```
unit frmMain;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes,
Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, UITypes, Vcl.ExtCtrls,
DBClient,
  CodeSiteLogging,LCRBenefits,
  LCRModel, LCRConfig, LCRGlobals, LCRMetricCollector, LCRCostVars, CostingSteps,
LCRPreCompile,
  LCRCompiledCost, LCRResultsFile, System.IOUtils;

type
  TForm1 = class(TForm)
    btnRunModel: TButton;
    btnStopModel: TButton;
    OpenDialog1: TOpenDialog;
    Timer1: TTimer;
    Label1: TLabel;
    Label2: TLabel;
    edtRunName: TEdit;
    Label3: TLabel;
    mmDescription: TMemo;
    Label4: TLabel;
    edtOptionLogicWorkbook: TEdit;
    btnOpenOptionLogicWrkbk: TButton;
    Label5: TLabel;
    edtOptionDatabase: TEdit;
    btnOpenOptionDatabase: TButton;
    Label6: TLabel;
    Panel1: TPanel;
    rbCWS: TRadioButton;
    rbNTNCWS: TRadioButton;
    Label7: TLabel;
    Panel2: TPanel;
    rb3Pct: TRadioButton;
    rb7Pct: TRadioButton;
    cmbOptions: TComboBox;
    Panel4: TPanel;
    rbFullRun: TRadioButton;
    rbBaselineOnly: TRadioButton;
    rbOptionOnly: TRadioButton;
    CheckBox1: TCheckBox;
    Label8: TLabel;
    Label9: TLabel;
    edtBaselineSDWISSample: TEdit;
    btnOpenBaselineSample: TButton;
    edtOptionSDWISSample: TEdit;
```

```
btnOpenOptionSample: TButton;
chkLeadBins: TCheckBox;
cbDebug: TCheckBox;
cbCCTToFull: TCheckBox;
Button1: TButton;
Label10: TLabel;
edtSmallProxyPop: TEdit;
Label13: TLabel;
cmbSchoolOption: TComboBox;
rbBoth: TRadioButton;
chkSmallSysFlex: TCheckBox;
cbNoBLAvg: TCheckBox;
cmbCCTCostEq: TComboBox;
Label14: TLabel;
edIQVS: TEdit;
Label15: TLabel;
Label16: TLabel;
edIQDR: TEdit;
edCVDR: TEdit;
Label17: TLabel;
Label18: TLabel;
edChildBL: TEdit;
Label19: TLabel;
CheckBox2: TCheckBox;
Label20: TLabel;
edtVolProg: TEdit;
lbStat: TLabel;
CheckBox3: TCheckBox;
Button2: TButton;
Label21: TLabel;
CheckBox4: TCheckBox;
Label22: TLabel;
Edit1: TEdit;
cbBenAll: TCheckBox;
Label23: TLabel;
cbBenByYear: TCheckBox;
Label24: TLabel;
edtBaseLogicWorkbook: TEdit;
btnOpenBaselineLogicWrkbk: TButton;
Label25: TLabel;
edtBaselineDatabase: TEdit;
btnOpenBaselineDatabase: TButton;
Label26: TLabel;
cmbALE: TComboBox;
Label11: TLabel;
Label12: TLabel;
cmbLSLOption: TComboBox;
Label27: TLabel;
cmbLSLRepPct: TComboBox;
Label28: TLabel;
```

```
  cmbTempPOU: TComboBox;
  Label29: TLabel;
  cmbWQPCutoff: TComboBox;
  Label30: TLabel;
  cmbLSLRCap: TComboBox;
  Button3: TButton;
  cbFilters: TCheckBox;
  edtLCRBaselineDB: TEdit;
  btnOpenLCRBaselineDB: TButton;
  cbUseLCRBaseline: TCheckBox;
  Label31: TLabel;
  Label32: TLabel;
  cmbBaseline: TComboBox;
  Label33: TLabel;
  Panel3: TPanel;
  rbRunTypeLow: TRadioButton;
  rbRunTypeHigh: TRadioButton;
  cbDelaware: TCheckBox;
  Label34: TLabel;

  procedure btnRunModelClick(Sender: TObject);
  procedure FormCreate(Sender: TObject);
  procedure FormDestroy(Sender: TObject);
  procedure FormActivate(Sender: TObject);
  procedure rbMeanRunClick(Sender: TObject);
  procedure rbVariableRunClick(Sender: TObject);

  procedure btnOpenBaselineLogicWrkbkClick(Sender: TObject);
  procedure btnOpenOptionLogicWrkbkClick(Sender: TObject);
  procedure btnOpenBaselineDatabaseClick(Sender: TObject);
  procedure btnOpenOptionDatabaseClick(Sender: TObject);

  procedure btnOpenBaselineSampleClick(Sender: TObject);
  procedure btnOpenOptionSampleClick(Sender: TObject);
  procedure Button1Click(Sender: TObject);
  procedure CheckBox2Click(Sender: TObject);
  procedure btnStopModelClick(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure rbOptionOnlyClick(Sender: TObject);
  procedure rbFullRunClick(Sender: TObject);
  procedure rbBaselineOnlyClick(Sender: TObject);
  procedure Button3Click(Sender: TObject);
  procedure btnOpenLCRBaselineDBClick(Sender: TObject);
private
  { Private declarations }
  RunName: string;
  OptionName: string;
  InitShow, InitActive, Stop: boolean;
  HiddenForm: boolean;
  CCTCostEquationLevel: string;
```

```
    procedure SaveLCRRunData(Filename: string);
    procedure SetParameters(Filename: string);
    procedure Status(Msg : string; var aStop : boolean);
    procedure SetUI;
  public
    { Public declarations }
    CurConfig: TLCRConfig;
    ModelRunning: boolean;
    LCRModel: TLCRModel;
    CurRunOutput : string;

    procedure ModelIsDone(Sender: TObject);
  end;

var
  Form1: TForm1;

implementation

uses DB, inifiles, SafewaterUncertBucket, frmTestParser;

{$R *.dfm}

procedure TForm1.btnOpenBaselineSampleClick(Sender: TObject);
begin
  if TButton(Sender).Tag = 1 then
    Opendialog1.Title := 'Select Baseline SDWIS Sample File';

  OpenDialog1.InitialDir := DataPath;
  OpenDialog1.Filter := 'CSV File|*.csv';
  OpenDialog1.FileName := '';
  if OpenDialog1.Execute then
    edtBaselineSDWISSample.Text := OpenDialog1.FileName;
end;

procedure TForm1.btnOpenLCRBaselineDBClick(Sender: TObject);
begin
  OpenDialog1.InitialDir := DataPath;
  OpenDialog1.Filter := 'Access database|*.accdb';
  OpenDialog1.FileName := '';
  if OpenDialog1.Execute then
    edtLCRBaselineDB.Text := OpenDialog1.FileName;
end;

procedure TForm1.btnOpenBaselineLogicWrkbkClick(Sender: TObject);
begin
  if TButton(Sender).Tag = 3 then
    Opendialog1.Title := 'Select Baseline Costing Workbook';
```

```
  OpenDialog1.InitialDir := DataPath;
  OpenDialog1.Filter := 'Excel Workbook|*.xlsx';
  OpenDialog1.FileName := '';
  if OpenDialog1.Execute then
    edtBaseLogicWorkbook.Text := OpenDialog1.FileName;
end;

procedure TForm1.btnOpenOptionLogicWrkbkClick(Sender: TObject);
begin
  if TButton(Sender).Tag = 5 then
    Opendialog1.Title := 'Select Option Costing Workbook';

  OpenDialog1.InitialDir := DataPath;
  OpenDialog1.Filter := 'Excel Workbook|*.xlsx';
  OpenDialog1.FileName := '';
  if OpenDialog1.Execute then
    edtOptionLogicWorkbook.Text := OpenDialog1.FileName;
end;

procedure TForm1.btnOpenOptionSampleClick(Sender: TObject);
begin
  if TButton(Sender).Tag = 2 then
    Opendialog1.Title := 'Select Option SDWIS Sample File';

  OpenDialog1.InitialDir := DataPath;
  OpenDialog1.Filter := 'CSV File|*.csv';
  OpenDialog1.FileName := '';
  if OpenDialog1.Execute then
    edtOptionSDWISSample.Text := OpenDialog1.FileName;
end;

procedure TForm1.btnOpenBaselineDatabaseClick(Sender: TObject);
begin
  if TButton(Sender).Tag = 4 then
    Opendialog1.Title := 'Select Baseline Variable Database';

  OpenDialog1.InitialDir := DataPath;
  OpenDialog1.Filter := 'Access database|*.accdb';
  OpenDialog1.FileName := '';
  if OpenDialog1.Execute then
    edtBaselineDatabase.Text := OpenDialog1.FileName;
end;

procedure TForm1.btnOpenOptionDatabaseClick(Sender: TObject);
begin
  if TButton(Sender).Tag = 6 then
    Opendialog1.Title := 'Select Option Variable Database';

  OpenDialog1.InitialDir := DataPath;
  OpenDialog1.Filter := 'Access database|*.accdb';
```

```
  OpenDialog1.FileName := '';
  if OpenDialog1.Execute then
    edtOptionDatabase.Text := OpenDialog1.FileName;
end;

procedure TForm1.btnRunModelClick(Sender: TObject);
var
  S,FN,FNP : string;
begin
  if cmbOptions.ItemIndex = -1 then
    exit;

  if MessageDlg('Run with current settings?',mtConfirmation,[mbYes,mbNo],0) <> mrYes
then exit;

  S:='';

  if edtRunName.Text <> 'Run Name' then
    OpenDialog1.FileName := edtRunName.Text;

  OpenDialog1.InitialDir := UserPath;
  if not OpenDialog1.Execute then exit;
  S:=OpenDialog1.FileName;
  FN:=ChangeFileExt(ExtractFileName(S),'');

  edtRunName.Text := FN;
  Application.ProcessMessages;
  OptionName := cmbOptions.Items[cmbOptions.ItemIndex];
  CCTCostEquationLevel := cmbCCTCostEq.Items[cmbCCTCostEq.ItemIndex];

  NoRandom:=Checkbox1.Checked;
  if NoRandom then RandSeed:=1;

  CurConfig.RunName:=FN;
  CurConfig.BaselineName := cmbBaseline.Items[cmbBaseline.ItemIndex];
  CurConfig.OptionName := OptionName;
  CurConfig.RunDescription := StringReplace(mmDescription.Lines.Text, #$D#$A, ' ',
[rfReplaceAll]);
  CurConfig.BasePWSDataFile := edtBaselineSDWISSample.Text;
  CurConfig.ScenPWSDataFile := edtOptionSDWISSample.Text;

  CurConfig.LCRBaselineDB := edtLCRBaselineDB.Text;

  CurConfig.BaseCostSteps := edtBaseLogicWorkbook.Text;
  CurConfig.BaseVarData := edtBaselineDatabase.Text;
  CurConfig.ScenCostSteps := edtOptionLogicWorkbook.Text;
  CurConfig.ScenVarData := edtOptionDatabase.Text;

  CurConfig.Debug:=cbDebug.Checked;
  CurConfig.CCTToFull:=cbCCTToFull.Checked;
```

```
CurConfig.CCTCostEquationLevel := CCTCostEquationLevel;
CurConfig.BenefitsAll := cbBenAll.Checked;

CurConfig.IQValueSens:=  strtoint(edIQVS.Text);
CurConfig.IQDRSens:=  strtoint(edIQDR.Text);
CurConfig.CVDDRSens:=strtoint(edCVDR.Text);
CurConfig.ChildBLSens:=strtoint(edChildBL.Text);
CurConfig.VolLeadProg := StrToInt(edtVolProg.Text);
CurConfig.BenByYear := cbBenByYear.Checked;

if rbCWS.Checked then
begin
  CurConfig.SystemType := sysCWS;
  CurConfig.RunSysType := 'CWS';
end
else
if rbNTNCWS.Checked then
begin
  CurConfig.SystemType := sysNTNC;
  CurConfig.RunSysType := 'NTNCWS';
end
else
if rbBoth.Checked then
begin
  CurConfig.SystemType := sysCWS;
  CurConfig.RunSysType := 'Both';
end;

if rb3Pct.Checked then
  CurConfig.DiscountRate := 0.03
else
if rb7Pct.Checked then
  CurConfig.DiscountRate := 0.07;

  CurConfig.VariabilityRun := true;
  CurConfig.MeanUncertLevel := false;
  CurConfig.NumberOfVLoops := 1;

CurConfig.RunBaselineOnly := rbBaselineOnly.Checked;
CurConfig.RunOptionOnly := rbOptionOnly.Checked;
CurConfig.RunDifference := rbFullRun.Checked;
CurConfig.RunNoBLAveraging := cbNoBLAvg.Checked;
CurConfig.OutputLeadBins := chkLeadBins.Checked;
CurConfig.SmallSystemFlexibility := True;

CurConfig.SmallProxyPop := StrToInt(edtSmallProxyPop.Text);
CurConfig.PWS90PctBp1 := 10;
CurConfig.PWS90PctBp2 := 15;

CurConfig.SchoolOption := cmbSchoolOption.Items[cmbSchoolOption.ItemIndex];
```

```
CurConfig.ALE := StrToInt(cmbALE.Items[cmbALE.ItemIndex]);
CurConfig.LSLOption := StrToInt(cmbLSLOption.Items[cmbLSLOption.ItemIndex]);
CurConfig.LSLRepPct := StrToFloat(cmbLSLRepPct.Items[cmbLSLRepPct.ItemIndex]);
CurConfig.TempPOUOption := StrToInt(cmbTempPOU.Items[cmbTempPOU.ItemIndex]);
CurConfig.UseCompiledCost := CheckBox4.Checked;
CurConfig.WQPCutoff := cmbWQPCutoff.Items[cmbWQPCutoff.ItemIndex];

CurConfig.LSLRCap := StrToFloat(cmbLSLRCap.Items[cmbLSLRCap.ItemIndex]);
if rbRunTypeLow.Checked then
  CurConfig.RunType := rtLow
else
if rbRunTypeHigh.Checked then
  CurConfig.RunType := rtHigh;

if cbFilters.Checked then
  CurConfig.LSLFilters := 1
else
  CurConfig.LSLFilters := 0;

if cbUseLCRBaseline.Checked then
  CurConfig.UseLCRBaseline := 1
else
  CurConfig.UseLCRBaseline := 0;

if cbDelaware.Checked then
  CurConfig.FastRun := 1
else
  CurConfig.FastRun := 0;

RunName:=CurConfig.RunName;
FNP:=ExtractFilePath(OpenDialog1.FileName);
FN:=ChangeFileExt(FN,'.swc');

_UseCompiled := CurConfig.UseCompiledCost;
CurConfig.Save(FNP+FN);
ModelRunning:=True;

LCRModel:=TLCRModel.Create(FNP+FN);
LCRModel.OnProgress := status;
LCRModel.MicroOutput := CheckBox3.Checked;
CurRunOutput:=ChangeFileExt(FNP+FN,'.swo');

LCRModel.OnModelDone:=ModelIsDone;
Screen.Cursor:=crHourGlass;

LCRModel.Initialize(nil,FNP);

Screen.Cursor:=crDefault;
btnRunModel.Enabled:=False;
btnStopModel.Enabled:=True;
```

```
  LCRModel.Run;
  showmessage('done');
end;

procedure TForm1.btnStopModelClick(Sender: TObject);
begin
  Stop:=True;
end;

procedure TForm1.Button1Click(Sender: TObject);
var B : TLCRBenefits;
begin
//  frmTestParse.Show;
  B:=TLCRBenefits.create(CurConfig, nil, nil, 'a');
  B.CalcBinMove(1,12,1,1000,False,btMandatory,True);
  B.Free;
end;

procedure TForm1.Button2Click(Sender: TObject);
var T : TLCRPreCompile;
    var BLCRFile, BLCRRFile, SFIle : string;
begin
  OpenDialog1.InitialDir := DataPath;
  OpenDialog1.Filter:='Excel (*.xlsx) |*.xlsx';
  OpenDialog1.Title := 'Select LCR Baseline Workbook';
  if OpenDialog1.Execute then begin
    BLCRFile := OpenDialog1.FileName;
    OpenDialog1.Title := 'Select LCRR Baseline Workbook';
    if OpenDialog1.Execute then begin
      BLCRRFile := OpenDialog1.FileName;
      OpenDialog1.Title := 'Select Option Workbook';
      if OpenDialog1.Execute then begin
        sFile := OpenDialog1.FileName;
        T:=TLCRPreCompile.create(edit1.text,sFile, bLCRFile, bLCRRFile);
        T.Go;
        T.Free;
        showmessage('done');
        Close;
      end;
    end;
  end;
  OpenDialog1.Title := '';
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  var C:=TLCRConfig.Create;
  var U := TUncertaintyStudy.Create(1);
  var O:=TMetricList.Create(C);
```

```
  var fResultsFile := TLCRResultsFile.Create(UserPath+'tt.swr',50, 1);
  O.ResultsFile := fResultsFile;
  var B := TBenefitsCollector.create(c,O,U);

  B.Free;
  C.Free;
  O.Free;
  U.Free;
  fResultsFile.Free;
end;

procedure TForm1.CheckBox2Click(Sender: TObject);
begin
  CodeSite.Enabled:=CheckBox2.Checked;
  CSL('Codesight enabled');
end;

procedure TForm1.FormActivate(Sender: TObject);
begin
  if InitActive then begin
    CodeSite.Enabled:=False;
    Label21.Caption := 'Compiled Costs LCR Base: '+_CworkBookLCRBaseline+'
'+_CWorkBookDate;
    Label34.Caption := 'Compiled Costs LCRR Base: '+_CworkBookLCRRBaseline+'
'+_CWorkBookDate;
    Label22.Caption := 'Compiled Costs Option: '+_CworkBookOption;

    if paramstr(1)<>'' then begin
      var FN:=ChangeFileExt(paramstr(1),'');
      var FNP:=ExtractFilePath(paramstr(1));
      FN:=ChangeFileExt(FN,'.swc');

      CurConfig.Load(paramstr(1));
      _UseCompiled := True;
      SetUI();
      Application.ProcessMessages;
      LCRModel:=TLCRModel.Create(FN);
      LCRModel.OnProgress := status;
      LCRModel.MicroOutput := CheckBox3.Checked;
      LCRModel.OnModelDone:=ModelIsDone;
      Screen.Cursor:=crHourGlass;
      CurRunOutput:=ChangeFileExt(FN,'.swo');
      LCRModel.Initialize(nil,FNP);


      Screen.Cursor:=crDefault;
      btnRunModel.Enabled:=False;
      btnStopModel.Enabled:=True;
      LCRModel.Run;
```

```
      var TabPath := ExcludeTrailingPathDelimiter(FNP) +'_tabs\';
      ForceDirectories(TabPath);
      var TabFile:=ChangeFileExt(FN,'.tab');
      try
        TFile.Copy(TabFile,TabPath+ExtractFileName(TabFile),true);
      except

      end;
      Application.Terminate;

    end else begin
      if MessageDlg('Do you want to load a saved configuration?  Hit Yes to load or
No to use the default settings.',mtConfirmation,[mbYes,mbNo],0)=mrYes then
      begin
        OpenDialog1.InitialDir := UserPath;
        OpenDialog1.Filter:='SW Config (*.swc) |*.swc';
        if OpenDialog1.Execute then
          CurConfig.Load(OpenDialog1.FileName);
        InitActive := false;
        BringToFront;
        SetUI();
      end;
    end;
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  InitShow := true;
  InitActive := true;

  AppPath := ExtractFilePath(Application.ExeName);
  DataPath := AppPath + 'data\';
  UserPath := AppPath + 'user\';
  HelpPath := AppPath + 'help\';
  LogPath := AppPath + 'logs\';
  ForceDirectories(LogPath);

  CurConfig := TLCRConfig.Create;

  ModelRunning := false;
  HiddenForm := false;
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  CurConfig.Free;
end;

procedure TForm1.ModelIsDone(Sender: TObject);
```

```
begin
  Timer1.Enabled:=False;
  lll.L('runevents','LCRModel.Destroy');
  LCRModel.Destroy;
  lll.L('runevents','LCRModel.Destroy Done.');
  ModelRunning:=False;

  if not HiddenForm then
  begin
    btnRunModel.Enabled:=True;
    btnStopModel.Enabled:=False;
  end;
  lll.L('runevents','saving output');
  SaveLCRRunData(CurRunOutput);
  lll.L('runevents','saving output - done');

  if HiddenForm then
    Application.Terminate;
end;

procedure TForm1.rbBaselineOnlyClick(Sender: TObject);
begin
    if rbBaselineOnly.Checked then
    begin
      cmbALE.Enabled := False;
      cmbLSLOption.Enabled := False;
      cmbLSLRepPct.Enabled := False;
      cmbTempPOU.Enabled := False;
    end
    else
    begin
      cmbALE.Enabled := True;
      cmbLSLOption.Enabled := True;
      cmbLSLRepPct.Enabled := True;
      cmbTempPOU.Enabled := True;
    end;
end;

procedure TForm1.rbFullRunClick(Sender: TObject);
begin
    if rbBaselineOnly.Checked then
    begin
      cmbALE.Enabled := False;
      cmbLSLOption.Enabled := False;
      cmbLSLRepPct.Enabled := False;
      cmbTempPOU.Enabled := False;
    end
    else
    begin
      cmbALE.Enabled := True;
```

```
      cmbLSLOption.Enabled := True;
      cmbLSLRepPct.Enabled := True;
      cmbTempPOU.Enabled := True;
    end;
end;

procedure TForm1.rbMeanRunClick(Sender: TObject);
begin
  //edtIterations.Enabled := false;
end;

procedure TForm1.rbOptionOnlyClick(Sender: TObject);
begin
  if rbBaselineOnly.Checked then
  begin
    cmbALE.Enabled := False;
    cmbLSLOption.Enabled := False;
    cmbLSLRepPct.Enabled := False;
    cmbTempPOU.Enabled := False;
  end
  else
  begin
    cmbALE.Enabled := True;
    cmbLSLOption.Enabled := True;
    cmbLSLRepPct.Enabled := True;
    cmbTempPOU.Enabled := True;
  end;
end;

procedure TForm1.rbVariableRunClick(Sender: TObject);
begin
  //edtIterations.Enabled := true;
end;

procedure TForm1.SaveLCRRunData(Filename: string);
var
  T: TCategoryOutputReader;
  cdsOutputData: TClientDataset;
begin
  cdsOutputData := TClientDataSet.Create(nil);
  cdsOutputData.Close;

  cdsOutputData.FieldDefs.Clear;
  with cdsOutputData do
  begin
    with FieldDefs.AddFieldDef do
    begin
      Name := 'Category';
      DataType := ftString;
      Size := 100;
```

```
end;
with FieldDefs.AddFieldDef do
begin
  Name := 'MetricClass';
  DataType := ftString;
  Size := 15;
end;
with FieldDefs.AddFieldDef do
begin
  Name := 'Metric';
  DataType := ftString;
  Size := 80;
end;
with FieldDefs.AddFieldDef do
begin
  Name := 'Option';
  DataType := ftString;
end;
with FieldDefs.AddFieldDef do
begin
  Name := 'DiscountRate';
  DataType := ftFloat;
end;
with FieldDefs.AddFieldDef do
begin
  Name := 'Statistic';
  DataType := ftString;
  Size := 10;
end;
with FieldDefs.AddFieldDef do
begin
  Name := 'P1';
  DataType := ftFloat;
end;
with FieldDefs.AddFieldDef do
begin
  Name := 'P5';
  DataType := ftFloat;
end;
with FieldDefs.AddFieldDef do
begin
  Name := 'Mean';
  DataType := ftFloat;
end;
with FieldDefs.AddFieldDef do
begin
  Name := 'P95';
  DataType := ftFloat;
end;
with FieldDefs.AddFieldDef do
```

```
    begin
      Name := 'P99';
      DataType := ftFloat;
    end;
    with FieldDefs.AddFieldDef do
    begin
      Name := 'sDR';
      DataType := ftString;
    end;

    with IndexDefs do
    begin
      Clear;
      AddIndexDef.Name := 'USEIDX';
      AddIndexDef.Fields := 'Category;Metric;Option;DiscountRate;Statistic';
      AddIndexDef.Options := [];
    end;

  end;

  cdsOutputData.CreateDataSet;
  cdsOutputData.LogChanges := false;

  T := TCategoryOutputReader.Create(ChangeFileExt(Filename, '.swomn'));

  T.DumpToSmallCSV(Filename);
  T.DumpToTabDelim(ChangeFileExt(Filename, '.tab'), true, CurConfig.NoICRCosts);

  T.Free;
  cdsOutputData.Free;
end;

procedure TForm1.SetParameters(Filename: string);
var
  iniFile: TIniFile;
  RunName, OptionName, CCTCostEquationLevel: string;
  BaselineSDWIS, OptionSDWIS, OptionCostingLogic, OptionVariables: string;
  Population, DiscountRate, RunType: string;
  FNP: string;
  BaselineCostingLogic, BaselineVariables: string;
begin
  iniFile := TIniFile.Create(AppPath + '\' + Filename);

  RunName := iniFile.ReadString('Params','RunName',RunName);
  OptionName := iniFile.ReadString('Params','OptionName',OptionName);
  BaselineSDWIS := iniFile.ReadString('Params','BaselineSDWIS',BaselineSDWIS);
  OptionSDWIS := iniFile.ReadString('Params','OptionSDWIS',OptionSDWIS);

  BaselineCostingLogic :=
iniFile.ReadString('Params','BaselineCostingLogic',BaselineCostingLogic);
```

```
  BaselineVariables :=
iniFile.ReadString('Params','BaselineVariables',BaselineVariables);

  OptionCostingLogic :=
iniFile.ReadString('Params','OptionCostingLogic',OptionCostingLogic);
  OptionVariables := iniFile.ReadString('Params','OptionVariables',OptionVariables);
  Population := iniFile.ReadString('Params','Population',Population);
  DiscountRate := iniFile.ReadString('Params','DiscountRate',DiscountRate);
  RunType := iniFile.ReadString('Params','RunType',RunType);
  CCTCostEquationLevel :=
iniFile.ReadString('Params','CCTCostEquationLevel',CCTCostEquationLevel);

  iniFile.Free;

  CurConfig.RunName:=RunName;
  CurConfig.OptionName := OptionName;
  //CurConfig.RunDescription := StringReplace(mmDescription.Lines.Text, #$D#$A, ' ',
[rfReplaceAll]);
  CurConfig.BasePWSDataFile := BaselineSDWIS;
  CurConfig.ScenPWSDataFile := OptionSDWIS;

  CurConfig.BaseCostSteps := BaselineCostingLogic;
  CurConfig.BaseVarData := BaselineVariables;

  CurConfig.ScenCostSteps := OptionCostingLogic;
  CurConfig.ScenVarData := OptionVariables;
  CurConfig.CCTCostEquationLevel := CCTCostEquationLevel;

  if Population = 'CWS' then
    CurConfig.SystemType := sysCWS
  else
  if Population = 'NTNCWS' then
    CurConfig.SystemType := sysNTNC;

  if DiscountRate = '3%' then
    CurConfig.DiscountRate := 0.03
  else
  if DiscountRate = '7%' then
    CurConfig.DiscountRate := 0.07;

  CurConfig.RunBaselineOnly := false;
  CurConfig.RunOptionOnly := false;
  CurConfig.RunDifference := false;

  if RunType = 'Baseline' then
    CurConfig.RunBaselineOnly := true
  else if RunType = 'Option' then
    CurConfig.RunOptionOnly := true
  else if RunType = 'Incremental' then
    CurConfig.RunDifference := true;
```

```
  FNP:=UserPath;
  RunName:=UserPath + ChangeFileExt(RunName,'.swc');

  CurConfig.Save(RunName);
  ModelRunning:=True;

  LCRModel:=TLCRModel.Create(RunName);
  LCRModel.OnProgress := status;
  LCRModel.MicroOutput := CheckBox3.Checked;
  CurRunOutput:=ChangeFileExt(RunName,'.swo');

  LCRModel.OnModelDone:=ModelIsDone;

  LCRModel.Initialize(nil,FNP);
  Stop:=false;

  LCRModel.Run;
end;

procedure TForm1.SetUI;
begin
  edtRunName.Text := CurConfig.RunName;
  mmDescription.Lines.Text := CurConfig.RunDescription;
  edtBaselineSDWISSample.Text := CurConfig.BasePWSDataFile;
  edtOptionSDWISSample.Text := CurConfig.ScenPWSDataFile;

  edtLCRBaselineDB.Text := CurConfig.LCRBaselineDB;
  edtBaseLogicWorkbook.Text := CurConfig.BaseCostSteps;
  edtBaselineDatabase.Text := CurConfig.BaseVarData;
  edtOptionLogicWorkbook.Text := CurConfig.ScenCostSteps;
  edtOptionDatabase.Text := CurConfig.ScenVarData;

  edIQVS.Text:=inttostr(CurConfig.IQValueSens);
  edIQDR.Text:=inttostr(CurConfig.IQDRSens);
  edCVDR.Text:=inttostr(CurConfig.CVDDRSens);
  edChildBL.Text:=inttostr(CurConfig.ChildBLSens);
  edtVolProg.Text := IntToStr(CurConfig.VolLeadProg);
  cbBenAll.Checked := CurConfig.BenefitsAll;
  cbBenByYear.Checked := CurConfig.BenByYear;

  if CurConfig.RunSysType = 'CWS' then
    rbCWS.Checked := True
  else
  if CurConfig.RunSysType = 'NTNCWS' then
    rbNTNCWS.Checked := True
  else
  if CurConfig.RunSysType = 'Both' then
    rbBoth.Checked := True;
```

```
  if Round(CurConfig.DiscountRate*100)/100 = 0.03 then
    rb3Pct.Checked := true
  else
  if Round(CurConfig.DiscountRate*100)/100 = 0.07 then
    rb7Pct.Checked := true;

  rbBaselineOnly.Checked := CurConfig.RunBaselineOnly;
  rbOptionOnly.Checked := CurConfig.RunOptionOnly;
  rbFullRun.Checked := CurConfig.RunDifference;
  chkLeadBins.Checked := CurConfig.OutputLeadBins;
  chkSmallSysFlex.Checked := CurConfig.SmallSystemFlexibility;
  cbNoBLAvg.Checked:=CurConfig.RunNoBLAveraging;
  edtSmallProxyPop.Text := CurConfig.SmallProxyPop.ToString;

  if CurConfig.RunType = rtLow then
    rbRunTypeLow.Checked := true
  else
  if CurConfig.RunType = rtHigh then
    rbRunTypeHigh.Checked := true;

  if rbBaselineOnly.Checked then
  begin
    cmbALE.Enabled := False;
    cmbLSLOption.Enabled := False;
    cmbLSLRepPct.Enabled := False;
    cmbTempPOU.Enabled := False;
  end
  else
  begin
    cmbALE.Enabled := True;
    cmbLSLOption.Enabled := True;
    cmbLSLRepPct.Enabled := True;
    cmbTempPOU.Enabled := True;
  end;

  cmbBaseline.ItemIndex := cmbBaseline.Items.IndexOf(CurConfig.BaselineName);
  cmbOptions.ItemIndex := cmbOptions.Items.IndexOf(CurConfig.OptionName);
  cmbCCTCostEq.ItemIndex :=
cmbCCTCostEq.Items.IndexOf(CurConfig.CCTCostEquationLevel);
  cmbSchoolOption.ItemIndex :=
cmbSchoolOption.Items.IndexOf(CurConfig.SchoolOption);
  cmbALE.ItemIndex := cmbALE.Items.IndexOf(IntToStr(CurConfig.ALE));
  cmbLSLOption.ItemIndex :=
cmbLSLOption.Items.IndexOf(IntToStr(CurConfig.LSLOption));
  cmbLSLRepPct.ItemIndex :=
cmbLSLRepPct.Items.IndexOf(FloatToStrF(CurConfig.LSLRepPct, ffFixed, 3, 2));
  cmbTempPOU.ItemIndex :=
cmbTempPOU.Items.IndexOf(IntToStr(CurConfig.TempPOUOption));
  CheckBox4.Checked := CurConfig.UseCompiledCost;
  cmbWQPCutoff.ItemIndex := cmbWQPCutoff.Items.IndexOf(CurConfig.WQPCutoff);
```

```
  cmbLSLRCap.ItemIndex := cmbLSLRCap.Items.IndexOf(FloatToStr(CurConfig.LSLRCap));
  if CurConfig.LSLFilters = 1 then
    cbFilters.Checked := true
  else
    cbFilters.Checked := false;

  if CurConfig.UseLCRBaseline = 1 then
    cbUseLCRBaseline.Checked := true
  else
    cbUseLCRBaseline.Checked := false;

  if CurConfig.FastRun = 1 then
    cbDelaware.Checked := true
  else
    cbDelaware.Checked := false;

end;

procedure TForm1.Status(Msg: string; var aStop: boolean);
begin
  aStop:=Stop;
  lbStat.Caption := Msg;
  Application.ProcessMessages;
end;

end.
```