPanel.color := clGreen;

DrawGraphBtn.Hint := 'Отрисовать графики';

end;

end;

function TRZDFrm.checkSignalInterval(s: iwpSignal; interval: point2d): boolean;

var

min, max: double;

begin

result := true;

if (s.MaxX < interval.x) or (s.MinX > interval.y) then

begin

JournalLB.AddItem('Интервал сигнала не совпадает с выбраным интервалом!' +

formatstr(s.MinX, c\_digits) + '..' + formatstr(s.MaxX, c\_digits), nil);

result := false;

end;

end;

function getAbsMax(v1, v2: double): double;

begin

result := v1;

if abs(v1) < abs(v2) then

result := v2;

end;

procedure TRZDFrm.CreateMatrixBtnClick(Sender: TObject);

var

s, f, res, smo: iwpSignal;

p: tgraphstruct;

interval, interval1, interval2, interval3, interval4, p1, p2: point2d;

c: double;

src: csrc;

frame: TRZDTareFrame;

e1fore2, mo: double;

i, j: integer;

b: boolean;

n: iwpnode;

str: string;

vt, gt: array [0 .. 3] of double;

ifile:tinifile;

begin

if not CheckSrc then

begin

PrepareDataforMatrix;

exit;

end;

JournalLB.clear;

// Считаем E1

src := vc\_src;

frame := VC\_Frame;

s := GetSignal(src, frame.S1Cbox.Text, NullCb.Checked, b);

f := GetSignal(src, frame.VFCbox.Text, NullCb.Checked, b);

interval := frame.GetInterval;

interval1 := interval;

JournalLB.AddItem('Интервал: ' + formatstr(interval.x,

c\_digits) + '..' + formatstr(interval.y, c\_digits), nil);

if not checkSignalInterval(s, interval) then

exit;

if not checkSignalInterval(f, interval) then

exit;

e1[0] := EvalEi(interval, s, f, 'E1\_1');

s := GetSignal(src, frame.S2Cbox.Text, NullCb.Checked, b);

e1[1] := EvalEi(interval, s, f, 'E1\_2');

s := GetSignal(src, frame.S3Cbox.Text, NullCb.Checked, b);

e1[2] := EvalEi(interval, s, f, 'E1\_3');

s := GetSignal(src, frame.S4Cbox.Text, NullCb.Checked, b);

e1[3] := EvalEi(interval, s, f, 'E1\_4');

for i := 0 to 3 do

begin

if e1[i].result = false then

begin

JournalLB.AddItem('Не удалось расчитать e1', nil);

exit;

end;

end;

// нужно ли учитывать знак fmax?

e1norm.i1 := e1[0].ei / e1[0].fmax;

e1norm.i2 := e1[1].ei / e1[1].fmax;

e1norm.i3 := e1[2].ei / e1[2].fmax;

e1norm.i4 := e1[3].ei / e1[3].fmax;

LinkEiSignals(e1);

addEiJournal('e1', e1, e1norm);

f := GetIntervalSignal(interval, f);

vt[0] := getAbsMax(f.MaxY, f.MinY);

f := GetSignal(src, frame.HFCbox.Text, NullCb.Checked, b);

f := GetIntervalSignal(interval, f);

gt[0] := getAbsMax(f.MaxY, f.MinY);

JournalLB.AddItem('', nil);

// Считаем E2

src := h\_src;

frame := H\_Frame;

s := GetSignal(src, frame.S1Cbox.Text, NullCb.Checked, b);

f := GetSignal(src, frame.HFCbox.Text, NullCb.Checked, b);

interval := frame.GetInterval;

interval2 := interval;

JournalLB.AddItem('Интервал: ' + formatstr(interval.x,

c\_digits) + '..' + formatstr(interval.y, c\_digits), nil);

if not checkSignalInterval(s, interval) then

exit;

if not checkSignalInterval(f, interval) then

exit;

e2[0] := EvalEi(interval, s, f, 'E2\_1');

s := GetSignal(src, frame.S2Cbox.Text, NullCb.Checked, b);

e2[1] := EvalEi(interval, s, f, 'E2\_2');

s := GetSignal(src, frame.S3Cbox.Text, NullCb.Checked, b);

e2[2] := EvalEi(interval, s, f, 'E2\_3');

s := GetSignal(src, frame.S4Cbox.Text, NullCb.Checked, b);

e2[3] := EvalEi(interval, s, f, 'E2\_4');

for i := 0 to 3 do

begin

if e2[i].result = false then

begin

JournalLB.AddItem('Не удалось расчитать e2', nil);

exit;

end;

end;

s := GetSignal(src, frame.VFCbox.Text, NullCb.Checked, b);

f := GetSignal(src, frame.HFCbox.Text, NullCb.Checked, b);

EvalFzFy(interval, s, f, 'Fz(Fy)');

c := 1 / e2[0].fmax;

for i := 0 to 3 do

begin

e1fore2 := fi(e1[i].polyC, m\_F2zFyValue);

// e1fore2 := fi(e1[i].polyc, m\_F2zFyMaxX);

case i of

0:

e2norm.i1 := c \* (e2[i].ei - e1fore2);

1:

e2norm.i2 := c \* (e2[i].ei - e1fore2);

2:

e2norm.i3 := c \* (e2[i].ei - e1fore2);

3:

e2norm.i4 := c \* (e2[i].ei - e1fore2);

end;

end;

LinkEiSignals(e2);

addEiJournal('e2', e2, e2norm);

f := GetSignal(src, frame.VFCbox.Text, NullCb.Checked, b);

f := GetIntervalSignal(interval, f);

vt[1] := getAbsMax(f.MaxY, f.MinY); ;

f := GetSignal(src, frame.HFCbox.Text, NullCb.Checked, b);

f := GetIntervalSignal(interval, f);

gt[1] := getAbsMax(f.MaxY, f.MinY); ;

JournalLB.AddItem('', nil);

// Считаем E3

src := hout\_src;

frame := Hout\_Frame;

s := GetSignal(src, frame.S1Cbox.Text, NullCb.Checked, b);

f := GetSignal(src, frame.VFCbox.Text, NullCb.Checked, b);

interval := frame.GetInterval;

interval3 := interval;

JournalLB.AddItem('Интервал: ' + formatstr(interval.x,

c\_digits) + '..' + formatstr(interval.y, c\_digits), nil);

if not checkSignalInterval(s, interval) then

exit;

if not checkSignalInterval(f, interval) then

exit;

e3[0] := EvalEi(interval, s, f, 'E3\_1');

s := GetSignal(src, frame.S2Cbox.Text, NullCb.Checked, b);

e3[1] := EvalEi(interval, s, f, 'E3\_2');

s := GetSignal(src, frame.S3Cbox.Text, NullCb.Checked, b);

e3[2] := EvalEi(interval, s, f, 'E3\_3');

s := GetSignal(src, frame.S4Cbox.Text, NullCb.Checked, b);

e3[3] := EvalEi(interval, s, f, 'E3\_4');

for i := 0 to 3 do

begin

if e3[i].result = false then

begin

JournalLB.AddItem('Не удалось расчитать e3', nil);

exit;

end;

end;

c := 1 / e3[0].fmax;

e3norm.i1 := c \* (e3[0].ei);

e3norm.i2 := c \* (e3[1].ei);

e3norm.i3 := c \* (e3[2].ei);

e3norm.i4 := c \* (e3[3].ei);

LinkEiSignals(e3);

addEiJournal('доп. e3', e3, e3norm);

f := GetSignal(src, frame.VFCbox.Text, NullCb.Checked, b);

f := GetIntervalSignal(interval, f);

vt[2] := getAbsMax(f.MaxY, f.MinY); ;

f := GetSignal(src, frame.HFCbox.Text, NullCb.Checked, b);

f := GetIntervalSignal(interval, f);

gt[2] := getAbsMax(f.MaxY, f.MinY); ;

JournalLB.AddItem('', nil);

// Считаем E4

src := hin\_src;

frame := Hin\_Frame;

s := GetSignal(src, frame.S1Cbox.Text, NullCb.Checked, b);

f := GetSignal(src, frame.VFCbox.Text, NullCb.Checked, b);

interval := frame.GetInterval;

interval4 := interval;

JournalLB.AddItem('Интервал: ' + formatstr(interval.x,

c\_digits) + '..' + formatstr(interval.y, c\_digits), nil);

if not checkSignalInterval(s, interval) then

exit;

if not checkSignalInterval(f, interval) then

exit;

e4[0] := EvalEi(interval, s, f, 'E4\_1');

s := GetSignal(src, frame.S2Cbox.Text, NullCb.Checked, b);

e4[1] := EvalEi(interval, s, f, 'E4\_2');

s := GetSignal(src, frame.S3Cbox.Text, NullCb.Checked, b);

e4[2] := EvalEi(interval, s, f, 'E4\_3');

s := GetSignal(src, frame.S4Cbox.Text, NullCb.Checked, b);

e4[3] := EvalEi(interval, s, f, 'E4\_4');

for i := 0 to 3 do

begin

if e4[i].result = false then

begin

JournalLB.AddItem('Не удалось расчитать e4', nil);

exit;

end;

end;

c := 1 / e4[0].fmax;

e4norm.i1 := c \* (e4[0].ei);

e4norm.i2 := c \* (e4[1].ei);

e4norm.i3 := c \* (e4[2].ei);

e4norm.i4 := c \* (e4[3].ei);

LinkEiSignals(e4);

addEiJournal('доп. e4 ', e4, e4norm);

f := GetSignal(src, frame.VFCbox.Text, NullCb.Checked, b);

f := GetIntervalSignal(interval, f);

vt[3] := getAbsMax(f.MaxY, f.MinY); ;

f := GetSignal(src, frame.HFCbox.Text, NullCb.Checked, b);

f := GetIntervalSignal(interval, f);

gt[3] := getAbsMax(f.MaxY, f.MinY); ;

JournalLB.AddItem('', nil);

// Вычисляем окончательно e3

c := 1 / (2 \* Hout\_D.FloatNum);

e3norm.i1 := c \* (e3norm.i1 - e4norm.i1);

e3norm.i2 := c \* (e3norm.i2 - e4norm.i2);

e3norm.i3 := c \* (e3norm.i3 - e4norm.i3);

e3norm.i4 := c \* (e3norm.i4 - e4norm.i4);

JournalLB.AddItem('e3norm: ' + formatstr(e3norm.i1,

c\_digits) + '; ' + formatstr(e3norm.i2, c\_digits) + '; ' + formatstr

(e3norm.i3, c\_digits) + '; ' + formatstr(e3norm.i4, c\_digits), nil);

E[0, 0] := e1norm.i1;

E[0, 1] := e1norm.i2;

E[0, 2] := e1norm.i3;

E[0, 3] := e1norm.i4;

E[1, 0] := e2norm.i1;

E[1, 1] := e2norm.i2;

E[1, 2] := e2norm.i3;

E[1, 3] := e2norm.i4;

E[2, 0] := e3norm.i1;

E[2, 1] := e3norm.i2;

E[2, 2] := e3norm.i3;

E[2, 3] := e3norm.i4;

buildG;

frame := VC\_Frame;

m\_matrix := TRZDMatrix.create;

m\_matrix.date:=datetostr(now);

for i := 0 to 3 do

begin

case i of

0:

begin

src := vc\_src;

m\_matrix.sensors[i].t1t2 := interval1;

frame := VC\_Frame;

end;

1:

begin

src := h\_src;

m\_matrix.sensors[i].t1t2 := interval2;

frame := H\_Frame;

end;

2:

begin

src := hout\_src;

m\_matrix.sensors[i].t1t2 := interval3;

frame := Hout\_Frame;

end;

3:

begin

src := hin\_src;

m\_matrix.sensors[i].t1t2 := interval4;

frame := Hin\_Frame;

end;

end;

m\_matrix.vtmax[i] := vt[i];

m\_matrix.gtmax[i] := gt[i];

m\_matrix.sensors[i].mf := ExtractFileName(frame.Path.Text);

m\_matrix.sensors[i].vt := frame.VFCbox.Text;

m\_matrix.sensors[i].gt := frame.HFCbox.Text;

m\_matrix.sensors[i].s1 := frame.S1Cbox.Text;

m\_matrix.sensors[i].s2 := frame.S2Cbox.Text;

m\_matrix.sensors[i].s3 := frame.S3Cbox.Text;

m\_matrix.sensors[i].s4 := frame.S4Cbox.Text;

end;

m\_matrix.poly := PolySE.Value;

m\_matrix.cut := CutSE.Value;

m\_matrix.region := RegionCB.Text;

m\_matrix.useSNames := not useSignalPrefix.Checked;

for i := 0 to 3 do

begin

for j := 0 to 2 do

begin

m\_matrix.m[i, j] := G[i, j];

end;

end;

addMatrix(m\_matrix);

ShowMatrixList;

// проверка качества работы матрицы

if m\_matrix.ApplyMatrix(vc\_src) then

begin

n := WP.Get(vc\_src.name + '/RZDForce') as iwpnode;

str := extractfiledir(vc\_src.merafile.filename);

WP.SaveUSML(n.absolutepath, str + '\RZDforce.mera');

ifile:=tinifile.Create(str + '\RZDForce\RZDforce.mera');

ifile.WriteString('RZDHexa','VT',m\_matrix.lastn1);

ifile.WriteString('RZDHexa','GT',m\_matrix.lastn2);

ifile.WriteString('RZDHexa','M',m\_matrix.lastn3);

ifile.Destroy;

CheckMatrix(m\_matrix, vc\_src);

end

else

exit;

if m\_matrix.ApplyMatrix(h\_src) then

begin

n := WP.Get(h\_src.name + '/RZDForce') as iwpnode;

str := extractfiledir(h\_src.merafile.filename);

WP.SaveUSML(n.absolutepath, str + '\RZDforce.mera');

ifile:=tinifile.Create(str + '\RZDForce\RZDforce.mera');

ifile.WriteString('RZDHexa','VT',m\_matrix.lastn1);

ifile.WriteString('RZDHexa','GT',m\_matrix.lastn2);

ifile.WriteString('RZDHexa','M',m\_matrix.lastn3);

ifile.Destroy;

CheckMatrix(m\_matrix, h\_src);

end

else

exit;

if m\_matrix.ApplyMatrix(hout\_src) then

begin

n := WP.Get(hout\_src.name + '/RZDForce') as iwpnode;

str := extractfiledir(hout\_src.merafile.filename);

WP.SaveUSML(n.absolutepath, str + '\RZDforce.mera');

ifile:=tinifile.Create(str + '\RZDForce\RZDforce.mera');

ifile.WriteString('RZDHexa','VT',m\_matrix.lastn1);

ifile.WriteString('RZDHexa','GT',m\_matrix.lastn2);

ifile.WriteString('RZDHexa','M',m\_matrix.lastn3);

ifile.Destroy;

CheckMatrix(m\_matrix, hout\_src);

end

else

exit;

if m\_matrix.ApplyMatrix(hin\_src) then

begin

n := WP.Get(hin\_src.name + '/RZDForce') as iwpnode;

str := extractfiledir(hin\_src.merafile.filename);

WP.SaveUSML(n.absolutepath, str + '\RZDforce.mera');

ifile:=tinifile.Create(str + '\RZDForce\RZDforce.mera');

ifile.WriteString('RZDHexa','VT',m\_matrix.lastn1);

ifile.WriteString('RZDHexa','GT',m\_matrix.lastn2);

ifile.WriteString('RZDHexa','M',m\_matrix.lastn3);

ifile.Destroy;

CheckMatrix(m\_matrix, hin\_src);

end

else

exit;

WP.refresh;

end;

procedure TRZDFrm.CheckMatrix(m: TRZDMatrix; src: csrc);

var

p: tgraphstruct;

sig, f, res, res1, res2: iwpSignal;

lcaption: string;

b: boolean;

frame: TRZDTareFrame;

i: integer;

lsrc: csrc;

vtmax, gtmax, V: double;

n: iwpnode;

begin

for i := 0 to 3 do

begin

case i of

0:

begin

lsrc := vc\_src;

frame := VC\_Frame;

lcaption := 'Сравнение для нагр-ия верт. силой'

end;

1:

begin

lsrc := h\_src;

frame := H\_Frame;

lcaption :=

'Сравнение для нагр-ия верт. силой с смещением по горизонтали'

end;

2:

begin

lsrc := hout\_src;

frame := Hout\_Frame;

lcaption := 'Сравнение для нагр-ия верт. силой с смещением наружу'

end;

3:

begin

lsrc := hin\_src;

frame := Hin\_Frame;

lcaption := 'Сравнение для нагр-ия верт. силой с смещением внутрь'

end;

end;

if src = lsrc then

break;

end;

vtmax := m.vtmax[i];

gtmax := m.gtmax[i];

//n := WP.Get(src.name + '/RZDForce/' + 'Vt\_' + inttostr(m\_matrix.cut)

// + '\_' + m.region) as iwpnode;

n:=m.n1;

if n = nil then

exit;

sig := n.Reference as iwpSignal;

f := GetSignal(src, frame.VFCbox.Text, false, b);

JournalLB.AddItem(lcaption, nil);

JournalLB.AddItem('Vtmax=' + formatstr(vtmax, c\_digits), nil);

//res := BuildCompare(sig, f, GetAbsRelFmax(vtmax, FmaxEdit.FloatNum, FmaxAbsFe.FloatNum), src.name + '/RZDForce/', res1, res2);

res := BuildCompare(sig, f, FmaxAbsFe.FloatNum, FmaxEdit.FloatNum, src.name + '/RZDForce/', res1, res2);

p := createline(sig);

IWPGraphs(WP.GraphAPI).SetPageDim(p.hpage, PAGE\_DM\_TABLE, 2, 2);

p := createline(f, p.hgraph, p.haxis);

//p := createline(res, p.hpage);

// res1 - сигнал разница между исходным и расчитанным

p := createline(res1, p.hpage);

// res - сигнал сравнение

//p := createline(res, p.hgraph, p.haxis);

// res2 - знаменатель

p := createline(res2, p.hgraph, p.haxis);

color:=clred;

IWPGraphs(WP.GraphAPI).SetLineOpt(p.hline, LNOPT\_color+LNOPT\_WIDTH,

LNOPT\_color+LNOPT\_WIDTH, 3, color);

V := GetRMS(res);

//IWPGraphs(WP.GraphAPI).AddComment(p.hgraph, lcaption + ' F=Vt ' + 'D= ' + formatstr(V, c\_digits), 15, 5, c\_CommentDx\*9, c\_CommentDy\*1);

IWPGraphs(WP.GraphAPI).AddComment(p.hgraph, lcaption + ' F=Vt', 15, 5, c\_CommentDx\*9, c\_CommentDy\*1);

n:=m.n2;

sig := n.Reference as iwpSignal;

//sig := (WP.Get(src.name + '/RZDForce/' + 'Gt\_' + inttostr(m\_matrix.cut)

// + '\_' + m.region) as iwpnode).Reference as iwpSignal;

f := GetSignal(src, frame.HFCbox.Text, false, b);

// f:=GetIntervalSignal(m.sensors[i].t1t2,f);

JournalLB.AddItem('Gtmax=' + formatstr(gtmax, c\_digits), nil);

res := BuildCompare(sig, f, GtmaxAbsFE.FloatNum, GtmaxEdit.FloatNum, src.name + '/RZDForce/', res1, res2);

p := createline(sig, p.hpage);

p := createline(f, p.hgraph, p.haxis);

//p := createline(res, p.hpage);

p := createline(res1, p.hpage);

p := createline(res2, p.hgraph, p.haxis);

color:=clred;

IWPGraphs(WP.GraphAPI).SetLineOpt(p.hline, LNOPT\_color+LNOPT\_WIDTH,

LNOPT\_color+LNOPT\_WIDTH, 3, color);

V := GetRMS(res);

//IWPGraphs(WP.GraphAPI).AddComment(p.hgraph, lcaption + ' F=Gt ' + 'D= ' + formatstr(V, c\_digits), 15, 5, c\_CommentDx\*9, c\_CommentDy\*1);

IWPGraphs(WP.GraphAPI).AddComment(p.hgraph, lcaption + ' F=Gt', 15, 5, c\_CommentDx\*9, c\_CommentDy\*1);

end;

procedure TRZDFrm.addgistproc(start, shift: integer; lvl, noise: double;

proc: boolean);

var

gist: cGist;

li: TListItem;

begin

gist := cGist.create;

gist.start := start;

gist.shift := shift;

gist.proc := proc;

gist.lvl := lvl;

gist.noise := noise;

li := GistLV.Items.Add;

li.data := gist;

GistLV.SetSubItemByColumnName('Первый импульс', inttostr(gist.start), li);

GistLV.SetSubItemByColumnName('Пропуск', inttostr(gist.shift), li);

end;

procedure TRZDFrm.cleargist;

var

i: integer;

li: TListItem;

begin

for i := 0 to GistLV.Items.count - 1 do

begin

li := GistLV.Items[i];

cGist(li.data).Destroy;

end;

GistLV.clear;

end;

procedure TRZDFrm.AddGistClick(Sender: TObject);

begin

addgistproc(GistStart.Value, GistShiftSE.Value, GistTrigFE.FloatNum,

GistNoiseFE.FloatNum, GistProcCB.Checked);

end;

procedure TRZDFrm.addMatrix(m: TRZDMatrix);

var

i: integer;

lm: TRZDMatrix;

begin

for i := 0 to MatrixList.count - 1 do

begin

lm := TRZDMatrix(MatrixList.Items[i]);

if lm.GetIniName = m.GetIniName then

begin

lm.Destroy;

MatrixList.Items[i] := m;

exit;

end;

end;

MatrixList.Add(m);

end;

procedure TRZDFrm.ApplyBtnClick(Sender: TObject);

var

i: integer;

m: TRZDMatrix;

j: integer;

cb: TComboBox;

begin

for i := 0 to MatrixLV.Items.count - 1 do

begin

if MatrixLV.Items[i].Selected then

begin

m := TRZDMatrix(MatrixLV.Items[i].data);

m.m:=GetFormMatrix;

if Matrix\_CutSE.Text <> '' then

m.cut := Matrix\_CutSE.Value;

if MatrixRegSE.Text <> '' then

m.region := MatrixRegSE.Text;

if mat\_UseNamesCB.State <> cbGrayed then

m.useSNames := mat\_UseNamesCB.Checked;

if m.useSNames then

begin

for j := 0 to 3 do

begin

cb := GetSensCB(j);

if (cb.Text <> '\_') and (cb.Text <> '') then

begin

m.sets(j, cb.Text);

end;

end;

end;

end;

end;

ShowMatrixList;

end;

procedure TRZDFrm.CutSEChange(Sender: TObject);

var

str, str1: string;

b: boolean;

i: integer;

begin

str := m\_sectionPref + '\_' + inttostr(CutSE.Value);

str := lowercase(str);

for i := 0 to sectionCB.Items.count - 1 do

begin

str1 := lowercase(sectionCB.Items[i]);

if CheckPosSubstr(m\_sectionPref, str1) then

begin

str1 := getendnum(str1);

if strtoint(str1) = CutSE.Value then

begin

sectionCB.ItemIndex := i;

break;

b := true;

end;

end;

end;

sectionCBChange(nil);

end;

procedure TRZDFrm.DelGistClick(Sender: TObject);

var

li: TListItem;

begin

li := GistLV.Selected;

if li <> nil then

cGist(li.data).Destroy;

li.Delete;

end;

procedure TRZDFrm.DelMatrixBtnClick(Sender: TObject);

var

li: TListItem;

m: TRZDMatrix;

begin

if MatrixLV.Selected <> nil then

begin

li := MatrixLV.Selected;

m := MatrixList.Items[li.Index];

m.Destroy;

MatrixList.Delete(li.Index);

MatrixLV.DeleteSelected;

ShowMatrixList;

end;

end;

function TRZDFrm.GetSensorsNameWithPref(mf: string; sect: integer): tstringlist;

var

i: integer;

str, str1, str2, str3, str4: string;

s1, s2, s3, s4: boolean;

sections: tstringlist;

ifile: tinifile;

begin

result := tstringlist.create;

s1 := false;

s2 := false;

s3 := false;

s4 := false;

ifile := tinifile.create(mf);

sections := tstringlist.create;

ifile.ReadSections(sections);

for i := 0 to sections.count - 1 do

begin

str := sections.Strings[i];

if CheckPosSubstr(m\_D1Pref + inttostr(sect), str) then

begin

if not s1 then

begin

str1 := str;

s1 := true;

end;

end;

if CheckPosSubstr(m\_D2Pref + inttostr(sect), str) then

begin

if not s2 then

begin

str2 := str;

s2 := true;

end;

end;

if CheckPosSubstr(m\_D3Pref + inttostr(sect), str) then

begin

if not s3 then

begin

str3 := str;

s3 := true;

end;

end;

if CheckPosSubstr(m\_D4Pref + inttostr(sect), str) then

begin

if not s4 then

begin

str4 := str;

s4 := true;

end;

end;

end;

result.Add(str1);

result.Add(str2);

result.Add(str3);

result.Add(str4);

ifile.Destroy;

sections.Destroy;

end;

function TRZDFrm.GetSensorsNameWithPref(src: csrc; sect: integer): tstringlist;

var

i: integer;

str: string;

s1, s2, s3, s4: boolean;

begin

result := tstringlist.create;

s1 := false;

s2 := false;

s3 := false;

s4 := false;

for i := 0 to src.childCount - 1 do

begin

str := src.GetWPSignal(i).name;

if CheckPosSubstr(m\_D1Pref + inttostr(sect), str) then

begin

if not s1 then

begin

result.Add(str);

s1 := true;

end;

end;

if CheckPosSubstr(m\_D2Pref + inttostr(sect), str) then

begin

if not s2 then

begin

result.Add(str);

s2 := true;

end;

end;

if CheckPosSubstr(m\_D3Pref + inttostr(sect), str) then

begin

if not s3 then

begin

result.Add(str);

s3 := true;

end;

end;

if CheckPosSubstr(m\_D4Pref + inttostr(sect), str) then

begin

if not s4 then

begin

result.Add(str);

s4 := true;

end;

end;

end;

end;

procedure TRZDFrm.ApplyMatrix(s1, s2, s3, s4: iwpSignal; p\_g: d2array;

srcpath: string; sname: string; m:TRZDMatrix);

var

// максимальная частота опроса среди сигналов

V, v1, v2, v3, v4, DeltaX, maxFS, min, max: double;

interval: point2d;

// датчик с максимальной частотой дискретизации

i, start, stop: integer;

s, r1, r2, r3: iwpSignal;

folder: string;

begin

maxFS := s1.DeltaX;

s := s1;

if s2.DeltaX < maxFS then

begin

s := s2;

maxFS := s2.DeltaX;

end;

if s3.DeltaX < maxFS then

begin

s := s3;

maxFS := s3.DeltaX;

end;

if s4.DeltaX < maxFS then

begin

s := s4;

maxFS := s4.DeltaX;

end;

DeltaX := maxFS;

maxFS := 1 / maxFS;

// находим интервал обработки

interval.x := s1.MinX;

if s2.MinX > interval.x then

interval.x := s2.MinX;

if s3.MinX > interval.x then

interval.x := s3.MinX;

if s4.MinX > interval.x then

interval.x := s4.MinX;

// находим интервал обработки

interval.y := s1.MaxX;

if s2.MaxX < interval.y then

interval.y := s2.MaxX;

if s3.MaxX > interval.y then

interval.y := s3.MaxX;

if s4.MaxX > interval.y then

interval.y := s4.MaxX;

s1 := GetIntervalSignal(interval, s1);

s2 := GetIntervalSignal(interval, s2);

s3 := GetIntervalSignal(interval, s3);

s4 := GetIntervalSignal(interval, s4);

if s1.DeltaX < DeltaX then

begin

s1 := Resample(s1, maxFS);

end;

if s2.DeltaX < DeltaX then

begin

s2 := Resample(s2, maxFS);

end;

if s3.DeltaX < DeltaX then

begin

s3 := Resample(s3, maxFS);

end;

if s4.DeltaX < DeltaX then

begin

s4 := Resample(s4, maxFS);

end;

start := s.IndexOf(interval.x);

stop := s.IndexOf(interval.y);

r1 := s.Clone(start, stop - start) as iwpSignal;

r1.sname := 'Vt\_' + sname;

r2 := s.Clone(start, stop - start) as iwpSignal;

r2.sname := 'Gt\_' + sname;

r3 := s.Clone(start, stop - start) as iwpSignal;

r3.sname := 'M\_' + sname;

for i := 0 to r1.size - 1 do

begin

if i = 0 then

begin

v1 := s1.GetX(i);

v2 := s2.GetX(i);

v3 := s3.GetX(i);

v4 := s4.GetX(i);

min := v1;

max := v1;

if v2 > max then

max := v2;

if v3 > max then

max := v3;

if v4 > max then

max := v4;

if v2 < min then

min := v2;

if v3 < min then

min := v3;

if v4 < min then

min := v4;

JournalLB.AddItem('Максимальная несинхронность: ' + formatstr(max - min,

c\_digits), nil);

end;

v1 := s1.GetY(i);

v2 := s2.GetY(i);

v3 := s3.GetY(i);

v4 := s4.GetY(i);

// if i=32712 then

// begin

// v1:=339;v2:=302;v3:=171;v4:=196;

V := p\_g[0, 0] \* v1 + p\_g[1, 0] \* v2 + p\_g[2, 0] \* v3 + p\_g[3, 0] \* v4;

r1.SetY(i, V);

V := p\_g[0, 1] \* v1 + p\_g[1, 1] \* v2 + p\_g[2, 1] \* v3 + p\_g[3, 1] \* v4;

r2.SetY(i, V);

V := p\_g[0, 2] \* v1 + p\_g[1, 2] \* v2 + p\_g[2, 2] \* v3 + p\_g[3, 2] \* v4;

r3.SetY(i, V);

// end;

end;

folder := srcpath + '/RZDForce/';

m.n1:=WP.Link(folder, r1.sname, r1) as iwpnode;

m.lastn1:=(m.n1.Name);

m.n2:=WP.Link(folder, r2.sname, r2) as iwpnode;

m.lastn2:=(m.n2.Name);

m.n3:=WP.Link(folder, r3.sname, r3) as iwpnode;

m.lastn3:=(m.n3.Name);

end;

function TRZDFrm.buildG: d2array;

var

Et\_e, ident, Et: d2array;

begin

Et := transpondmatrix(E);

MultMatrix(Et, E, Et\_e);

Et\_e := Invers(Et\_e);

MultMatrix(Et\_e, Et, G);

result := G;

ShowG(G);

end;

procedure TRZDFrm.SelectIntervalCursorBtnClick(Sender: TObject);

var

i: integer;

G: tgraphstruct;

p: point2d;

fr: TRZDTareFrame;

begin

for i := 0 to 3 do

begin

G := graph[i];

p := GetGraphCursorX(G.hpage, G.hgraph);

case i of

0:

fr := VC\_Frame;

1:

fr := H\_Frame;

2:

fr := Hout\_Frame;

3:

fr := Hin\_Frame;

end;

fr.T1FE.FloatNum := p.x;

fr.T2FE.FloatNum := p.y;

end;

end;

procedure TRZDFrm.SelectIntervalGraphBtnClick(Sender: TObject);

var

i: integer;

G: tgraphstruct;

p: point2d;

fr: TRZDTareFrame;

begin

for i := 0 to 3 do

begin

G := graph[i];

if g.hgraph=0 then

begin

p.x:=0;

p.y:=10;

end

else

begin

p := GetGraphX(G.hgraph);

case i of

0:

fr := VC\_Frame;

1:

fr := H\_Frame;

2:

fr := Hout\_Frame;

3:

fr := Hin\_Frame;

end;

end;

fr.T1FE.FloatNum := p.x;

fr.T2FE.FloatNum := p.y;

end;

end;

procedure TRZDFrm.ShowG(p\_g: d2array);

begin

G11.FloatNum := p\_g[0, 0];

G12.FloatNum := p\_g[0, 1];

G13.FloatNum := p\_g[0, 2];

G21.FloatNum := p\_g[1, 0];

G22.FloatNum := p\_g[1, 1];

G23.FloatNum := p\_g[1, 2];

G31.FloatNum := p\_g[2, 0];

G32.FloatNum := p\_g[2, 1];

G33.FloatNum := p\_g[2, 2];

G41.FloatNum := p\_g[3, 0];

G42.FloatNum := p\_g[3, 1];

G43.FloatNum := p\_g[3, 2];

end;

function TRZDFrm.GetFormMatrix: d2array;

begin

setlength(result,4,3);

result[0, 0]:=G11.FloatNum;

result[0, 1]:=G12.FloatNum;

result[0, 2]:=G13.FloatNum;

result[1, 0]:=G21.FloatNum;

result[1, 1]:=G22.FloatNum;

result[1, 2]:=G23.FloatNum;

result[2, 0]:=G31.FloatNum;

result[2, 1]:=G32.FloatNum;

result[2, 2]:=G33.FloatNum;

result[3, 0]:=G41.FloatNum;

result[3, 1]:=G42.FloatNum;

result[3, 2]:=G43.FloatNum;

end;

procedure TRZDFrm.addEiJournal(p\_name: string; E: array of TEi; enorm: TVec4);

begin

JournalLB.AddItem(p\_name + ': ' + formatstr(E[0].ei,

c\_digits) + '; ' + formatstr(E[1].ei, c\_digits) + '; ' + formatstr

(E[2].ei, c\_digits) + '; ' + formatstr(E[3].ei, c\_digits), nil);

JournalLB.AddItem('Fmax: ' + formatstr(E[0].fmax, c\_digits), nil);

JournalLB.AddItem(p\_name + 'norm: ' + formatstr(enorm.i1,

c\_digits) + '; ' + formatstr(enorm.i2, c\_digits) + '; ' + formatstr

(enorm.i3, c\_digits) + '; ' + formatstr(enorm.i4, c\_digits), nil);

end;

procedure TRZDFrm.LinkEiSignals(E: array of TEi);

var

graph: tgraphstruct;

i, color: integer;

begin

// графика

graph.hpage := 0;

if LinkEvalWP.Checked then

begin

for i := 0 to 3 do

begin

E[i].nCloud := iwpnode(WP.Link('Signals/MNK/', E[i].cloud.sname,

E[i].cloud));

E[i].nPoly := iwpnode(WP.Link('Signals/MNK/', E[i].poly.sname,

E[i].poly));

// строим графику

if graph.hpage = 0 then

begin

graph := createline(iwpSignal(E[i].nCloud.Reference));

end

else

begin

graph := createline(iwpSignal(E[i].nCloud.Reference), graph.hpage);

end;

IWPGraphs(WP.GraphAPI).SetLineOpt(graph.hline, LNOPT\_ONLYPOINTS,

LNOPT\_ONLYPOINTS, 0, $00D2D5);

graph := createline(E[i].poly, graph.hgraph, graph.haxis);

// цвет тренда

color:=clred;

IWPGraphs(WP.GraphAPI).SetLineOpt(graph.hline, LNOPT\_color+LNOPT\_WIDTH,

LNOPT\_color+LNOPT\_WIDTH, 3, color);

end;

IWPGraphs(WP.GraphAPI).SetPageDim(graph.hpage, PAGE\_DM\_TABLE, 2, 2);

WP.refresh;

end;

end;

function TRZDFrm.BuildGistogram(s: iwpSignal; dx, lvl, noise: double;

start, shift: integer; folder, gistname: string): iwpSignal;

var

res: cP2dList;

p2: point2d;

findtrg: boolean;

i, lshift, j, ind, count: integer;

// экстремум на участке

localmax, localmax\_x, V, sum, min, max: double;

begin

res := cP2dList.create;

if abs(s.MaxY) < abs(s.MinY) then

s := Multiply(s, -1);

findtrg := false;

localmax := s.MinY;

for i := 0 to s.size - 1 do

begin

V := s.GetY(i);

if V > lvl then

begin

findtrg := true;

if V > localmax then

begin

localmax := V;

localmax\_x := s.GetX(i);

end;

end

else

begin

// проверяем вышли мы за уровень гистерезиса или нет

if findtrg then

begin

if V + noise < lvl then

begin

end

else

begin

p2.x := localmax\_x;

p2.y := localmax;

localmax := s.MinY;

res.Add(p2);

findtrg := false;

end;

end;

end;

end;

result := WP.CreateSignal(VT\_R4) as iwpSignal;

result.sname := gistname;

//V := (s.MaxY - s.MinY) / dx;

V := s.MaxY / dx;

if round(V) < V then

V := round(V) + 1

else

V := round(V);

result.size := round(V) \* 2;

//result.StartX := s.MinY;

result.StartX := 0;

result.StartX := -1\*dx/2;

result.DeltaX := dx;

result.SetProperty('Gist', '1');

lshift := 0;

sum := 0;

// не случайно! Обновляется внутри цикла

min := s.MaxY;

max := s.MinY;

count:=0;

for j := start to res.count - 1 do

begin

if lshift = shift then

begin

p2 := res.getP(j);

sum := p2.y + sum;

if p2.y > max then

max := p2.y;

if p2.y < min then

min := p2.y;

ind := round((p2.y - result.StartX) / dx);

V := result.GetY(ind);

result.SetY(ind, V + 1);

lshift := 0;

inc(count);

end

else

begin

inc(lshift);

end;

end;

result.SetProperty('GistN', inttostr(count));

result.SetProperty('MO', floattostr(sum / res.count));

result.SetProperty('GistMax', floattostr(max));

result.SetProperty('GistMin', floattostr(min));

// нормировка сигнала

for i := 0 to result.size - 1 do

begin

V := result.GetY(i);

if V <> 0 then

begin

//result.SetY(i, V / (res.count \* dx));

result.SetY(i, V / (count \* dx));

end;

end;

WP.Link(folder, result.sname, result);

res.Destroy;

end;

function TRZDFrm.BuildGistogram(s: iwpSignal; dx: double;

folder: string): iwpSignal;

var

min, max, from, range: double;

gist, sig: iwpSignal;

bres, findtrg: boolean;

ind: integer;

i, j: integer;

V, localmax, localmax\_x, lvl, noise: double;

res: cP2dList;

p2: point2d;

graph: tgraphstruct;

G: cGist;

lshift, lvltype: integer;

begin

if GistLV.Items.count = 0 then

exit;

res := cP2dList.create;

sig := s;

if abs(s.MaxY) < abs(s.MinY) then

s := Multiply(s, -1);

//min := s.MinY;

min := s.MinY;

max := s.MaxY;

from := s.MinX;

if GistProcCB.Checked then

lvltype := 1

else

lvltype := 0;

lvl := GistTrigFE.FloatNum;

noise := GistNoiseFE.FloatNum;

case lvltype of

c\_abs:

begin

end;

c\_Percent:

begin

range := max - min;

lvl := range \* lvl + min;

noise := range \* noise;

end;

end;

findtrg := false;

localmax := s.MinY;

for i := 0 to s.size - 1 do

begin

V := s.GetY(i);

if V > lvl then

begin

findtrg := true;

if V > localmax then

begin

localmax := V;

localmax\_x := s.GetX(i);

end;

end

else

begin

// проверяем вышли мы за уровень гистерезиса или нет

if findtrg then

begin

if V + noise < lvl then

begin

end

else

begin

p2.x := localmax\_x;

p2.y := localmax;

localmax := s.MinY;

res.Add(p2);

findtrg := false;

end;

end;

end;

end;

for i := 0 to GistLV.Items.count - 1 do

BEGIN

G := cGist(GistLV.Items[i].data);

result := WP.CreateSignal(VT\_R4) as iwpSignal;

result.sname := s.sname + '\_Gist\_' + inttostr(G.start) + '\_' + inttostr

(G.shift);

//V := (s.MaxY - s.MinY) / dx;

V := s.MaxY / dx;

if round(V) < V then

V := round(V) + 1

else

V := round(V);

result.size := round(V) \* 2;

//result.StartX := s.MinY;

result.StartX := 0;

result.DeltaX := dx;

result.SetProperty('Gist', '1');

lshift := 0;

for j := G.start to res.count - 1 do

begin

if lshift = G.shift then

begin

p2 := res.getP(j);

ind := round((p2.y - result.StartX) / dx);

V := result.GetY(ind);

result.SetY(ind, V + 1);

lshift := 0;

end

else

begin

inc(lshift);

end;

end;

WP.Link(folder, result.sname, result);

graph := createline(sig);

graph := createline(result, graph.hpage);

END;

WP.refresh;

res.Destroy;

end;

function TRZDFrm.GetAbsRelFmax(fmax, rel, p\_abs:double):double;

var

v:double;

begin

v:=abs(fmax\*rel);

if v<p\_abs then

begin

result:=p\_abs;

end

else

begin

result:=v;

end;

end;

function getmax(v1,v2:double):double;

begin

result:=v1;

if v2>v1 then

result:=v2;

end;

// base - тарировочная сила, s1 - расчитанная

function TRZDFrm.BuildCompare(s1, base: iwpSignal; fmax, rel: double;

resfolder: string; var res1, res2:iwpsignal): iwpSignal;

var

// максимальная частота опроса среди сигналов

DeltaX, maxFS, min, max, lmax, V: double;

interval: point2d;

// датчик с максимальной частотой дискретизации

i, start, stop: integer;

s, res: iwpSignal;

begin

maxFS := s1.DeltaX;

s := s1;

if base.DeltaX < maxFS then

begin

s := base;

maxFS := base.DeltaX;

end;

DeltaX := maxFS;

maxFS := 1 / maxFS;

// находим интервал обработки

interval.x := s1.MinX;

if base.MinX > interval.x then

interval.x := base.MinX;

// находим интервал обработки

interval.y := s1.MaxX;

if base.MaxX < interval.y then

interval.y := base.MaxX;

s1 := GetIntervalSignal(interval, s1);

base := GetIntervalSignal(interval, base);

if s1.DeltaX < DeltaX then

begin

s1 := Resample(s1, maxFS);

end;

start := s.IndexOf(interval.x);

stop := s.IndexOf(interval.y);

V := base.MaxY - base.MinY;

max := abs(fmax);

res := s.Clone(start, stop - start) as iwpSignal;

res.sname := s1.sname + '\_' + base.sname + '\_compare';

res1 := s.Clone(start, stop - start) as iwpSignal;

res1.sname := s1.sname + '\_Delta';

res2 := s.Clone(start, stop - start) as iwpSignal;

res2.sname := s1.sname + '\_AbsTol';

for i := 0 to s1.size - 1 do

begin

V := abs(s1.GetY(i) - base.GetY(i));

if max < abs(base.GetY(i))\*rel then

lmax := abs(base.GetY(i))\*rel

else

lmax:=max;

//V := V / lmax;

V := V / getmax(abs(base.GetY(i)),v\_MaxF);

res.SetY(i, V);

res1.SetY(i, abs(s1.GetY(i) - base.GetY(i)));

res2.SetY(i, lmax);

end;

WP.Link(resfolder, res.sname, res);

WP.Link(resfolder, res1.sname, res1);

WP.Link(resfolder, res2.sname, res2);

result := res;

end;

function TRZDFrm.GetCalibrFolder: string;

var

L: integer;

str: string;

begin

L := length(BaseFolderEdit.Text);

str := BaseFolderEdit.Text;

if BaseFolderEdit.Text[L] = '\' then

setlength(str, L - 1);

result := str + '\Calibr';

end;

procedure TRZDFrm.ReadRegionFolders;

var

list: tstringlist;

i: integer;

laststr: string;

begin

list := tstringlist.create;

// сканируемый каталог, маска, список результатов

FindFolders(GetCalibrFolder, '\*' + m\_regionPref + '\*', list, 1);

laststr := RegionCB.Text;

RegionCB.clear;

for i := 0 to list.count - 1 do

begin

RegionCB.AddItem(ExtractFileName(list.Strings[i]), nil);

end;

if RegionCB.Items.count > 0 then

begin

RegionCB.ItemIndex := 0;

end;

for i := 0 to RegionCB.Items.count - 1 do

begin

if RegionCB.Items[i] = laststr then

begin

RegionCB.ItemIndex := i;

break;

end;

end;

RegionCBChange(nil);

list.Destroy;

end;

procedure TRZDFrm.ReadSegments(RegionPath: string);

var

list: tstringlist;

i: integer;

sectiontext: string;

begin

list := tstringlist.create;

// сканируемый каталог, маска, список результатов

FindFolders(RegionPath, '\*' + m\_sectionPref + '\*', list, 1);

sectiontext := sectionCB.Text;

sectionCB.clear;

for i := 0 to list.count - 1 do

begin

sectionCB.AddItem(ExtractFileName(list.Strings[i]), nil);

end;

for i := 0 to sectionCB.Items.count - 1 do

begin

if i = 0 then

sectionCB.ItemIndex := 0;

if sectionCB.Items[i] = sectiontext then

begin

sectionCB.ItemIndex := i;

break;

end;

end;

GetNumFromSectionCB;

sectionCBChange(nil);

list.Destroy;

end;

Procedure TRZDFrm.GetNumFromSectionCB;

var

str: string;

begin

str := getendnum(sectionCB.Text);

if isDigit(str) then

begin

CutSE.Value := strtoint(str);

end;

end;

// заполняем пути к сигналам калибровки

procedure TRZDFrm.ReadTests(segmentpath: string);

var

list: tstringlist;

i: integer;

begin

list := tstringlist.create;

// сканируемый каталог, маска, список результатов

FindFileext('\*.mera', segmentpath, 2, list);

VC\_Frame.Path.clear;

H\_Frame.Path.clear;

Hin\_Frame.Path.clear;

Hout\_Frame.Path.clear;

for i := 0 to list.count - 1 do

begin

if useSignalPrefix.Checked then

begin

if CheckPosSubstr(m\_VCPref, list.Strings[i]) then

VC\_Frame.Path.AddItem(list.Strings[i], nil);

if CheckPosSubstr(m\_H\_Pref, list.Strings[i]) then

H\_Frame.Path.AddItem(list.Strings[i], nil);

if CheckPosSubstr(m\_Hin\_Pref, list.Strings[i]) then

Hin\_Frame.Path.AddItem(list.Strings[i], nil);

if CheckPosSubstr(m\_Hout\_Pref, list.Strings[i]) then

Hout\_Frame.Path.AddItem(list.Strings[i], nil);

end

else

begin

VC\_Frame.Path.AddItem(list.Strings[i], nil);

H\_Frame.Path.AddItem(list.Strings[i], nil);

Hin\_Frame.Path.AddItem(list.Strings[i], nil);

Hout\_Frame.Path.AddItem(list.Strings[i], nil);

end;

end;

if VC\_Frame.Path.Items.count > 0 then

VC\_Frame.Path.ItemIndex := 0;

if H\_Frame.Path.Items.count > 0 then

H\_Frame.Path.ItemIndex := 0;

if Hin\_Frame.Path.Items.count > 0 then

Hin\_Frame.Path.ItemIndex := 0;

if Hout\_Frame.Path.Items.count > 0 then

Hout\_Frame.Path.ItemIndex := 0;

list.Destroy;

FillVCCB;

FillHCB;

FillHoutCB;

FillHinCB;

UpdateIPortion;

// обновляем статус кнопки Загрузить/Создать. Если прописан новый замер то src сбрасывается

VC\_FramePathChange(nil);

H\_FramePathChange(nil);

Hin\_FramePathChange(nil);

Hout\_FramePathChange(nil);

CheckSrc;

end;

function GetRegPath(base: string; reg: string): string;

begin

result := reg;

if pos('\', result) < 1 then

begin

if base[length(base)] = '\' then

result := base + result

else

result := base + '\' + result;

end;

end;

function TRZDFrm.GetregionPath: string;

begin

result := GetRegPath(BaseFolderEdit.Text, RegionCB.Text);

end;

procedure TRZDFrm.RegionCBChange(Sender: TObject);

var

num,Path: string;

begin

Path := GetRegPath(GetCalibrFolder, RegionCB.Text);

ReadSegments(Path);

num:=getendnum(RegionCB.Text);

if isDigit(num) then

begin

RegionSE.Value:=strtoint(num);

end;

end;

procedure TRZDFrm.RegionSEChange(Sender: TObject);

var

str, str1: string;

b: boolean;

i: integer;

begin

str := m\_regionPref + '\_' + inttostr(RegionSE.Value);

str := lowercase(str);

for i := 0 to RegionCB.Items.count - 1 do

begin

str1 := lowercase(RegionCB.Items[i]);

if CheckPosSubstr(m\_regionPref, str1) then

begin

str1 := getendnum(str1);

if strtoint(str1) = RegionSE.Value then

begin

RegionCB.ItemIndex := i;

break;

b := true;

end;

end;

end;

RegionCBChange(nil);

end;

procedure TRZDFrm.ReportCBClick(Sender: TObject);

begin

if ReportCB.Checked then

begin

OpenReportCB.enabled:=true;

end

else

begin

OpenReportCB.enabled:=false;

OpenReportCB.Checked:=false;

end;

end;

procedure TRZDFrm.sectionCBChange(Sender: TObject);

var

num,Path, region: string;

begin

Path := sectionCB.Text;

if pos('\', Path) < 1 then

begin

region := GetRegPath(GetCalibrFolder, RegionCB.Text) + '\';

end;

if ChangeTestByCutCB.Checked then

begin

region := region + Path;

ReadTests(region);

end

else

begin

ReadTests(region);

end;

num:=getendnum(sectionCB.Text);

if isDigit(num) then

begin

cutse.Value:=strtoint(num);

end;

end;

procedure TRZDFrm.TestPathBtnClick(Sender: TObject);

begin

if OpenDialog1.Execute(0) then

begin

TestPath.Text := OpenDialog1.filename;

end;

end;

procedure TRZDFrm.TestPathChange(Sender: TObject);

begin

FillMatrixSensorsCB;

ShowMatrixList;

// str:=RelativePathToAbsolute(BaseFolderEdit.Text,ExportMatrixPath.Text);

TestPath.Hint := gettestpath;

CheckFolderComponent(TestPath, TestPath.Hint, false);

end;

procedure TRZDFrm.TestsFolderEChange(Sender: TObject);

begin

UpdateTestsFolder;

end;

procedure TRZDFrm.Timer1Timer(Sender: TObject);

begin

Timer1.Enabled := false;

UpdateDB(f\_updFolder);

end;

function TRZDFrm.gettestpath: string;

begin

result := RelativePathToAbsolute(GetTestsFolder, TestPath.Text);

end;

procedure TRZDFrm.TrendPortionFEChange(Sender: TObject);

begin

UpdateIPortion;

end;

procedure TRZDFrm.TrendPortionIEChange(Sender: TObject);

begin

UpdateFPortion;

end;

procedure TRZDFrm.useSignalPrefixClick(Sender: TObject);

begin

CheckBaseFolder;

end;

procedure TRZDFrm.createEvents;

begin

m\_wpMng.Events.AddEvent('RZD\_Addline', E\_OnAddLine, OnAddLine);

m\_wpMng.Events.AddEvent('RZD\_Addline', E\_OnDestroySignal, OnDestroySignal);

m\_wpMng.Events.AddEvent('RZD\_DestroySrc', E\_OnDestroySrc, OnDestroySrc);

end;

procedure TRZDFrm.destroyEvents;

begin

m\_wpMng.Events.removeEvent(OnAddLine, E\_OnAddLine);

m\_wpMng.Events.removeEvent(OnDestroySignal, E\_OnDestroySignal);

end;

procedure TRZDFrm.DrawGraphBtnClick(Sender: TObject);

var

i: integer;

fr: TRZDTareFrame;

p: point2d;

page: integer;

src: csrc;

str: array [0 .. 3] of string;

begin

page := 0;

if not CheckSrc then

exit;

for i := 0 to 3 do

begin

case i of

0:

begin

fr := VC\_Frame;

src := vc\_src;

str[i] := 'Верт. центр. нагружение';

end;

1:

begin

fr := H\_Frame;

src := h\_src;

str[i] := 'Верт. с смещением по гор-ли';

end;

2:

begin

fr := Hout\_Frame;

src := hout\_src;

str[i] := 'Гор. с смещением наружу';

end;

3:

begin

fr := Hin\_Frame;

src := hin\_src;

str[i] := 'Гор. с смещением внутрь';

end;

end;

graph[i] := ShowTimeGraphs(page, VC\_Frame, src, str[i]);

page := graph[i].hpage;

p := fr.GetInterval;

if i = 0 then

IWPGraphs(WP.GraphAPI).ShowCursor(graph[0].hpage, 2);

IWPGraphs(WP.GraphAPI).SetXCursorPos(graph[i].hgraph, p.x, false);

IWPGraphs(WP.GraphAPI).SetXCursorPos(graph[i].hgraph, p.y, true);

end;

IWPGraphs(WP.GraphAPI).SetPageDim(graph[0].hpage, PAGE\_DM\_TABLE, 2, 2);

for i := 0 to 3 do

begin

IWPGraphs(WP.GraphAPI).AddComment(graph[i].hgraph, str[i], 15, 10, 25, 5);

end;

WP.refresh;

end;

procedure formatgist(hline: integer);

begin

IWPGraphs(WP.GraphAPI).SetLineOpt(hline, LNOPT\_HIST, LNOPT\_HIST, 0, 0);

IWPGraphs(WP.GraphAPI).SetLineOpt(hline, LNOPT\_LINE2BASE, LNOPT\_LINE2BASE, 0, 0);

//setGistXScale(hline);

end;

procedure TRZDFrm.OnAddLine(Sender: TObject);

var

s: iwpSignal;

str: string;

V: double;

begin

if pos('cloud', m\_wpMng.m\_str) > 0 then

begin

IWPGraphs(WP.GraphAPI).SetLineOpt(m\_wpMng.m\_hline, LNOPT\_ONLYPOINTS,

LNOPT\_ONLYPOINTS, 0, $00D2D5);

exit;

end;

if m\_wpMng.m\_hline <> -1 then

begin

s := IWPGraphs(WP.GraphAPI).GetSignal(m\_wpMng.m\_hline) as iwpSignal;

if s = nil then

exit;

V := 0;

str := s.GetProperty('Gist');

if str <> '' then

V := strtofloatext(str);

if V = 1 then

begin

formatgist(m\_wpMng.m\_hline);

end;

end;

end;

procedure TRZDFrm.OnDestroySignal(Sender: TObject);

begin

end;

procedure TRZDFrm.OnDestroySrc(Sender: TObject);

begin

if Sender = vc\_src then

begin

vc\_src := nil;

end;

if Sender = h\_src then

begin

h\_src := nil;

end;

if Sender = hout\_src then

begin

hout\_src := nil;

end;

if Sender = hin\_src then

begin

hin\_src := nil;

end;

if Sender = m\_src then

begin

m\_src := nil;

end;

CheckSrc;

end;

function TRZDFrm.maxFS(s1, s2, s3, s4, vt, gt, mera: string): double;

var

ifile: tinifile;

f, fmax: double;

begin

ifile := tinifile.create(mera);

f := readFloatFromIni(ifile, s1, 'Freq');

fmax := f;

f := readFloatFromIni(ifile, s2, 'Freq');

if f > fmax then

fmax := f;

f := readFloatFromIni(ifile, s3, 'Freq');

if f > fmax then

fmax := f;

f := readFloatFromIni(ifile, s4, 'Freq');

if f > fmax then

fmax := f;

f := readFloatFromIni(ifile, vt, 'Freq');

if f > fmax then

fmax := f;

f := readFloatFromIni(ifile, gt, 'Freq');

if f > fmax then

fmax := f;

ifile.Destroy;

result := fmax;

end;

procedure TRZDFrm.NullBtnClick(Sender: TObject);

var

i: integer;

G: tgraphstruct;

p: point2d;

fr: TRZDTareFrame;

begin

for i := 0 to 3 do

begin

G := graph[i];

p := GetGraphCursorX(G.hpage, G.hgraph);

case i of

0:

fr := VC\_Frame;

1:

fr := H\_Frame;

2:

fr := Hout\_Frame;

3:

fr := Hin\_Frame;

end;

fr.NullFE1.FloatNum := p.x;

fr.NullFE2.FloatNum := p.y;

end;

end;

procedure TRZDFrm.UpdateIPortion;

var

fs: double;

frame: TRZDTareFrame;

begin

case TareFramePageControl.ActivePageIndex of

0:

frame := VC\_Frame;

1:

frame := H\_Frame;

2:

frame := Hout\_Frame;

3:

frame := Hin\_Frame;

end;

fs := maxFS(frame.S1Cbox.Text, frame.S2Cbox.Text, frame.S3Cbox.Text,

frame.S4Cbox.Text, frame.VFCbox.Text, frame.HFCbox.Text, frame.Path.Text);

if fs <> 0 then

TrendPortionIE.IntNum := round(TrendPortionFE.FloatNum \* fs);

end;

procedure TRZDFrm.UpdatePathBtnClick(Sender: TObject);

var

str, fld: string;

i: integer;

begin

if DirectoryExists(BaseFolderEdit.Text) then

begin

str := BaseFolderEdit.Text;

i := length(BaseFolderEdit.Text);

if BaseFolderEdit.Text[i] <> '\' then

str := BaseFolderEdit.Text + '\'

else

str := BaseFolderEdit.Text;

fld := str + 'Calibr';

if not DirectoryExists(fld) then

begin

ForceDirectories(fld);

end;

fld := str + 'Tests';

if not DirectoryExists(fld) then

begin

ForceDirectories(fld);

end;

fld := str + 'RZDmatrix';

if not DirectoryExists(fld) then

begin

ForceDirectories(fld);

end;

end;

BaseFolderEditChange(nil);

f\_needUpdateDB := true;

// UpdateDB(f\_updFolder);

UpdateDB;

end;

procedure TRZDFrm.UpdateFPortion;

var

fs: double;

frame: TRZDTareFrame;

begin

case TareFramePageControl.ActivePageIndex of

0:

frame := VC\_Frame;

1:

frame := H\_Frame;

2:

frame := Hout\_Frame;

3:

frame := Hin\_Frame;

end;

fs := maxFS(frame.S1Cbox.Text, frame.S2Cbox.Text, frame.S3Cbox.Text,

frame.S4Cbox.Text, frame.VFCbox.Text, frame.HFCbox.Text, frame.Path.Text);

if fs <> 0 then

TrendPortionFE.FloatNum := TrendPortionIE.IntNum / fs;

end;

procedure TRZDFrm.ExpMatrixBtnClick(Sender: TObject);

var

str, dir: string;

i: integer;

m: TRZDMatrix;

begin

str := RelativePathToAbsolute(BaseFolderEdit.Text, ExportMatrixPath.Text);

dir := extractfiledir(str);

str := dir + '\' + ExtractFileName(str);

if not DirectoryExists(dir) then

ForceDirectories(dir);

if DelMatrixOnExpCB.Checked then

begin

if fileexists(str) then

begin

DeleteFile(str);

end;

end;

for i := 0 to MatrixList.count - 1 do

BEGIN

m := MatrixList.Items[i];

ExportMatrix(m, str);

END;

end;

procedure TRZDFrm.ImpMatrixBtnClick(Sender: TObject);

var

str, dir, pref, V: string;

begin

str := RelativePathToAbsolute(BaseFolderEdit.Text, ExportMatrixPath.Text);

dir := extractfiledir(str);

str := dir + '\' + ExtractFileName(str);

ImpMatrix(str);

end;

procedure TRZDFrm.ImpMatrix(str: string);

var

dir, pref, V: string;

ini: tinifile;

sections: tstringlist;

i, j, n, k: integer;

m: TRZDMatrix;

begin

if not fileexists(str) then

exit;

// удаляем прошлые матрицы

ClearMList;

sections := tstringlist.create;

ini := tinifile.create(str);

ini.ReadSections(sections);

for i := 0 to sections.count - 1 do

begin

m := TRZDMatrix.create;

dir := sections.Strings[i];

for j := 0 to 3 do

begin

case j of

0:

pref := 'vc';

1:

pref := 'h';

2:

pref := 'hout';

3:

pref := 'hin';

end;

m.sensors[j].mf := ini.readString(dir, 'File\_' + pref, '');

m.sensors[j].vt := ini.readString(dir, 'Vt\_' + pref, '');

m.sensors[j].gt := ini.readString(dir, 'Gt\_' + pref, '');

m.sensors[j].s1 := ini.readString(dir, 'S1\_' + pref, '');

m.sensors[j].s2 := ini.readString(dir, 'S2\_' + pref, '');

m.sensors[j].s3 := ini.readString(dir, 'S3\_' + pref, '');

m.sensors[j].s4 := ini.readString(dir, 'S4\_' + pref, '');

m.sensors[j].t1t2.x := IniReadFloatEx(ini,dir, 't1\_' + pref, 0);

m.sensors[j].t1t2.y := IniReadFloatEx(ini,dir, 't2\_' + pref, 0);

end;

m.Date := ini.ReadString(dir, 'Date', '01.01.0001');

m.ls1 := ini.readString(dir, 'S1', '');

m.ls2 := ini.readString(dir, 'S2', '');

m.ls3 := ini.readString(dir, 'S3', '');

m.ls4 := ini.readString(dir, 'S4', '');

m.region := ini.readString(dir, 'Reg', m.region);

m.cut := ini.ReadInteger(dir, 'Segment', m.cut);

m.poly := ini.ReadInteger(dir, 'Poly', 1);

m.useSNames := ini.ReadBool(dir, 'UseSNames', m.useSNames);

m.m\_active := ini.ReadBool(dir, 'Active', true);

for j := 0 to 2 do

begin

str := ini.readString(dir, 'Row' + inttostr(j + 1), '');

V := GetSubString(str, ';', 1, k);

m.m[0, j] := strtofloatext(V);

V := GetSubString(str, ';', k + 1, k);

m.m[1, j] := strtofloatext(V);

V := GetSubString(str, ';', k + 1, k);

m.m[2, j] := strtofloatext(V);

V := GetSubString(str, ';', k + 1, k);

m.m[3, j] := strtofloatext(V);

end;

MatrixList.Add(m);

end;

sections.Destroy;

ShowMatrixList;

end;

procedure TRZDFrm.ExportMatrix(m: TRZDMatrix; ifile: string);

var

ini: tinifile;

fname, str, pref: string;

i: integer;

begin

ini := tinifile.create(ifile);

fname := m.GetIniName;

for i := 0 to 3 do

begin

case i of

0:

pref := 'vc';

1:

pref := 'h';

2:

pref := 'hout';

3:

pref := 'hin';

end;

ini.writeString(fname, 'File\_' + pref, m.sensors[i].mf);

ini.writeString(fname, 'Vt\_' + pref, m.sensors[i].vt);

ini.writeString(fname, 'Gt\_' + pref, m.sensors[i].gt);

ini.writeString(fname, 'S1\_' + pref, m.sensors[i].s1);

ini.writeString(fname, 'S2\_' + pref, m.sensors[i].s2);

ini.writeString(fname, 'S3\_' + pref, m.sensors[i].s3);

ini.writeString(fname, 'S4\_' + pref, m.sensors[i].s4);

ini.writeFloat(fname, 't1\_' + pref, m.sensors[i].t1t2.x);

ini.writeFloat(fname, 't2\_' + pref, m.sensors[i].t1t2.y);

end;

// Сохраняем только предустановленные имена

ini.writeString(fname, 'Date', m.date);

ini.writeString(fname, 'S1', m.ls1);

ini.writeString(fname, 'S2', m.ls2);

ini.writeString(fname, 'S3', m.ls3);

ini.writeString(fname, 'S4', m.ls4);

ini.writeString(fname, 'Reg', m.region);

ini.writeInteger(fname, 'Segment', m.cut);

ini.writeInteger(fname, 'Poly', m.poly);

ini.writeBool(fname, 'UseSNames', m.useSNames);

ini.writeBool(fname, 'Active', m.m\_active);

for i := 0 to 2 do

begin

str := m.getRow(i);

ini.writeString(fname, 'Row' + inttostr(i + 1), str);

end;

ini.Destroy;

end;

procedure TRZDFrm.ExportMatrixBtnClick(Sender: TObject);

begin

if SaveDialog1.Execute(0) then

begin

end;

end;

procedure TRZDFrm.ClearMList;

var

i: integer;

m: TRZDMatrix;

begin

for i := 0 to MatrixList.count - 1 do

begin

m := TRZDMatrix(MatrixList.Items[i]);

m.Destroy;

end;

MatrixList.clear;

end;

procedure TRZDFrm.FillMatrixSensorsCB;

var

i: integer;

str: string;

sections: tstringlist;

ifile: tinifile;

begin

ifile := tinifile.create(TestPath.Text);

sections := tstringlist.create;

ifile.ReadSections(sections);

Mat\_S1CB.clear;

Mat\_S2CB.clear;

Mat\_S3CB.clear;

Mat\_S4CB.clear;

for i := 0 to sections.count - 1 do

begin

str := sections.Strings[i];

if lowercase(str) <> 'mera' then

begin

Mat\_S1CB.AddItem(str, nil);

Mat\_S2CB.AddItem(str, nil);

Mat\_S3CB.AddItem(str, nil);

Mat\_S4CB.AddItem(str, nil);

end;

end;

ifile.Destroy;

sections.Destroy;

end;

procedure TRZDFrm.ShowMatrixList;

var

i: integer;

m: TRZDMatrix;

li: TListItem;

begin

MatrixLV.clear;

for i := 0 to MatrixList.count - 1 do

begin

m := TRZDMatrix(MatrixList.Items[i]);

m.UpdateSNames(gettestpath);

li := MatrixLV.Items.Add;

li.data := m;

li.Checked := m.m\_active;

MatrixLV.SetSubItemByColumnName('Матрица', m.Infostr, li);

MatrixLV.SetSubItemByColumnName('Рег.', m.region, li);

MatrixLV.SetSubItemByColumnName('Сеч.', inttostr(m.cut), li);

MatrixLV.SetSubItemByColumnName('Poly', inttostr(m.poly), li);

if m.cut = CutSE.Value then

begin

// отображает на форме матрицу

if m.region = RegionCB.Text then

ShowG(m.m);

end;

MatrixLV.SetSubItemByColumnName('S1', m.s1, li);

MatrixLV.SetSubItemByColumnName('S2', m.s2, li);

MatrixLV.SetSubItemByColumnName('S3', m.s3, li);

MatrixLV.SetSubItemByColumnName('S4', m.s4, li);

end;

LVChange(MatrixLV);

end;

function TRZDMatrix.ApplyMatrix(src: csrc): boolean;

var

l\_s1, l\_s2, l\_s3, l\_s4: iwpSignal;

b1, b2, b3, b4: boolean;

begin

result := false;

if useSNames then

begin

if (s1 = '') or (s2 = '') or (s1 = '') or (s2 = '') then

begin

RZDFrm.JournalLB.AddItem(

'Не определен один из сигналов, попытка поиска по префиксам...', nil);

exit;

end;

end

else

begin

UpdateSNames(src);

end;

l\_s1 := RZDFrm.GetSignal(src, s1, false, b1);

l\_s2 := RZDFrm.GetSignal(src, s2, false, b2);

l\_s3 := RZDFrm.GetSignal(src, s3, false, b3);

l\_s4 := RZDFrm.GetSignal(src, s4, false, b4);

if b1 and b2 and b3 and b4 then

begin

result := true;

RZDFrm.ApplyMatrix(l\_s1, l\_s2, l\_s3, l\_s4, m, src.name,

inttostr(cut) + '\_' + region,self);

end

else

begin

RZDFrm.JournalLB.AddItem('Не найдены сигналы', nil);

end;

end;

procedure TRZDMatrix.UpdateSNames(mf: string);

var

list: tstringlist;

begin

list := RZDFrm.GetSensorsNameWithPref(mf, cut);

if not useSNames then

s1 := list.Strings[0];

if not useSNames then

s2 := list.Strings[1];

if not useSNames then

s3 := list.Strings[2];

if not useSNames then

s4 := list.Strings[3];

list.Destroy;

end;

procedure TRZDMatrix.UpdateSNames(src: csrc);

var

list: tstringlist;

begin

list := RZDFrm.GetSensorsNameWithPref(src, cut);

if s1 = '' then

begin

if list.count <> 0 then

s1 := list.Strings[0];

end;

if s2 = '' then

begin

if list.count > 0 then

s2 := list.Strings[1];

end;

if s3 = '' then

begin

if list.count > 1 then

s3 := list.Strings[2];

end;

if s4 = '' then

begin

if list.count > 2 then

s4 := list.Strings[3];

end;

list.Destroy;

end;

function TRZDMatrix.getRow(i: integer): string;

begin

result := floattostr(m[0, i]) + ';' + floattostr(m[1, i]) + ';' + floattostr

(m[2, i]) + ';' + floattostr(m[3, i]);

end;

function TRZDMatrix.GetResPostfix: string;

begin

result := '\_' + inttostr(cut) + '\_' + region;

end;

function TRZDMatrix.Infostr: string;

begin

result := formatstr(m[0, 0], c\_digits) + ';' + formatstr(m[1, 0], c\_digits)

+ ';' + formatstr(m[2, 0], c\_digits) + ';' + formatstr(m[3, 0], c\_digits);

end;

function TRZDMatrix.GetIniName: string;

begin

result := 'REG=' + region + ';' + 'Segment=' + inttostr(cut);

end;

constructor TRZDMatrix.create;

begin

useSNames := false;

m\_active := true;

setlength(m, 4, 3);

end;

function TRZDMatrix.gets1: string;

begin

if useSNames then

result := ls1

else

result := fs1;

end;

function TRZDMatrix.gets2: string;

begin

if useSNames then

result := ls2

else

result := fs2;

end;

function TRZDMatrix.gets3: string;

begin

if useSNames then

result := ls3

else

result := fs3;

end;

function TRZDMatrix.gets4: string;

begin

if useSNames then

result := ls4

else

result := fs4;

end;

function TRZDMatrix.gets(index: integer): string;

begin

case index of

0:

result := s1;

1:

result := s2;

2:

result := s3;

3:

result := s4;

end;

end;

procedure TRZDMatrix.sets(index: integer; str: string);

begin

case index of

0:

ls1 := str;

1:

ls2 := str;

2:

ls3 := str;

3:

ls4 := str;

end;

end;

procedure TRZDFrm.UpdateDB;

var

capt: string;

begin

EnterCriticalSection(f\_updDBCS);

if f\_needUpdateDB then

begin

f\_needUpdateDB := false;

capt := RZDFrm.Caption;

RZDFrm.Caption := 'Обновление базы данных';

m\_DB.m\_BaseFolder.Path := BaseFolderEdit.Text;

// отображение в дереве

doUpdateBase(nil);

RZDFrm.Caption := capt;

end;

LeaveCriticalSection(f\_updDBCS);

end;

procedure TRZDFrm.UpdateDB(str: string);

var

fld, capt: string;

begin

EnterCriticalSection(f\_updDBCS);

if f\_needUpdateDB then

begin

f\_needUpdateDB := false;

// inc(ThreadCount);

capt := RZDFrm.Caption;

RZDFrm.Caption := 'Обновление базы данных';

fld := GetPathLevel(m\_DB.m\_BaseFolder.absolutepath, str, 1);

// log.addInfoMes('UpdateDB\_1');

m\_DB.UpdateDB(fld);

// отображение в дереве

doUpdateBase(nil);

RZDFrm.Caption := capt;

end;

LeaveCriticalSection(f\_updDBCS);

end;

procedure TRZDFrm.createDB;

begin

// log := cLogFile.create('f:\Oburec\delphi\2011\wp\RZD\signals\calibr\log.txt',';');

InitializeCriticalSection(f\_updDBCS);

// DeleteCriticalSection(f\_updDBCS);

if m\_DB = nil then

begin

m\_DB := cRZDbase.create;

m\_DB.m\_wpMng := m\_wpMng;

m\_DB.m\_syncHandle := Handle;

m\_DB.m\_BaseFolder.name := 'BaseFolder';

m\_DB.images\_16 := ImageList\_16;

m\_DB.images\_32 := ImageList\_32;

m\_DB.m\_MeraPref := '.mera';

m\_DB.m\_VCPref := m\_VCPref;

m\_DB.m\_Hpref := m\_H\_Pref;

m\_DB.m\_Hinpref := m\_Hin\_Pref;

m\_DB.m\_HOutpref := m\_Hout\_Pref;

m\_DB.regionPref := m\_regionPref;

m\_DB.m\_SegPref := m\_sectionPref;

crzdbasefolder(m\_DB.m\_BaseFolder).SetMatrixPref('.rzd');

crzdbasefolder(m\_DB.m\_BaseFolder).SetTestsPref;

crzdbasefolder(m\_DB.m\_BaseFolder).SetMainPath

(extractfiledir(ImportMatrixPath.Text), 'calibr', 'tests');

crzdbasefolder(m\_DB.m\_BaseFolder).m\_MatrixFolder.Caption := 'Matrix';

m\_DB.InitBaseFolder(BaseFolderEdit.Text);

// m\_db.Events.AddEvent('RZD\_UpdateBase', E\_OnUpdateFolder, doUpdateBaseMessage);

doUpdateBase(nil);

end;

f\_needUpdateDB := true;

end;

procedure TRZDFrm.UpdateHandler(var Message: TMessage);

var

str: widestring;

foldername: string;

L: integer;

obj: cDBObject;

baseobj: cDBFolder;

begin

case message.Msg of

wm\_UpdateFolder:

begin

f\_updFolder := m\_DB.m\_FileNotStr;

Timer1.Enabled := false;

Timer1.Enabled := true;

end;

end;

end;

procedure TRZDFrm.doUpdateBaseMessage(Sender: TObject);

begin

PostMessage(Handle, wm\_UpdateFolder, 0, 0);

end;

procedure TRZDFrm.doUpdateBase(Sender: TObject);

begin

// if GetCurrentThreadId=MainThreadID then

// begin

showInVTreeView(VTree1, m\_DB.m\_BaseFolder, ImageList\_16);

// end;

end;

procedure TRZDFrm.copyFile(str: string; p: crzdpars);

var

base, fname: string;

begin

if p.pars.sType > 0 then

begin

base := GetregionPath + '\' + m\_regionPref + inttostr(p.pars.reg)

+ '\' + m\_sectionPref + inttostr(p.pars.seg) + '\';

fname := ExtractFileName(str);

end;

p.Destroy;

end;

end.