```
unit fmrMainBatchMaker;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes,
Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Mask, Vcl.ExtCtrls,
  Vcl.ComCtrls, VCL.FileCtrl, Vcl.Grids, LCRConfig, Math, LCRGlobals,
System.IOUtils;

type
  TForm1 = class(TForm)
    PageControl1: TPageControl;
    Panel1: TPanel;
    TabSheet1: TTabSheet;
    TabSheet2: TTabSheet;
    TabSheet3: TTabSheet;
    edRootDir: TLabeledEdit;
    Button1: TButton;
    cbFastRun: TCheckBox;
    Button2: TButton;
    Button3: TButton;
    edBatchName: TLabeledEdit;
    OD: TOpenDialog;
    gColumns: TStringGrid;
    TabSheet4: TTabSheet;
    RawFile: TMemo;
    gParsed: TStringGrid;
    cbSimul: TCheckBox;
    memOut: TMemo;
    Label1: TLabel;
    Label2: TLabel;
    edMax: TLabeledEdit;
    edEXEName: TLabeledEdit;
    cbInc: TCheckBox;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure gColumnsExit(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
  private
    { Private declarations }
    BatchDir, DataDir: string;
    Config: TLCRConfig;
    SettingCount: integer;
    Found : array of boolean;
    function GetIndex(s: string) : integer;
    procedure AddSetting(s: string; defcol: integer);
```

```
    procedure RefreshColIdx;
    procedure Parse;
    function ProcessSetting(SettingName, Value: string): boolean;
    procedure Process;
    function iSameText(SettingName, Value: string): boolean;
    function GetColNameIndex(s: string): integer;
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

const Waiting =
'$process = Get-Process $safewaterlcrr$ -ErrorAction silentlycontinue'+#13#10+
'while ($process) {'+#13#10+
'  Start-Sleep -Seconds 5'+#13#10+
'  $process = Get-Process safewaterlcrr -ErrorAction silentlycontinue'+#13#10+
'}'+#13#10;


implementation

{$R *.dfm}

procedure TForm1.AddSetting(s: string; defcol: integer);
begin
  inc(SettingCount);
  gColumns.Cells[0,SettingCount] := s;
  gColumns.Cells[1,SettingCount] := defcol.ToString;
  gColumns.RowCount := SettingCount+1;
  setlength(Found,SettingCount);
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  var s:string := edRootDir.Text;
  if SelectDirectory(s,[],0) then begin
    edRootDir.Text := s + '\';
    BatchDir := edRootDir.Text+'batch\';
    DataDir := edRootDir.Text+'data\';
    ForceDirectories(BatchDir);
  end;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  if OD.Execute() then begin
    edBatchName.Text := ChangeFileExt(ExtractFileName(OD.FileName),'');
    RawFile.Lines.LoadFromFile(OD.Filename);
```

```
    gParsed.RowCount:=3;
    gParsed.ColCount:=3;
    Parse;
    RefreshColIdx();
  end;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  Process;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  edRootDir.Text := ExtractFilePath(Application.ExeName);
  BatchDir := edRootDir.Text+'batch\';
  DataDir := edRootDir.Text+'data\';
  ForceDirectories(BatchDir);
  PageControl1.ActivePageIndex := 0;

  //set initial column locations
  gColumns.ColCount := 2;
  gColumns.Cells[0,0] := 'Name';
  gColumns.Cells[1,0] := 'Column';
  SettingCount := 0;

  AddSetting('RunName',1);
  AddSetting('BOI',2);
  AddSetting('ALE',3);
  AddSetting('LSLRepPct',4);
  AddSetting('LSLOption',5);
  AddSetting('TempPOUOption',6);
  AddSetting('LSLFilters',7);
  AddSetting('SmallProxyPop',8);
  AddSetting('LSLRCap',9);
  AddSetting('IQValueSens',10);
  AddSetting('RunType',11);
  AddSetting('DiscountRate',12);
  AddSetting('BaselineName',13);
  AddSetting('OptionName',14);
  AddSetting('BasePWSDataFile',15);
  AddSetting('ScenPWSDataFile',16);
  AddSetting('LCRBaselineDB',17);
  AddSetting('BaseCostSteps',18);
  AddSetting('BaseVarData',19);
  AddSetting('ScenCostSteps',20);
  AddSetting('ScenVarData',21);
  AddSetting('CCTToFull',22);
  AddSetting('CCTCostLevel',23);
  AddSetting('SchoolOption',24);
```

```
  AddSetting('LCRBaseline',25);

  RefreshColIdx();
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  if Assigned(Config) then Config.Free;
end;

procedure TForm1.gColumnsExit(Sender: TObject);
begin
  RefreshColIdx();
end;

function TForm1.GetIndex(s: string): integer;
begin
  Result := -1;
  for var i:= 1 to gColumns.RowCount-1 do begin
    if SameText(s,gColumns.Cells[0,i]) then begin
      Result := gColumns.Cells[1,i].ToInteger();
      break;
    end;
  end;
end;

function TForm1.GetColNameIndex(s: string): integer;
begin
  Result := -1;
  for var i:= 1 to gColumns.RowCount-1 do begin
    if SameText(s,gColumns.Cells[0,i]) then begin
      Result := i;
      break;
    end;
  end;
end;


procedure TForm1.Parse;
begin
  var T := TStringLIst.Create;
  T.StrictDelimiter := True;
  var r := 1;
  for var i  := 0 to RawFile.Lines.Count-1 do begin
    gParsed.RowCount := max(gParsed.RowCount,r);
    T.CommaText := RawFile.Lines[i];
    gParsed.ColCount := max(gParsed.ColCount,T.Count);
    for var c := 0 to T.Count-1 do begin
      gParsed.cells[c,r] := T[c];
    end;
```

```
    inc(r);
  end;
  T.Free;
end;

procedure TForm1.Process;
begin
  const EXEname = edEXEName.Text;
  const ImWaiting =
StringReplace(Waiting,'$safewaterlcrr$',ChangeFileExt(EXEname,''), [rfReplaceAll]);
  const wait = 'Start-Sleep -Seconds 1';
  var suffix := '.bat';
  var MaxFileCnt := strtoint(edMax.Text);
  if MaxFileCnt <=0 then MaxFileCnt := MaxInt;
  if cbSimul.Checked then
   suffix := '.ps1';

  var BFile := TStreamWriter.Create(BatchDir+edBatchName.Text+suffix);
  BFile.WriteLine('cd '+edRootDir.Text);
  const OutDir = BatchDir+edBatchName.Text+'\';
  ForceDirectories(OutDir);
  var prefix := '';
  var cnt := 0;
  var totfiles := 0;
  if cbSimul.Checked then prefix := 'start ';
  if cbInc.Checked and cbSimul.Checked then prefix := prefix +'"" ';

  var nIX := GetIndex('RunName');
  var oIX := GetIndex('BOI');
  for var incr := 0 to 1 do begin
    if (incr = 1) and (cbSimul.Checked) then BFile.Write(ImWaiting);
    cnt := 0;
    for var r := 2 to gParsed.RowCount do begin
      try
        if gParsed.Cells[0,r].ToInteger = 0 then continue;
      except
        continue;
      end;

      //order incremental runs at ened - and after waiting
      var rt := gParsed.Cells[oIX,r].ToInteger;
      if (incr=1) and (rt<2) then continue;
      if (incr=0) and (rt=2) then continue;

      if Assigned(Config) then Config.Free;
      Config := TLCRConfig.Create;
      if cbFastRun.Checked then Config.FastRun := 1;
      //other assumed defaults - probably should make sure evrything in config
default state is good
      Config.SystemType := sysCWS;
```

```
      Config.RunSysType := 'Both';
      Config.SmallSystemFlexibility := True;
      Config.PWS90PctBp1 := 10;
      Config.PWS90PctBp2 := 15;
      Config.VariabilityRun := true;
      Config.MeanUncertLevel := false;
      Config.NumberOfVLoops := 1;
      Config.CCTCostEquations := DataDir + 'CCTCostEquations.xlsx';

      for var c := 0 to high(Found) do Found[c] := false;

      var OutFileName := OutDir+gParsed.Cells[nIX,r]+'.swc';
      var l := prefix + '"'+edrootdir.Text+EXEName + '"'+' "' + OutFileName +'"';
      BFile.WriteLine(l);
      if cbSimul.Checked then BFile.WriteLine(wait);
      //set switches, etc.
      for var c := 1 to gColumns.RowCount-1 do begin
        var value := gParsed.Cells[gColumns.Cells[1,c].ToInteger(),r];
        ProcessSetting(gColumns.Cells[0,c], value);
      end;
      //make sure all settings found - redundant after first line
      for var c := 0 to high(Found) do
        if not Found[c] then showmessage('setting not set:'+gColumns.Cells[0,c+1]);

      Config.Save(OutFileName);
      inc(totfiles);
      inc(cnt);
      if (cnt>= MaxFileCnt) and (cbSimul.Checked)  then begin
        BFile.Write(ImWaiting);
        cnt:=0;
      end;
    end;
  end;
  BFile.Free;
  memOut.Lines.LoadFromFile(BatchDir+edBatchName.Text+suffix);
  showmessage(Format('%d total run files created',[totfiles]));
end;

function TForm1.iSameText(SettingName, Value: string): boolean;
begin
  Result := False;
  if SameText(SettingName,Value) then begin
    var i := self.GetColNameIndex(SettingName);
    Found[i-1] := true;
    Result := True;
  end;
end;

function TForm1.ProcessSetting(SettingName, Value: string): boolean;
begin
```

```
  //just find it and set it.
  var tmp: string;
  if iSameText(SettingName,'Number') then begin
    //do nothing
  end else
  if iSameText(SettingName,'RunName') then begin
    Config.RunName := Value;
  end else
  if iSameText(SettingName,'ALE') then begin
    Config.ALE := value.ToInteger();
  end else
  if iSameText(SettingName,'LSLRepPct') then begin
    tmp := StringReplace(Value,'%','',[rfReplaceAll]);
    Config.LSLRepPct := tmp.ToDouble / 100;
  end else
  if iSameText(SettingName,'LSLOption') then begin
    Config.LSLOption := value.ToInteger();
  end else
  if iSameText(SettingName,'TempPOUOption') then begin
    Config.TempPOUOption := value.ToInteger();
  end else
  if iSameText(SettingName,'LSLFilters') then begin
    Config.LSLFilters := value.ToInteger();
  end else
  if iSameText(SettingName,'SmallProxyPop') then begin
    Config.SmallProxyPop := value.ToInteger();
  end else
  if iSameText(SettingName,'LSLRCap') then begin
    Config.LSLRCap := value.ToDouble();
  end else
  if iSameText(SettingName,'IQValueSens') then begin
    if SameText(value,'Lin') then Config.IQValueSens := 1;
  end else
  if iSameText(SettingName,'RunType') then begin
    if SameText(value,'high') then Config.RunType := rtHigh else Config.RunType :=
rtLow;
  end else
  if iSameText(SettingName,'DiscountRate') then begin
    tmp := StringReplace(Value,'%','',[rfReplaceAll]);
    Config.DiscountRate := tmp.toDouble()/ 100;
  end else
  if iSameText(SettingName,'OptionName') then begin
    Config.OptionName := Value;
  end else
  if iSameText(SettingName,'BaselineName') then begin
    Config.BaselineName := Value;
  end else
  if iSameText(SettingName,'BasePWSDataFile') then begin
    Config.BasePWSDataFile := Value;
    if pos(':',Config.BasePWSDataFile)<=0 then Config.BasePWSDataFile :=
```

```
DataDir+Config.BasePWSDataFile;
  end else
  if iSameText(SettingName,'ScenPWSDataFile') then begin
    Config.ScenPWSDataFile := Value;
    if pos(':',Config.ScenPWSDataFile)<=0 then Config.ScenPWSDataFile :=
DataDir+Config.ScenPWSDataFile;
  end else
  if iSameText(SettingName,'LCRBaselineDB') then begin
    Config.LCRBaselineDB := Value;
    if pos(':',Config.LCRBaselineDB)<=0 then Config.LCRBaselineDB :=
DataDir+Config.LCRBaselineDB;
  end else
  if iSameText(SettingName,'BaseCostSteps') then begin
    Config.BaseCostSteps := Value;
    if pos(':',Config.BaseCostSteps)<=0 then Config.BaseCostSteps :=
DataDir+Config.BaseCostSteps;
  end else
  if iSameText(SettingName,'BaseVarData') then begin
    Config.BaseVarData := Value;
    if pos(':',Config.BaseVarData)<=0 then Config.BaseVarData :=
DataDir+Config.BaseVarData;
  end else
  if iSameText(SettingName,'ScenCostSteps') then begin
    Config.ScenCostSteps := Value;
    if pos(':',Config.ScenCostSteps)<=0 then Config.ScenCostSteps :=
DataDir+Config.ScenCostSteps;
  end else
  if iSameText(SettingName,'ScenVarData') then begin
    Config.ScenVarData := Value;
    if pos(':',Config.ScenVarData)<=0 then Config.ScenVarData :=
DataDir+Config.ScenVarData;
  end else
  if iSameText(SettingName,'CCTToFull') then begin
    if value='1' then Config.CCTToFull := true else Config.CCTToFull:=false;
  end else
  if iSameText(SettingName,'CCTCostLevel') then begin
    Config.CCTCostEquationLevel := value;
  end else
  if iSameText(SettingName,'SchoolOption') then begin
    Config.SchoolOption := lowercase(value);
  end else
  if iSameText(SettingName,'LCRBaseline') then begin
    Config.UseLCRBaseline := value.ToInteger();
  end else
  if iSameText(SettingName,'BOI') then begin
    Config.RunBaselineOnly := false;
    Config.RunOptionOnly := false;
    Config.RunDifference := false;
    if value = '0' then
      Config.RunBaselineOnly := true
```

```
    else if value = '1' then
      Config.RunOptionOnly := true
    else
      Config.RunDifference := true;
  end else
    ShowMessage('not found:'+SettingName);
end;

procedure TForm1.RefreshColIdx;
begin
  for var R := 1 to SettingCount do begin
    var c := gColumns.Cells[1,r];
    gParsed.cells[strtoint(c),0] := gColumns.Cells[0,r] + ' ('+c+')';
  end;
end;

end.
```