```
unit CostingSteps;

interface

uses SysUtils, StrUtils, Math, Generics.Collections, Classes, Parser, ValueTypes,
ValueUtils,
     ParseTypes, Contnrs, System.IOUtils,
     DB, SafewaterUncertBucket,CodeSiteLogging,
     LCRCostVars, LCRGlobals, CCTCostEquations, LCRConfig, StateSchoolSampData,
     LCRCompiledCost;

type
  TCostStepRec = record
    ID : integer;
    CostName: string;
    ProbabilityCost: string;

    TotalCost: string;
    Hours: string;
    Labor: string;
    OM: string;
    Domain: string;
    OutputCostGroup: string;
    Frequency: string;
    Agglomerator1Xw: string;
    Agglomerator2Xw: string; // Column R in cost logic spreadsheet
    CostNo : string; // from input spreadsheet...
    Year: string;
    ICRRow : string;
    AgglomeratorICR: string;
    LSLRCost : boolean;
    IncludeCost: string;
    VLSEpLevel : boolean;
    Bin1: integer;
    Bin2: integer;
    Bin3: integer;
    Bin4: integer;
  end;

  TCostingStep=class
  private
    P : TParser;  //pointer to parent Parser - too much memory needed for
individuals
    fCostStepRec : TCostStepRec;
    fCC : TLSRCompiledCost;
    pCost,pLabor,pOM,pHours,pIn : TScript;
    fMyVars : TStringList;
    fOKP,fOKC,fOKH,fOKO,fOKL,fOKAll : boolean;
    fError:string;
    fCostVars : TCostVars;
    fAgg2ID,fAgg2IDO,fAgg2IDH,fAgg2IDL : integer;
    fAgg1ID,fAgg1IDO,fAgg1IDH,fAgg1IDL : integer;
    fAggICR_IDH1,fAggICR_IDH2,fAggICR_IDH3,fAggICR_IDH4,fAggICR_IDH10 : integer;
    fAggICR_IDC1,fAggICR_IDC2,fAggICR_IDC3,fAggICR_IDC4,fAggICR_IDC10 : integer;
    fImAStateCost, fCCTCost, fVLSEpLevel : boolean;
    fBaselineCCTInstall, fBaselineCCTModify, fBabyBlueCCTInstall, 
fBabyBlueCCTModify: boolean;
    fBaselineCCTModify_ale, fBaselineCCTModify_tle: boolean;
    fOWCCTInstall, fOWCCTModify, fOWCCTModify_ale, fOWCCTModify_tle: boolean;
    fSysLSLRCapital, fHhLSLRCapital, fSysCCTCapital: boolean;
    farrCalculateYr: array[1..100] of boolean;
    DirArrCalculateYr: array[1..100] of boolean;
    fEvalCount : int64;
    fDEBUG : boolean;
    fOneTimeCost: boolean;

    procedure AddVars(s : string);
    procedure pSetup(var TS : TScript; s : string; var fOK : boolean);
    procedure InitArrCalculateYr(IsBaseline: boolean);
    procedure ResetArrCalculateYr(IsBaseline : boolean);
    function getArrCalculateYr(Index: integer): boolean;
    procedure SetArrCalculateYr(Index: integer; const Value: boolean);
  public
    constructor Create(aCostStepRec : TCostStepRec; aCostVars : TCostVars; aP : 
TParser; IsBaseline: boolean; aCC : TLSRCompiledCost);
    destructor Destroy; override;

    procedure Evaluate(var Cost,Labor, OM,Hours, DoIt: double);
    procedure EvaluateState(var Cost,Labor, OM,Hours, DoIt: double);

    property Ok : boolean read fOkAll;
    property Error : string read fError;
    property EvalCount : int64 read fEvalCount;
    property CCTCost : boolean read fCCTCost;
    property BaselineCCTInstall : boolean read fBaselineCCTInstall;
    property BaselineCCTModify : boolean read fBaselineCCTModify;
    property BabyBlueCCTInstall : boolean read fBabyBlueCCTInstall;
    property BabyBlueCCTModify : boolean read fBabyBlueCCTModify;
    property SysLSLRCapital : boolean read fSysLSLRCapital;
    property HhLSLRCapital : boolean read fHhLSLRCapital;
    property SysCCTCapital : boolean read fSysCCTCapital;
    property OWCCTInstall : boolean read fOWCCTInstall;
    property OWCCTModify : boolean read fOWCCTModify;
    property OWCCTModify_ale : boolean read fOWCCTModify_ale;
    property OWCCTModify_tle : boolean read fOWCCTModify_tle;
    property OneTimeCost: boolean read fOneTimeCost;
    property ArrCalculateYr[Index: integer] : boolean read getArrCalculateYr write 
SetArrCalculateYr;
  end;

  TCustomVarIdxs = record
    EP, Pws_Cct, P_Fail, num_lsl_replace, Meet_Lslr_Goal, Pws_first_ale,
    Pws_sw, Pws_gw, Pws_pop, Numb_hh : integer;

    p_sal_one, p_sal_two, p_sal_three, p_adjust_cct_sal, p_wqp_chng, p_wqp_chng_adj,
    p_nonagg, p_copper_agg_adj,
    p_source_chng, p_adjust_cct_source_chng,p_cct_guide_apply, p_cct_guid_chng,
    p_treat_change, p_adjust_cct_treat_chng, p_source_sig : integer;

    p_install_cct_sal, p_lead_agg, p_lead_agg_inst,p_copper_agg_inst,
    p_install_cct_source_chng, p_install_cct_treat_chng : integer;

    p_lsl, perc_lsl, pp_lsl_replaced_one, pp_lsl_replaced_two,
    pp_lsl_replaced_three, lslr_goal_one, lslr_goal_two, lslr_goal_three : integer;

    p_inventory, p_tap_nine, p_tap_annual, p_tap_triennial, p_spec_req,
    p_wqp_annual, p_wqp_triennial, p_wqp_six_red, p_b3: integer;

    p_copper_ale, p_adjust_cct_copper, p_lead_ale, p_install_cct_copper: integer;

    fail_nm_1_2_3, fail_nm_4, fail_nm_5, fail_nm_6, fail_nm_7,
    fail_nm_8_9_10, hh_remain_lsl: integer;

    b_wqp_chng_adj, b_copper_agg_adj, b_adjust_cct_source_chng,
    b_cct_guid_chng, b_adjust_cct_treat_chng, b_lead_agg_inst,
    b_copper_agg_inst, b_install_cct_source_chng, b_install_cct_treat_chng: integer;

    b_adjust_cct_copper, b_install_cct_lead_ale, b_install_cct_copper_ale,
    b_install_cct_treat: integer;

    cct_existing_cost, cct_modify_cost, cct_install_cost, cct_findfix_cost: integer;
    cct_modify_cost_umra, cct_install_cost_umra, cct_findfix_cost_umra: integer;
    cct_modify_cost_umra_om, cct_install_cost_umra_om, cct_findfix_cost_umra_om: 
integer;
    cct_modify_cost_p, cct_install_cost_p, cct_findfix_cost_p: integer;

    pbaseph, pbasepo4, pbasephpo4, baselinepo4dose, baselineph_w, baselineph_wo,
    targetph, targetpo4: integer;

    hh_consumption: integer;

    b_adjust_cct_sal_p1, b_adjust_cct_sal_p2, b_adjust_cct_sal_p3,
    b_install_cct_sal_p1, b_install_cct_sal_p2, b_install_cct_sal_p3: integer;
    p_lead_ale_one, p_lead_ale_two, p_lead_ale_three: integer;

    b_adjust_cct_lead_plat_1, b_adjust_cct_lead_plat_2, b_adjust_cct_lead_plat_3,
    b_copper_agg_adj_plat, b_adjust_cct_source_chng_plat,
b_adjust_cct_treat_chng_plat: integer;

    b_install_cct_lead_plat_1, b_install_cct_lead_plat_2, b_install_cct_lead_plat_3,
    b_copper_agg_inst_plat: integer;

    pp_lslr_partial, pp_lslr_paper: integer;

    p_green_cct_adjust, p_green_cct_install,
    b_modify_cct_50_lsl, b_install_cct_50_lsl,
    b_adjust_cct_lead_green_1, b_adjust_cct_lead_green_2, b_adjust_cct_lead_green_3,
    b_install_cct_lead_green_1, b_install_cct_lead_green_2,
b_install_cct_lead_green_3,
    adjust_cct, install_cct,
    b_copper_agg_adj_green, b_copper_agg_inst_green,
    b_install_cct_source_chng_green, b_adjust_cct_source_chng_green,
    b_install_cct_treat_chng_green, b_adjust_cct_treat_chng_green,
    num_hh_per_connect, pws_fail, lslr_green_rate : integer;

    b_cct_guid_chng_five: integer;

    b_adjust_cct_treat,
    b_install_cct_lead_ale_one, b_install_cct_lead_ale_two,
b_install_cct_lead_ale_three: integer;

    p_sanit_surv_chng, b_cct_sanitary_survey, p_ss_cct_guide_apply: integer;

    pp_lcr_test, pp_lcr_test_yes, num_lsl_paper: integer;

    dist_lead_base_bin1, dist_lead_base_bin2, dist_lead_base_bin3,
    bin_distr, p_bin3_nonzero: integer;

    pp_lsl_replaced_vol_goal, pp_lsl_replaced_vol_pct, rnd_p90_error: integer;
    b_modify_cct, b_install_cct, b_install_pou, system_pou, b_modify_cct_mc,
b_install_cct_mc: integer;

    numb_reduced_tap, numb_samp_customer,
    pp_above_al_bin_one, pp_above_al_bin_two, pp_above_al_bin_three: integer;

    b_findfix: integer;

    b_lslr_study, b_pou_study, b_lslr_mand, b_lslr_vol, b_lslr_requested: integer;

    school_1a,school_1b,school_3a,school_3b,school_3c,school_3d,school_5a,school_5b:
integer;
```

```
    post_cct_p90_bin1, post_cct_p90_bin2, post_ff_p90_bin1, post_ff_p90_bin2 :
integer;

    p_cct_study, b_cct_study_install, b_cct_study_rec_install,
b_state_cct_treatment_install: integer;
    b_cct_study_mod, b_cct_study_rec_mod, b_state_cct_treatment_mod: integer;

    fail_nm1, fail_nm2: integer;

    num_vol_leadtap_samples_per_k, p_vol_leadtap_prog, hrs_act_wqp_op, cost_act_wqp,
rate_op: integer;

    b_cct_study_rec_mod_tl, b_cct_study_mod_tl, b_modify_cct_tl,
b_state_cct_treatment_mod_tl, b_cct_study_rec_mod_al,
    b_cct_study_mod_al, b_modify_cct_al, b_state_cct_treatment_mod_al: integer;

    numb_wqp_sites_added, numb_wqp_sites_added_prev, pp_overlap_find_fix,
    numb_reduced_wqp, numb_enhance_wqp: integer;

    num_lsl_requested, pp_cust_init_lslr, annual_pou_cost_hh: integer;

    numb_second_schools_pub, numb_elem_schools_pub, numb_second_schools_priv,
    numb_elem_schools_priv, numb_daycares, p_grandfather_opt_pub,
p_grandfather_opt_priv,
    p_grandfather_mand_pub, p_grandfather_mand_priv, p_grandfather_opt_child,
    p_grandfather_mand_child: integer;

    b_state_one, b_state_two: integer;
    num_paper_remain, num_lsl_remain: integer;
  end;

  TCostingSteps = class
    private
      RowNo: integer;
      fErrors : TstringList;
      fCostVars : TCostVars;
      fParser : TParser;
      fYears, fYearsOutput : integer;

      fI : TCustomVarIdxs;
      fPA : array of double;
      fPI : array of double;

      slVariables, slCosts, slCalcs, slCCTCosts: TStringList;
      fIsBaseline : boolean;

      fAdjust_CCT, fInstall_CCT: integer;
      GBinA: array [1..16] of double;
      GBin: array [1..16] of double;

      arrPBinBaseline: array[0..1, 0..1, 0..2] of double;
      arrPBinOption: array[0..1, 0..1, 0..2] of double;
      arrPBinOption5L: array[0..1, 0..1, 0..2] of double;

      fnum_proxies: integer;

      procedure Add(IsBaseline: boolean; aCostNo : string; aCostName,
aProbabilityCost, aTotalCost, aHours, aLabor, aOM,
                    aDomain, aOutputCostGroup,
                    aFrequency, aAgglomerator1Xw, aAgglomerator2Xw, aYear, aICR,
aAgglomeratorICR, aIncludeCost: string;
                    aVLSEpLevel, aBin1, aBin2, aBin3, aBin4: string; id : integer);
      procedure ReadSteps(Filename: string; Filter: string);

      function calc_pws_lslr_base(baseline: boolean): integer; overload;
      function calc_pws_lslr_base(baseline: boolean; yr: integer): integer;
overload;
      function Discount(const Value : double; const Yrs : integer; const Rate :
double) : double;
      function CCTAdjustChoice(option: string) : integer;
      function CCTInstallChoice(option: string) : integer;
      procedure InitCCTBVarsToZero(option: string);
      function GetHHConsumption: double;
      procedure LeadConcentrationBins(pwsid, option: string; yr, aSz, aSrc, aLSL,
CCT, POU, aPop, aNC,
                                      fAdjust_CCT, fInstall_CCT,
                                      cct_adjust_yr, cct_install_yr, pou_install_yr:
integer;
                                      bCCT_Change: boolean;
                                      pwswgt, num_lsl_replaced2, num_lsl_requested,
num_lsl_remain: double;
                                      Num_Proxies, partial_cct_level: integer;
                                      hp_lslr_paper: array of double;
                                      hp_lslr_partial: array of double;
                                      pp_lsl_replacement_rates: array of double;
ResetBins : boolean=false);
      function calc_prob_downstream_P_limit(baseline: boolean; year: integer):
integer;
      function get_category_value(category: string): double;
    public
      CostSteps: TObjectList<TCostingStep>;
      AggInputs2, AggInputs1, AggICR : TArray<string>;

      //holds current values of all variables for steps..
      Variables : TDoubleArray;
      RawVariables: TDoubleArray;
      //hold final results of all aggregated calculations
      Values2, Values1 : array[0..1000] of double;
      Values2p, Values1p : array[0..1000] of double;
      ValuesICR : array[0..1000] of double;
      //Undiscounted values by year.....
      Values2Y : array[1..100,0..1000] of double;
      //identifies balues that are in hourse for annualization...
      HourValue2, HourValue1 : array[0..1000] of boolean;
      POTWCost: double;

      prerule_ploading_lbs_5, prerule_ploading_lbs_15, prerule_ploading_lbs_25,
prerule_ploading_lbs_35: double;
      postrule_ploading_lbs_5, postrule_ploading_lbs_15, postrule_ploading_lbs_25,
postrule_ploading_lbs_35: double;
      incr_ploading_lbs_5, incr_ploading_lbs_15, incr_ploading_lbs_25,
incr_ploading_lbs_35: double;
      count_incr_ploading_lbs_5, count_incr_ploading_lbs_15,
count_incr_ploading_lbs_25, count_incr_ploading_lbs_35: integer;

      StateCost,
      StateICRHours1, StateICRHours2, StateICRHours3, StateICRHours4,
StateICRHours10 : double;
      StateICRCost1, StateICRCost2, StateICRCost3, StateICRCost4, StateICRCost10 :
double;

      // hold undiscounted capital cost
      // 0 - System LSLR capital cost
      // 1 - Household LSLR capital cost
      // 2 - CCT capital cost
      ValuesCapital: array[0..2] of double;

      PWS_p_values: TPWS_p_valuesRec;
      DiscRate : double;
      CheckPWS: string;

      // These indicate if the PWS has any of the two costs in the 35 years and
      // report back to LCRCosts.GenerateCosts
      HasLSLRCost: boolean;
      HasCCTCost: boolean;
      LSLReplaced: double;
      LSLReplacedMandatory: double;
      LSLReplacedVoluntary: double;
      CCTInstalled: boolean;
      CCTAdjusted, CCTAdjusted_ale, CCTAdjusted_tle: boolean;
      CCTExisting: boolean;
      HasFindAndFixCost: boolean;
      POUInstalled: boolean;
      LSLRequested: double;

      // These variables are for the very large systems
      HasLSLRCostVLS: boolean;
      HasCCTCostVLS: boolean;
      LSLReplacedMandatoryVLS: double;
      LSLReplacedVoluntaryVLS: double;
      LSLReplacedVLS: double;
      LSLReplacedPopVLS: double;
      LSLRequestedVLS: double;
      LSLRequestedPopVLS: double;
      LSLReplacedPopMandatoryVLS: double;
      LSLReplacedPopVoluntaryVLS: double;
      CCTInstalledVLS: boolean;
      CCTAdjustedVLS, CCTAdjustedVLS_ale, CCTAdjustedVLS_tle: boolean;
      CCTExistingVLS: boolean;
      HasFindAndFixCostVLS: boolean;
      POUInstalledVLS: boolean;
      CCTInstalledPopVLS: double;
      CCTAdjustedPopVLS, CCTAdjustedPopVLS_ale, CCTAdjustedPopVLS_tle : double;
      CCTExistingPopVLS: double;
      HasFindAndFixCostPopVLS: double;
      POUInstalledPopVLS: double;
      SystemAFlowVLS: double;

      POTWCostVLS: double;

      prerule_ploading_lbs_5VLS, prerule_ploading_lbs_15VLS,
prerule_ploading_lbs_25VLS, prerule_ploading_lbs_35VLS: double;
      postrule_ploading_lbs_5VLS, postrule_ploading_lbs_15VLS,
postrule_ploading_lbs_25VLS, postrule_ploading_lbs_35VLS: double;
      incr_ploading_lbs_5VLS, incr_ploading_lbs_15VLS, incr_ploading_lbs_25VLS,
incr_ploading_lbs_35VLS: double;
      count_incr_ploading_lbs_5VLS, count_incr_ploading_lbs_15VLS,
count_incr_ploading_lbs_25VLS, count_incr_ploading_lbs_35VLS: integer;

      strLeadConcentrations: TBufferedFileStream;
      Config: TLCRConfig;
      GMoveBin :  TYearlyMovement;
      GMoveBinMicro : TYearlyMovementMicro;
      GMoveBinTotal : TMovementMicro;
      CC : TLSRCompiledCost;

      pws90pctCCT_yr : array[0..100] of double;
      pws90pctLSL_yr : array[0..100] of double;

      constructor Create(aCostVars : TCostVars; Filename: string; Filter: string;
aYears,aYearsOutput : integer;  IsBaseline : boolean);
      destructor Destroy; override;
      function CheckAggAgreement(T : TCostingSteps) : boolean;

      procedure SetVariablesAndCalculate(const aSz, aSrc, aLSL, aCCT, aType, aEP,
aNC, aFirstAle : integer;
                                         const aPop: integer; const CostCapital :
double;
                                         const SetProbsTo01 : boolean; const pwsid,
option: string;
                                         const CCTCostEquations: TCCTCostEquations;
const num_lsl, pwswgt: double;
                                         const O : TDictionary<string,double>; const
prtDebug, VLSystem: boolean;
                                         const Small_Correct: integer; const Bin:
integer; const P90_base: double;
                                         const Num_Proxies: integer; slProxies:
TStringList; const SchoolSampData: TSchoolSampDataRec);

      procedure SetVariablesAndCalculateVLS(const CostingData : TCostGenRec;
                                            const AddCostingData: TAddCostGenRec;
                                            const VLSEpWorkbook: TVLSEpWorkbookRec;
                                            const SetProbsTo01 : boolean; const
option: string;
                                            const CCTCostEquations:
TCCTCostEquations;
                                            const O : TDictionary<string,double>;
const SchoolSampData: TSchoolSampDataRec; const ResetBin : boolean=false);

      procedure Annualize(const CostCapital : double);

      procedure StateCostsCalculate(const SetProbsTo01 : boolean; option: string);
      procedure SetRandomYears;
      procedure ResetRandomSeeds(PWSSeed : integer);

      procedure DetermineSystemPValues(const aSz, aSrc, aLSL, aCCT, aType : integer;
const SetProbsTo01: boolean; const PWSId: string; GetBaseCurValue : boolean=false);
      procedure ResolveDatabaseVariables(const aSz, aSrc, aLSL, aCCT, aType,
aPWS90PctBp1, aPWS90PctBp2: integer); overload;
      procedure ResolveDatabaseVariables(const aSz, aSrc, aLSL, aCCT, aType,
aPWS90PctBp1, aPWS90PctBp2: integer;
                                         const BaselineCols, BaselineData:
TStringList); overload;
      function AssignNewBinBaseline(CCT, LSL: integer): integer;
      function AssignNewBinOption(CCT, LSL: integer; Option: string): integer;

      property Errors : TStringList  read fErrors;
      property HHConsumption: double read GetHHConsumption;
      property CostVars: TCostVars read fCostVars;
  end;

implementation

uses VCL.FlexCel.Core, FlexCel.XlsAdapter;

{$R+}

//{$DEFINE NOTRIGCCT}
//{$DEFINE NOGOALLCR}

//{$DEFINE NDWAC_TEST}
//{$DEFINE BASELINE_3PctReplacement}

{
g.      Calculate the number of LSLR done each year (Num_lsl_replacew,y*):
a.      Num_lsl_replacew,y*  = min ( Num_lsl_remain w,y , max
(Num_lsl_replace_alew,y*, (Num_LSL_basew * p_lsl_replacedw,y*)))
(Assuming max of ale triggered and proactive replacements)

b.      Num_lsl_replacew,y*  = min ( Num_lsl_remain w,y ,  (Num_lsl_replace_alew,y*
+ (Num_LSL_basew * p_lsl_replacedw,y*)))
(Assuming sum of ale triggered and proactive replacements)

c.      Num_lsl_replacew,y*  = min ( Num_lsl_remain w,y ,  (Num_LSL_basew *
p_lsl_replacedw,y*))
(Assuming just proactive replacements)
}

//{$DEFINE BABYBLUE_Max}
//{$DEFINE BABYBLUE_Additive}
//{$DEFINE BABYBLUE_Proactive}

{
  set pp_lsl_replacement_rates to data, 5% or 10%
}

{$DEFINE LSLReplacementRates_Data}
//{$DEFINE LSLReplacementRates_05}
//{$DEFINE LSLReplacementRates_10}

{
  set Config Years of Analysis to 35, 10 or 20
}

{$DEFINE AnalysisYears_35}
//{$DEFINE AnalysisYears_10}
```

```
//{$DEFINE AnalysisYears_20}

{ TCostingSteps }

procedure TCostingSteps.Add(IsBaseline: boolean; aCostNo : string; aCostName,
aProbabilityCost, aTotalCost, aHours, aLabor,
                    aOM, aDomain, aOutputCostGroup,
                    aFrequency, aAgglomerator1Xw, aAgglomerator2Xw, aYear, aICR,
aAgglomeratorICR, aIncludeCost: string;
                    aVLSEpLevel, aBin1, aBin2, aBin3, aBin4: string; id : integer);
var T : TCostingStep;
    CostStep : TCostStepRec;
begin
  CostStep.ID := ID;
  CostStep.CostNo:=aCostNo;
  CostStep.CostName := aCostName;
  CostStep.ProbabilityCost := aProbabilityCost;
  //for missing probabilities - assume always calculated
  if CostStep.ProbabilityCost='' then
    CostStep.ProbabilityCost:='1';
  CostStep.TotalCost := aTotalCost;
  CostStep.Hours := aHours;
  CostStep.Labor := aLabor;
  CostStep.OM := aOM;
  CostStep.Domain := aDomain;
  CostStep.OutputCostGroup := aOutputCostGroup;
  CostStep.Frequency := aFrequency;
  CostStep.Agglomerator1Xw := aAgglomerator1Xw;
  CostStep.Agglomerator2Xw := aAgglomerator2Xw;
  CostStep.Year := aYear;
  CostStep.ICRRow:=aICR;
  CostStep.AgglomeratorICR := aAgglomeratorICR;
  CostStep.IncludeCost := aIncludeCost;
  if aVLSEpLevel = 'Y' then
    CostStep.VLSEpLevel := True
  else
    CostStep.VLSEpLevel := False;
  CostStep.Bin1 := aBin1.ToInteger;
  CostStep.Bin2 := aBin2.ToInteger;
  CostStep.Bin3 := aBin3.ToInteger;
  CostStep.Bin4 := aBin4.ToInteger;

  T:=TCostingStep.Create(CostStep, fCostVars, fParser, IsBaseline, CC);
  if not T.Ok then
    if T.Error <> '' then
      fErrors.Add('Line:'+inttostr(RowNo+2)+' Msg:'+T.Error);
  CostSteps.Add(T);
  Inc(RowNo);
end;

function TCostingSteps.Discount(const Value : double; const Yrs : integer; const
Rate : double) : double;
begin
  if Rate<0 then
    Result:=Value/PreCalcDiscRate[yrs]
  else
    Result:=Value/intpower((1+Rate),Yrs);
end;

function TCostingSteps.GetHHConsumption: double;
begin
  Result := Variables[fi.hh_consumption];
end;

function TCostingSteps.get_category_value(category: string): double;
var i: integer;
begin
  for i:=0 to high(AggInputs2) do begin
    if AggInputs2[i] = category then
    begin
      Result := Values2p[i];
      break;
    end;
  end;
end;

procedure TCostingSteps.InitCCTBVarsToZero(option: string);
begin
    Variables[fI.b_adjust_cct_copper] := 0;
    Variables[fI.b_adjust_cct_source_chng] := 0;
    Variables[fI.b_adjust_cct_treat_chng] := 0;

    Variables[fI.b_install_cct_lead_ale] := 0;
    Variables[fI.b_install_cct_copper_ale] := 0;
    Variables[fI.b_install_cct_source_chng] := 0;
    Variables[fI.b_install_cct_treat] := 0;

    Variables[fI.b_modify_cct] := 0;
    Variables[fI.b_install_cct] := 0;
    Variables[fI.b_modify_cct_mc] := 0;
    Variables[fI.b_install_cct_mc] := 0;
    Variables[fI.b_install_pou] := 0;
    Variables[fI.b_lslr_study] := 0;
    Variables[fI.b_pou_study] := 0;
    Variables[fI.b_lslr_mand] := 0;
    Variables[fI.b_lslr_vol] := 0;
    Variables[fI.b_lslr_requested] := 0;
end;

function CB(const c : integer) : integer;
//this converts the Bin Numbering in pops to BL table numbering.
begin
  Result:=C;
  exit;
  //short cicuiting to use this numbeing scheme instaed of the blood lead tables...
  if c=1 then result:=4 else
  if c=2 then result:=7 else
  if c=3 then result:=1 else
  if c=4 then result:=5 else
  if c=5 then result:=8 else
  if c=6 then result:=2 else
  if c=7 then result:=6 else
  if c=8 then result:=9 else
  if c=9 then result:=3 else
    result:=c;
end;


procedure TCostingSteps.LeadConcentrationBins(pwsid, option: string; yr, aSz, aSrc,
  aLSL, CCT, POU, aPop, aNC, fAdjust_CCT, fInstall_CCT, cct_adjust_yr,
  cct_install_yr, pou_install_yr: integer; bCCT_Change: boolean; pwswgt,
num_lsl_replaced2, num_lsl_requested, num_lsl_remain: double;
  Num_Proxies, partial_cct_level: integer;
  hp_lslr_paper: array of double; hp_lslr_partial: array of double;
pp_lsl_replacement_rates: array of double; ResetBins : boolean=false);
var
  pop_per_connection: double;
  pp_lsl_replaced: double;
  pp_lslr_partial, pp_lslr_paper: double;
  perc_lsl: double;
  sLine: string;
  GB1, GB2, GB3, GB4, GB5, GB6, GB7, GB8, GB9,
  Gb10, GB11, GB12, GB13, GB14, GB15, GB16: double;
  i, y, f, t: integer;
  GBSum: double;
  num_lsl_replaced: double;

  cpp_lsl_replaced, hcpp_lsl_replaced, zpp_lsl_replaced: double;
  cpp_lslr_partial, cpp_lslr_paper: double;
  hcpp_lslr_partial, hcpp_lslr_paper: double;

  procedure Up(ff,tt,yy : integer; pp : double);
  begin
```

```
    GMoveBinMicro[yy,ff,tt]:=GMoveBinMicro[yy,ff,tt] + Round(pp*10)/10;
    GMoveBinTotal[ff,tt]:=GMoveBinTotal[ff,tt] + Round(pp*10)/10;
  end;

begin
  if ResetBins then begin
    for y:=0 to high(GMoveBin) do begin
      for f:=low(GMoveBin[y]) to high(GMoveBin[y]) do begin
        GMoveBin[y,f]:=0;
        for t:=low(GMoveBinMicro[y,f]) to high(GMoveBinMicro[y,f]) do begin
          GMoveBinMicro[y,f,t]:=0;
          if y=0 then begin
            GMoveBinTotal[f,t]:=0;
          end;
        end;
      end;
    end;
  end;

  num_lsl_replaced := num_lsl_replaced2 + num_lsl_requested;

  if aNC > 0 then pop_per_connection := aPop / aNC
  else pop_per_connection := 0;

  pp_lsl_replaced := 0;
  pp_lslr_partial := 0;
  pp_lslr_paper := 0;

  perc_lsl := Variables[fI.perc_lsl];

  for i := 1 to 16 do GBin[i] := 0;

  if yr = 0 then
  begin
    hcpp_lsl_replaced := 0;
    hcpp_lslr_partial := 0;
    hcpp_lslr_paper := 0;

    // Create the following variables for each PWS by running a unique 35 year loop:
    // pp_lslr_partial, pp_lslr_paper

    for y := 1 to fYears do
    begin
      if option = 'Baseline' then
      begin
        if (y >= 4) then
        begin
          pp_lslr_partial := hp_lslr_partial[0];
        end;
      end
      else
      begin
        pp_lslr_paper := hp_lslr_paper[0];
        pp_lslr_partial := hp_lslr_partial[0];
        pp_lsl_replaced := pp_lsl_replacement_rates[y];
      end;

      cpp_lsl_replaced := hcpp_lsl_replaced + pp_lsl_replaced;

      if cpp_lsl_replaced >= 1 then
      begin
        cpp_lsl_replaced := 1;
        zpp_lsl_replaced := 1 - hcpp_lsl_replaced;
      end
      else
      begin
        cpp_lsl_replaced := cpp_lsl_replaced;
        zpp_lsl_replaced := pp_lsl_replaced;
      end;

      cpp_lslr_partial := hcpp_lslr_partial + (pp_lslr_partial * zpp_lsl_replaced);
      cpp_lslr_paper := hcpp_lslr_paper + (pp_lslr_paper * zpp_lsl_replaced);

      hcpp_lsl_replaced := cpp_lsl_replaced;
      hcpp_lslr_partial := cpp_lslr_partial;
      hcpp_lslr_paper := cpp_lslr_paper;
    end;

    if cpp_lsl_replaced > 0 then
    begin
      pp_lslr_partial := cpp_lslr_partial / cpp_lsl_replaced;
      pp_lslr_paper := cpp_lslr_paper / cpp_lsl_replaced;
    end;


    // Before the year loop, calculate for each PWS:

    // Bin 1: No CCT and LSL
    if CCT = 1 then GBinA[1] := 0
    else if aLSL = 0 then GBinA[1] := 0
    else GBinA[1] := aPop * perc_lsl * (1 - pp_lslr_partial);

    // BIN 2: No CCT and Partial LSL
    if CCT = 1 then GBinA[2] := 0
    else if aLSL = 0 then GBinA[2] := 0
    else GBinA[2] := aPop * perc_lsl * pp_lslr_partial;
```

```
//BIN 3: No CCT and No LSL
if CCT = 1 then GBinA[3] := 0
else if aLSL = 0 then GBinA[3] := aPop
else GBinA[3] := (aPop * (1-perc_lsl));


//BIN 4: Partial CCT and LSL
if CCT = 0 then GBinA[4] := 0
else if aLSL = 0 then GBinA[4] := 0
else GBinA[4] := aPop * perc_lsl * (1 - pp_lslr_partial);


GBinA[5] := 0;

GBinA[6] := 0;

//BIN 7: Partial CCT and Partial LSL
if CCT = 0 then GBinA[7] := 0
else if aLSL = 0 then GBinA[7] := 0
else GBinA[7] := aPop * perc_lsl * pp_lslr_partial;

GBinA[8] := 0;

GBinA[9] := 0;


//BIN 10: Partial CCT and No LSL
if CCT = 0 then GBinA[10] := 0
else if aLSL = 0 then GBinA[10] := aPop
else GBinA[10] := (aPop * (1-perc_lsl));

GBinA[11] := 0;

GBinA[12] := 0;


//BIN 13: Representative CCT and LSL
GBinA[13] := 0;

//BIN 14: Representative CCT and Partial LSL
GBinA[14] := 0;

//BIN 15: Representative CCT and No LSL
GBinA[15] := 0;

//BIN 16: POU
GBinA[16] := 0;
```

```
    // move shadow bins into bins that the benefits calculation will use
    for i := 1 to 16 do
    begin
       GBin[i] := GBinA[i];
       if (GBin[i] > -0.01) and (GBin[i] < 0) then GBin[i] := 0;
    end;
  end
  else
  // Every year calculate
  begin
    if option = 'Baseline' then
    begin
      if (yr >= 4) then
      begin
        pp_lslr_partial := hp_lslr_partial[0];
      end;
    end
    else
    begin
      pp_lslr_partial := hp_lslr_partial[0];
      pp_lslr_paper := hp_lslr_paper[0];
    end;

    // BIN 1: No CCT, No POU and LSL
    GB1 := GBinA[1];
    if CCT = 1 then GBinA[1] := 0
    else if POU = 1 then GBinA[1] := 0
    else if GB1 = 0 then GBinA[1] := 0
    else GBinA[1] := GB1 - (pop_per_connection * num_lsl_replaced * (1 - 
pp_lslr_partial));

    if (CCT=1) and (GB1>0) then begin
      Up(1,13,yr,GB1 - (pop_per_connection * num_lsl_replaced * (1 - 
pp_lslr_partial)));
      Up(1,15,yr,(pop_per_connection * num_lsl_replaced * (1 - pp_lslr_partial)));
    end else
    if POU=1 then begin
      Up(1,16,yr,GB1);
    end else
    if (POU=0) and (CCT=0) and (GB1>0) then begin
      Up(1,3,yr,(pop_per_connection * num_lsl_replaced * (1 - pp_lslr_partial)));
    end;


    // BIN 2: No CCT, No POU and Partial LSL
    GB2 := GBinA[2];
    if CCT = 1 then GBinA[2] := 0
    else if POU = 1 then GBinA[2] := 0
    else if GB2 = 0 then GBinA[2] := 0
    else GBinA[2] := GB2 - (pop_per_connection * num_lsl_replaced *
pp_lslr_partial);

    if (CCT=1) and (GB2>0) then begin
      Up(2,14,yr,GB2 - (pop_per_connection * num_lsl_replaced * (pp_lslr_partial)));
      Up(2,15,yr,(pop_per_connection * num_lsl_replaced * (pp_lslr_partial)));
    end else
    if POU=1 then begin
      Up(2,16,yr,GB2);
    end else
    if (POU=0) and (CCT=0) and (GB2>0) then begin
      Up(2,3,yr,(pop_per_connection * num_lsl_replaced * (pp_lslr_partial)));
    end;

    //BIN 3: No CCT, No POU and No LSL
    GB3 := GBinA[3];
    if CCT = 1 then GBinA[3] := 0
    else if POU = 1 then GBinA[3] := 0
    else if ((GB1 + GB2) = 0) then GBinA[3] := GB3
    else GBinA[3] := GB3 + (pop_per_connection * num_lsl_replaced);

    if CCT=1 then begin
      Up(3,15,yr,GB3);
    end else
    if POU=1 then begin
      Up(3,16,yr,GB3);
    end;

    //BIN 4: Partial CCT, No POU and LSL
    GB4 := GBinA[4];
    if POU = 1 then GBinA[4] := 0
    else if cct_adjust_yr = yr then GBinA[4] := 0
    else if GB4 = 0 then GBinA[4] := 0
    else GBinA[4] := GB4 - (pop_per_connection * num_lsl_replaced * (1 -
pp_lslr_partial));

    if POU=1 then begin
      Up(4,16,yr,GB4);
    end else
    if (cct_adjust_yr = yr) and (GB4>0) then begin
      Up(4,13,yr,GB4 - (pop_per_connection * num_lsl_replaced * (1 -
pp_lslr_partial)));
      Up(4,15,yr,(pop_per_connection * num_lsl_replaced * (1 - pp_lslr_partial)));
    end else
    if (bCCT_Change = false) and (GB4>0) then begin
      Up(4,10,yr,(pop_per_connection * num_lsl_replaced * (1 - pp_lslr_partial)));
    end;

    GB5 := 0;
    GBinA[5] := 0;

    GB6 := 0;
    GBinA[6] := 0;


    //BIN 7: Partial CCT, No POU and Partial LSL
    GB7 := GBinA[7];
    if POU = 1 then GBinA[7] := 0
    else if cct_adjust_yr = yr then GBinA[7] := 0
    else if GB7 = 0 then GBinA[7] := 0
    else GBinA[7] := GB7 - (pop_per_connection * num_lsl_replaced *
pp_lslr_partial);

    if POU=1 then begin
      Up(7,16,yr,GB7);
    end else
    if (cct_adjust_yr = yr) and (GB7>0)then begin
      Up(7,14,yr,GB7 - (pop_per_connection * num_lsl_replaced * (pp_lslr_partial)));
      Up(7,15,yr,(pop_per_connection * num_lsl_replaced * (pp_lslr_partial)));
    end else
    if (bCCT_Change = false) and (GB7>0) then begin
      Up(7,10,yr,(pop_per_connection * num_lsl_replaced * (pp_lslr_partial)));
    end;


    GB8 := 0;
    GBinA[8] := 0;

    GB9 := 0;
    GBinA[9] := 0;


    //BIN 10: Partial CCT, No POU and No LSL
    GB10 := GBinA[10];
    if POU = 1 then GBinA[10] := 0
    else if cct_adjust_yr = yr then GBinA[10] := 0
    else if ((GB4 + GB7) = 0) then GBinA[10] := GB10
    else GBinA[10] := GB10 + (pop_per_connection * num_lsl_replaced);

    if POU=1 then begin
      Up(10,16,yr,GB10);
    end else
    if cct_adjust_yr = yr then begin
      Up(10,15,yr,GB10);
    end;

    GB11 := 0;
    GBinA[11] := 0;

    GB12 := 0;
    GBinA[12] := 0;


    //BIN 13: Representative CCT, No POU and LSL
    GB13 := GBinA[13];
    if POU = 1 then GBinA[13] := 0
    else if cct_install_yr = yr then
      GBinA[13] := GB1 - (pop_per_connection * num_lsl_replaced * (1 -
pp_lslr_partial))
    else if cct_adjust_yr = yr then
      GBinA[13] := (GB4 + GB5 + GB6) - (pop_per_connection * num_lsl_replaced * (1 -
pp_lslr_partial))
    else if GB13 = 0 then GBinA[13] := 0
    else
      GBinA[13] := GB13 - (pop_per_connection * num_lsl_replaced * (1 -
pp_lslr_partial));

    if POU=1 then begin
      Up(13,16,yr,GB13);
    end else
    if (GB13>0) and (bCCT_Change) then begin
      Up(13,15,yr,(pop_per_connection * num_lsl_replaced * (1 - pp_lslr_partial)));
    end;


    //BIN 14: Representative CCT, No POU and Partial LSL
    GB14 := GBinA[14];
    if POU = 1 then GBinA[14] := 0
    else if cct_install_yr = yr then
      GBinA[14] := GB2 - (pop_per_connection * num_lsl_replaced * pp_lslr_partial)
    else if cct_adjust_yr = yr then
      GBinA[14] := (GB7 + GB8 + GB9) - (pop_per_connection * num_lsl_replaced *
pp_lslr_partial)
    else if GB14 = 0 then GBinA[14] := 0
    else
      GBinA[14] := GB14 - (pop_per_connection * num_lsl_replaced * pp_lslr_partial);

    if POU=1 then begin
      Up(14,16,yr,GB14);
    end else
    if (GB14>0) and (bCCT_Change) then begin
      Up(14,15,yr,(pop_per_connection * num_lsl_replaced * (pp_lslr_partial)));
```

```
  end;

  //BIN 15: Representative CCT, No POU and No LSL
  GB15 := GBinA[15];
  if POU = 1 then GBinA[15] := 0
  else if cct_install_yr = yr then
    GBinA[15] := GB3 + (pop_per_connection * num_lsl_replaced)
  else if cct_adjust_yr = yr then
    GBinA[15] := (GB10 + GB11 + GB12) + (pop_per_connection * num_lsl_replaced)
  else if ((GB13 + GB14) = 0) then GBinA[15] := GB15
  else
    GBinA[15] := GB15 + (pop_per_connection * num_lsl_replaced);

  if POU=1 then begin
    Up(15,16,yr,GB15);
  end;

  //BIN 16: POU
  GB16 := GBinA[16];
  if pou_install_yr = yr then GBinA[16]:= apop;

  // move shadow bins into bins that the benefits calculation will use
  for i := 1 to 16 do
  begin
     GBin[i] := GBinA[i];
     if (GBin[i] > -0.01) and (GBin[i] < 0) then GBin[i] := 0;
  end;
end;

GBSum := 0;
for i := 1 to 16 do begin
  GBSum := GBsum + GBin[i];
  GMoveBin[yr,cb(i)]:= GMoveBin[yr,cb(i)] + GBin[i];
end;

if Num_Proxies = 0 then
  if Config.OutputLeadBins then begin
    sLine := pwsid + ',' +
           aSz.ToString + ',' +
           aSrc.ToString + ',' +
           pwswgt.ToString + ',' +
           aPop.ToString + ',' +
           yr.ToString + ',' +
           GBin[1].ToString + ',' +
           GBin[2].ToString + ',' +
           GBin[3].ToString + ',' +
           GBin[4].ToString + ',' +
           GBin[5].ToString + ',' +
             GBin[6].ToString + ',' +
             GBin[7].ToString + ',' +
             GBin[8].ToString + ',' +
             GBin[9].ToString + ',' +
             GBin[10].ToString + ',' +
             GBin[11].ToString + ',' +
             GBin[12].ToString + ',' +
             GBin[13].ToString + ',' +
             GBin[14].ToString + ',' +
             GBin[15].ToString + ',' +
             GBin[16].ToString + ',' +
             GBSum.ToString + ',' +
             aLSL.ToString + ',' +
             CCT.ToString + ',' +
             pop_per_connection.ToString + ',' +
             num_lsl_replaced.ToString + ',' +
             pp_lslr_partial.ToString + ',' +
             pp_lslr_paper.ToString + ',' +
             cct_adjust_yr.ToString + ',' +
             cct_install_yr.ToString + ',' +
             perc_lsl.ToString + ',' +
             fAdjust_CCT.ToString + ',' +
             fInstall_CCT.ToString + ',' +
             num_lsl_remain.ToString + ',' +
             partial_cct_level.ToString +
             sLineBreak;

      strLeadConcentrations.WriteBuffer(sLine[1], Length(sLine)*SizeOf(Char));
    end;

end;

procedure TCostingSteps.Annualize(const CostCapital : double);
var v : double;
  i : integer;
  f: integer;
begin
  if fnum_proxies > 0 then f := fYears
  else f := fyearsoutput;

  v:=(DiscRate / (1 - IntPower((1 + DiscRate),-f)));
  for i:=0 to high(AggInputs2) do begin
    if HourValue2[i] then
      Values2[i]:=Values2[i]/f
    else
      Values2[i]:=Values2[i] * V;
  end;
  for i:=0 to high(AggInputs1) do begin
```

```
    if HourValue1[i] then
      Values1[i]:=Values1[i]/f
    else
      Values1[i]:=Values1[i] * V;
  end;

  POTWCost := POTWCost * V;

  v:=(CostCapital / (1 - IntPower((1 + CostCapital),-f)));
  for i:=0 to high(AggInputs2) do begin
    if HourValue2[i] then
      Values2p[i]:=Values2p[i]/f
    else
      Values2p[i]:=Values2p[i] * V;
  end;
  for i:=0 to high(AggInputs1) do begin
    if HourValue1[i] then
      Values1p[i]:=Values1p[i]/f
    else
      Values1p[i]:=Values1p[i] * V;
  end;
end;

function TCostingSteps.AssignNewBinBaseline(CCT, LSL: integer): integer;
var
  r: double;
  i, iBin: integer;
  tmp: double;
begin
  r := iiRandom(true);

  iBin := 0;
  tmp:=0;
  i:=1;
  repeat
    tmp:=tmp+arrPBinBaseline[CCT, LSL, i-1];
    iBin:=i;
    inc(i);
    if i>length(arrPBinBaseline[CCT, LSL]) then break;
  until tmp>=r;

  Result := iBin;
end;

function TCostingSteps.AssignNewBinOption(CCT, LSL: integer; Option: string):
integer;
var
  r: double;
  i, iBin: integer;
  tmp: double;
begin
  r := iiRandom(true);

  if Option = 'OW' then
  begin
    iBin := 0;
    tmp:=0;
    i:=1;
    repeat
      tmp:=tmp+arrPBinOption[CCT, LSL, i-1];
      iBin:=i;
      inc(i);
      if i>length(arrPBinOption[CCT, LSL]) then break;
    until tmp>=r;
  end
  else
  if Option = 'OW5L' then
  begin
    iBin := 0;
    tmp:=0;
    i:=1;
    repeat
      tmp:=tmp+arrPBinOption5L[CCT, LSL, i-1];
      iBin:=i;
      inc(i);
      if i>length(arrPBinOption5L[CCT, LSL]) then break;
    until tmp>=r;
  end;

  Result := iBin;
end;

function TCostingSteps.calc_prob_downstream_P_limit(baseline: boolean; year:
integer): integer;
var r, pp, baseProb: double;
begin
  // base probability of downstream POTW Phosphorus limit in 2016
  // annual growth rate of 3.3%
  baseProb := 0.132;
  pp := baseProb * IntPower(1.033, year-1);
  r := iiRandom(baseline);
  if r < pp then Result := 1
  else Result := 0;
end;

function TCostingSteps.calc_pws_lslr_base(baseline: boolean; yr: integer): integer;
```

```
var r, pp: double;
begin
  if Variables[fI.Pws_Cct] = 1 then
  begin
    case yr of
       4..18 : pp := 1-IntPower(1-RawVariables[fI.p_lead_ale_one],3);
      19..24 : pp := 1-IntPower(1-RawVariables[fI.p_lead_ale_two],3);
      25..100 : pp := 1-IntPower(1-RawVariables[fI.p_lead_ale_three],3);
    end;
    r := iiRandom(baseline);
    if r < pp then Result := 1
    else Result := 0;
  end
  else
    Result := 0;
end;

function TCostingSteps.calc_pws_lslr_base(baseline: boolean): integer;
var r, pp: double;
begin
  if Variables[fI.Pws_Cct] = 1 then
  begin
    pp := 1-IntPower(1-RawVariables[fI.p_lead_ale],3);
    // change iRandom to iiRandom 2/28/18
    r := iiRandom(baseline);
    if r < pp then Result := 1
    else Result := 0;
  end
  else
    Result := 0;
end;
function TCostingSteps.CCTAdjustChoice(option: string): integer;
var tp,r : double;
    i : integer;
begin
  Result := 0;
  if option = 'Baseline' then
  begin
    {
    tp:=(1-IntPower(1-RawVariables[fI.p_copper_ale],fYears-3)) *
RawVariables[fI.p_adjust_cct_copper];
    R:=iiRandom(True);
    if R<tp then Result:=1
    else Result := 0;
    }
    fPA[0]:=(1-IntPower(1-RawVariables[fI.p_copper_ale],fYears-3)) *
RawVariables[fI.p_adjust_cct_copper];
    fPA[1]:=(RawVariables[fI.p_tap_annual] + RawVariables[fI.p_tap_triennial] +
RawVariables[fI.p_tap_nine]) *
              (1-IntPower(1-RawVariables[fI.p_source_chng],fYears-3)) *
RawVariables[fI.p_adjust_cct_source_chng];
    fPA[2]:=(RawVariables[fI.p_tap_annual] + RawVariables[fI.p_tap_triennial] +
RawVariables[fI.p_tap_nine]) *
              (1-IntPower(1-RawVariables[fI.p_treat_change],fYears-3)) *
RawVariables[fI.p_adjust_cct_treat_chng];
    tp:=0;
    for i:=0 to 2 do tp:=tp+fPA[i];
    if tp>1 then begin
      //This is terrible.  But I guess it could happen.  Means no possible to choose
nothing.
      for i:=0 to 2 do fPA[i] := fPA[i]/tp;
    end;
    R:=iiRandom(True);
    tp:=0;
    Result:=0;
    for i:=0 to 2 do begin
      tp:=tp+fPA[i];
      if R<TP then begin
        Result:=i+1;
        break;
      end;
    end;
  end;
end;

function TCostingSteps.CCTInstallChoice(option: string): integer;
var tp, r: double;
    i: integer;
begin
  Result := 0;
  if option = 'Baseline' then
  begin
    fPI[0]:=(1-IntPower(1-RawVariables[fI.p_lead_ale],fYears-3)) *
RawVariables[fI.p_install_cct_sal];
    fPI[1]:=(1-IntPower(1-RawVariables[fI.p_copper_ale],fYears-3)) *
RawVariables[fI.p_install_cct_copper];
    fPI[2]:=(RawVariables[fI.p_tap_annual] + RawVariables[fI.p_tap_triennial] +
RawVariables[fI.p_tap_nine]) *
              (1-IntPower(1-RawVariables[fI.p_source_chng],fYears-3)) *
RawVariables[fI.p_install_cct_source_chng];
    fPI[3]:=(RawVariables[fI.p_tap_annual] + RawVariables[fI.p_tap_triennial] +
RawVariables[fI.p_tap_nine]) *
              (1-IntPower(1-RawVariables[fI.p_treat_change],fYears-3)) *
RawVariables[fI.p_install_cct_treat_chng];
    tp:=0;
    for i:=0 to 3 do tp:=tp+fPI[i];
    if tp>1 then begin
      //This is terrible.  But I guess it could happen.  Means no possible to choose
nothing.
      for i:=0 to 3 do fPI[i] := fPI[i]/tp;
    end;
    R:=iiRandom(True);
    tp:=0;
    Result:=0;
    for i:=0 to 3 do begin
      tp:=tp+fPI[i];
      if R<tp then begin
        Result:=i+1;
        break;
      end;
    end;
  end
  else if option = 'NDWAC' then
  begin
    fPI[0]:=(1-IntPower(1-RawVariables[fI.p_sal_one],15)) *
RawVariables[fI.p_install_cct_sal];
    fPI[1]:=(1-IntPower(1-RawVariables[fI.p_sal_two],6)) *
RawVariables[fI.p_install_cct_sal];
    fPI[2]:=(1-IntPower(1-RawVariables[fI.p_sal_three],11)) *
RawVariables[fI.p_install_cct_sal];
    fPI[3]:=(1-IntPower(1-RawVariables[fI.p_lead_agg],fYears-3)) *
RawVariables[fI.p_lead_agg_inst];
    fPI[4]:=(1-RawVariables[fI.p_nonagg]) * RawVariables[fI.p_copper_agg_inst];
    fPI[5]:=(1-IntPower(1-RawVariables[fI.p_source_chng],fYears-3)) *
RawVariables[fI.p_install_cct_source_chng];
    fPI[6]:=(1-IntPower(1-RawVariables[fI.p_treat_change],fYears-3)) *
RawVariables[fI.p_install_cct_treat_chng];
    tp:=0;
    for i:=0 to 6 do tp:=tp+fPI[i];
    if tp>1 then begin
      //This is terrible.  But I guess it could happen.  Means no possible to choose
nothing.
      for i:=0 to 6 do fPI[i] := fPI[i]/tp;
    end;
    R:=iiRandom(False);
    tp:=0;
    Result:=0;
    for i:=0 to 6 do begin
      tp:=tp+fPI[i];
      if R<tp then begin
        Result:=i+1;
        break;
      end;
    end;
```

```
 end
 else if option = 'Platinum' then
 begin
  fPI[0]:=(1-IntPower(1-RawVariables[fI.p_lead_ale_one],15)) * 
RawVariables[fI.p_install_cct_sal];
  fPI[1]:=(1-IntPower(1-RawVariables[fI.p_lead_ale_two],6)) * 
RawVariables[fI.p_install_cct_sal];
  fPI[2]:=(1-IntPower(1-RawVariables[fI.p_lead_ale_three],11)) * 
RawVariables[fI.p_install_cct_sal];
  fPI[3]:=(1-RawVariables[fI.p_nonagg]) * RawVariables[fI.p_copper_agg_inst];
  tp:=0;
  for i:=0 to 3 do tp:=tp+fPI[i];
  if tp>1 then begin
   //This is terrible.  But I guess it could happen.  Means no possible to choose 
nothing.
   for i:=0 to 3 do fPI[i] := fPI[i]/tp;
  end;
  R:=iiRandom(False);
  tp:=0;
  Result:=0;
  for i:=0 to 3 do begin
   tp:=tp+fPI[i];
   if R<tp then begin
    Result:=i+1;
    break;
   end;
  end;
 end
 else if option = 'Green' then
 begin
  fPI[0]:=(1-IntPower(1-RawVariables[fI.p_lead_ale_one],3)) * 
RawVariables[fI.p_green_cct_install];
  fPI[1]:=(1-IntPower(1-RawVariables[fI.p_lead_ale_two],18)) * 
RawVariables[fI.p_green_cct_install];
  fPI[2]:=(1-IntPower(1-RawVariables[fI.p_lead_ale_three],11)) * 
RawVariables[fI.p_green_cct_install];
  fPI[3]:=(1-RawVariables[fI.p_nonagg]) * RawVariables[fI.p_copper_agg_inst];
  fPI[4]:=(1-IntPower(1-RawVariables[fI.p_source_chng],fYears-3)) * 
RawVariables[fI.p_install_cct_source_chng];
  fPI[5]:=(1-IntPower(1-RawVariables[fI.p_treat_change],fYears-3)) * 
RawVariables[fI.p_install_cct_treat_chng];
  tp:=0;
  for i:=0 to 5 do tp:=tp+fPI[i];
  if tp>1 then begin
   //This is terrible.  But I guess it could happen.  Means no possible to choose 
nothing.
   for i:=0 to 5 do fPI[i] := fPI[i]/tp;
  end;
```

```
    R:=iiRandom(False);
    tp:=0;
    Result:=0;
    for i:=0 to 5 do begin
      tp:=tp+fPI[i];
      if R<tp then begin
        Result:=i+1;
        break;
      end;
    end;
  end
  else if option = 'BabyBlue' then
  begin
{$IFDEF AnalysisYears_35}
    fPI[0]:=(1-IntPower(1-RawVariables[fI.p_lead_ale_one],15)) *
RawVariables[fI.p_install_cct_sal];
    fPI[1]:=(1-IntPower(1-RawVariables[fI.p_lead_ale_two],6)) *
RawVariables[fI.p_install_cct_sal];
    fPI[2]:=(1-IntPower(1-RawVariables[fI.p_lead_ale_three],11)) *
RawVariables[fI.p_install_cct_sal];
    fPI[3]:=(1-IntPower(1-RawVariables[fI.p_copper_ale],fYears-3)) *
RawVariables[fI.p_install_cct_copper];
    fPI[4]:=(1-IntPower(1-RawVariables[fI.p_source_chng],fYears-3)) *
RawVariables[fI.p_install_cct_source_chng];
    fPI[5]:=(1-IntPower(1-RawVariables[fI.p_treat_change],fYears-3)) *
RawVariables[fI.p_install_cct_treat_chng];
    tp:=0;
    for i:=0 to 5 do tp:=tp+fPI[i];
    if tp>1 then begin
      //This is terrible.  But I guess it could happen.  Means no possible to choose
nothing.
      for i:=0 to 5 do fPI[i] := fPI[i]/tp;
    end;
    R:=iiRandom(False);
    tp:=0;
    Result:=0;
    for i:=0 to 5 do begin
      tp:=tp+fPI[i];
      if R<tp then begin
        Result:=i+1;
        break;
      end;
    end;
  end;
{$ENDIF}
{$IFDEF AnalysisYears_20}
    fPI[0]:=(1-IntPower(1-RawVariables[fI.p_lead_ale_one],15)) *
RawVariables[fI.p_install_cct_sal];
```

```
    fPI[1]:=(1-IntPower(1-RawVariables[fI.p_lead_ale_two],2)) *
RawVariables[fI.p_install_cct_sal];
    fPI[2]:=(1-IntPower(1-RawVariables[fI.p_copper_ale],fYears-3)) *
RawVariables[fI.p_install_cct_copper];
    fPI[3]:=(1-IntPower(1-RawVariables[fI.p_source_chng],fYears-3)) *
RawVariables[fI.p_install_cct_source_chng];
    fPI[4]:=(1-IntPower(1-RawVariables[fI.p_treat_change],fYears-3)) *
RawVariables[fI.p_install_cct_treat_chng];
    tp:=0;
    for i:=0 to 4 do tp:=tp+fPI[i];
    if tp>1 then begin
      for i:=0 to 4 do fPI[i] := fPI[i]/tp;
    end;
    R:=iiRandom(False);
    tp:=0;
    Result:=0;
    for i:=0 to 4 do begin
      tp:=tp+fPI[i];
      if R<tp then begin
        Result:=i+1;
        break;
      end;
    end;
  end;
{$ENDIF}
{$IFDEF AnalysisYears_10}
    fPI[0]:=(1-IntPower(1-RawVariables[fI.p_lead_ale_one],7)) *
RawVariables[fI.p_install_cct_sal];
    fPI[1]:=(1-IntPower(1-RawVariables[fI.p_copper_ale],fYears-3)) *
RawVariables[fI.p_install_cct_copper];
    fPI[2]:=(1-IntPower(1-RawVariables[fI.p_source_chng],fYears-3)) *
RawVariables[fI.p_install_cct_source_chng];
    fPI[3]:=(1-IntPower(1-RawVariables[fI.p_treat_change],fYears-3)) *
RawVariables[fI.p_install_cct_treat_chng];
    tp:=0;
    for i:=0 to 3 do tp:=tp+fPI[i];
    if tp>1 then begin
      for i:=0 to 3 do fPI[i] := fPI[i]/tp;
    end;
    R:=iiRandom(False);
    tp:=0;
    Result:=0;
    for i:=0 to 3 do begin
      tp:=tp+fPI[i];
      if R<tp then begin
        Result:=i+1;
        break;
      end;
    end;
  end;
{$ENDIF}
end;

function TCostingSteps.CheckAggAgreement(T: TCostingSteps): boolean;
var i : integer;
begin
  Result:=True;

  if high(AggInputs2)<>high(T.AggInputs2) then begin
    Result:=False;
    exit;
  end;
  for i:=0 to high(AggInputs2) do begin
    if CompareText(AggInputs2[i],T.AggInputs2[i])<>0 then begin
      Result:=False;
      exit;
    end;
  end;

  if high(AggInputs1)<>high(T.AggInputs1) then begin
    Result:=False;
    exit;
  end;
  for i:=0 to high(AggInputs1) do begin
    if CompareText(AggInputs1[i],T.AggInputs1[i])<>0 then begin
      Result:=False;
      exit;
    end;
  end;
end;

constructor TCostingSteps.Create( aCostVars : TCostVars; Filename: string; Filter:
string; aYears,aYearsOutput : integer;  IsBaseline : boolean);
var T : TCostVar;
    y,f,tt,ix : integer;
begin
  if IsBaseline then
    CC := TLSRCompiledCostBaseline.Create
  else
    CC := TLSRCompiledCostOption.Create;
  fIsBaseline:=IsBaseline;
  fErrors:=TStringList.Create;
  fParser:=TParser.Create(nil);
  fParser.Cached := False;
  fillchar(fI,SizeOf(fI),-1);
  fCostVars:=aCostVars;
```

```
CostSteps:=TObjectList<TCostingStep>.create();
DiscRate:=0.03;
setlength(Variables,fCostVars.Count);
setlength(RawVariables,fCostVars.Count);
//set from Word Doc CCT install/adjust
setlength(fPA,9);
setlength(fPI,7);

fillchar(GBin,SizeOf(GBin),0);
setlength(GMOveBin,aYears+1);
setlength(GMOveBinMicro,aYears+1);
for y:=0 to high(GMoveBin) do begin
  for f:=low(GMoveBin[y]) to high(GMoveBin[y]) do begin
    GMoveBin[y,f]:=0;
    for tt:=low(GMoveBinMicro[y,f]) to high(GMoveBinMicro[y,f]) do begin
      GMoveBinMicro[y,f,tt]:=0;
    end;
  end;
end;

slVariables := TStringList.Create;
slCosts := TStringList.Create;
slCalcs := TStringList.Create;
slCCTCosts := TStringList.Create;

fI.ep:=-1;
fI.Pws_Cct := -1;
fI.P_Fail := -1;
fI.num_lsl_replace := -1;
fI.Meet_Lslr_Goal := -1;
fI.Pws_first_ale := -1;
fYears:=aYears;
fYearsOutput:=aYearsOutput;

CheckPWS := '';

for ix:=0 to fCostVars.Count-1 do begin
  //use this to be consistent with the other accesses to the CostVars
  //.ToArray might not come out in "v in values" order.  Maybe.
  t:=fCostVars.DirectArray[ix];
  CC._SetVarPointer(t.fID,@Variables[ix]);

  if t.fID='numb_ep' then fI.ep:=ix else
  if t.fID='pws_cct' then fI.Pws_Cct:=ix else
  if t.fID='p_fail' then fI.P_Fail:=ix else
  if t.fID='num_lsl_replace' then fI.num_lsl_replace:=ix else
  if t.fID='meet_lslr_goal' then fI.Meet_Lslr_Goal:=ix else
  if t.fID='pws_first_ale' then fI.Pws_first_ale:=ix else
  if t.fID='pws_sw' then fI.Pws_sw:=ix else
  if t.fID='pws_gw' then fI.Pws_gw:=ix else
  if t.fID='pws_pop' then fI.Pws_pop:=ix else
  if t.fID='numb_hh' then fI.Numb_hh:=ix else

  //and now a ton of indexes from the custom calcs...
  if t.fID='p_sal_one' then fI.p_sal_one:=ix else
  if t.fID='p_sal_two' then fI.p_sal_two:=ix else
  if t.fID='p_sal_three' then fI.p_sal_three:=ix else
  if t.fID='p_adjust_cct_sal' then fI.p_adjust_cct_sal:=ix else
  if t.fID='p_wqp_chng' then fI.p_wqp_chng:=ix else
  if t.fID='p_wqp_chng_adj' then fI.p_wqp_chng_adj:=ix else
  if t.fID='p_nonagg' then fI.p_nonagg:=ix else
  if t.fID='p_copper_agg_adj' then fI.p_copper_agg_adj:=ix else
  if t.fID='p_source_chng' then fI.p_source_chng:=ix else
  if t.fID='p_source_sig' then fI.p_source_sig:=ix else
  if t.fID='p_adjust_cct_source_chng' then fI.p_adjust_cct_source_chng:=ix else
  if t.fID='p_cct_guide_apply' then fI.p_cct_guide_apply:=ix else
  if t.fID='p_cct_guid_chng' then fI.p_cct_guid_chng:=ix else
  if t.fID='p_treat_change' then fI.p_treat_change:=ix else
  if t.fID='p_adjust_cct_treat_chng' then fI.p_adjust_cct_treat_chng:=ix else
  if t.fID='p_install_cct_sal' then fI.p_install_cct_sal:=ix else
  if t.fID='p_lead_agg' then fI.p_lead_agg:=ix else
  if t.fID='p_lead_agg_inst' then fI.p_lead_agg_inst:=ix else
  if t.fID='p_copper_agg_inst' then fI.p_copper_agg_inst:=ix else
  if t.fID='p_install_cct_source_chng' then fI.p_install_cct_source_chng:=ix else
  if t.fID='p_install_cct_treat_chng' then fI.p_install_cct_treat_chng:=ix else

  //indexes for lslr activities
  if t.fID='p_lsl' then fI.p_lsl:=ix else
  if t.fID='perc_lsl' then fI.perc_lsl:=ix else
  if t.fID='pp_lsl_replaced_one' then fI.pp_lsl_replaced_one:=ix else
  if t.fID='pp_lsl_replaced_two' then fI.pp_lsl_replaced_two:=ix else
  if t.fID='pp_lsl_replaced_three' then fI.pp_lsl_replaced_three:=ix else
  if t.fID='lslr_goal_one' then fI.lslr_goal_one:=ix else
  if t.fID='lslr_goal_two' then fI.lslr_goal_two:=ix else
  if t.fID='lslr_goal_three' then fI.lslr_goal_three:=ix else

  if t.fID='p_inventory' then fI.p_inventory:=ix else
  if t.fID='p_tap_nine' then fI.p_tap_nine:=ix else
  if t.fID='p_tap_annual' then fI.p_tap_annual:=ix else
  if t.fID='p_tap_triennial' then fI.p_tap_triennial:=ix else
  if t.fID='p_spec_req' then fI.p_spec_req:=ix else
  if t.fID='p_wqp_annual' then fI.p_wqp_annual:=ix else
  if t.fID='p_wqp_triennial' then fI.p_wqp_triennial:=ix else
  if t.fID='p_wqp_six_red' then fI.p_wqp_six_red:=ix else
  if t.fID='p_b3' then fI.p_b3:=ix else
```

```
    if t.fID='p_copper_ale' then fI.p_copper_ale:=ix else
    if t.fID='p_adjust_cct_copper' then fI.p_adjust_cct_copper:=ix else
    if t.fID='p_lead_ale' then fI.p_lead_ale:=ix else
    if t.fID='p_install_cct_copper' then fI.p_install_cct_copper:=ix else

    if t.fID='fail_nm_1_2_3' then fI.fail_nm_1_2_3:=ix else
    if t.fID='fail_nm_4' then fI.fail_nm_4:=ix else
    if t.fID='fail_nm_5' then fI.fail_nm_5:=ix else
    if t.fID='fail_nm_6' then fI.fail_nm_6:=ix else
    if t.fID='fail_nm_7' then fI.fail_nm_7:=ix else
    if t.fID='fail_nm_8_9_10' then fI.fail_nm_8_9_10:=ix else
    if t.fID='hh_remain_lsl' then fI.hh_remain_lsl:=ix else

    if t.fID='b_wqp_chng_adj' then fI.b_wqp_chng_adj:=ix else
    if t.fID='b_copper_agg_adj' then fI.b_copper_agg_adj:=ix else
    if t.fID='b_adjust_cct_source_chng' then fI.b_adjust_cct_source_chng:=ix else
    if t.fID='b_cct_guid_chng' then fI.b_cct_guid_chng:=ix else
    if t.fID='b_adjust_cct_treat_chng' then fI.b_adjust_cct_treat_chng:=ix else
    if t.fID='b_lead_agg_inst' then fI.b_lead_agg_inst:=ix else
    if t.fID='b_copper_agg_inst' then fI.b_copper_agg_inst:=ix else
    if t.fID='b_install_cct_source_chng' then fI.b_install_cct_source_chng:=ix else
    if t.fID='b_install_cct_treat_chng' then fI.b_install_cct_treat_chng:=ix else

    if t.fID='b_adjust_cct_copper' then fI.b_adjust_cct_copper:=ix else
    if t.fID='b_install_cct_lead_ale' then fI.b_install_cct_lead_ale:=ix else
    if t.fID='b_install_cct_copper_ale' then fI.b_install_cct_copper_ale:=ix else
    if t.fID='b_install_cct_treat' then fI.b_install_cct_treat:=ix else

    if t.fID='cct_existing_cost' then fI.cct_existing_cost:=ix else
    if t.fID='cct_modify_cost' then fI.cct_modify_cost:=ix else
    if t.fID='cct_install_cost' then fI.cct_install_cost:=ix else
    if t.fID='cct_findfix_cost' then fI.cct_findfix_cost:=ix else

    if t.fID='cct_modify_cost_umra' then fI.cct_modify_cost_umra:=ix else
    if t.fID='cct_install_cost_umra' then fI.cct_install_cost_umra:=ix else
    if t.fID='cct_findfix_cost_umra' then fI.cct_findfix_cost_umra:=ix else
    if t.fID='cct_modify_cost_umra_om' then fI.cct_modify_cost_umra_om:=ix else
    if t.fID='cct_install_cost_umra_om' then fI.cct_install_cost_umra_om:=ix else
    if t.fID='cct_findfix_cost_umra_om' then fI.cct_findfix_cost_umra_om:=ix else

    if t.fID='cct_modify_cost_p' then fI.cct_modify_cost_p:=ix else
    if t.fID='cct_install_cost_p' then fI.cct_install_cost_p:=ix else
    if t.fID='cct_findfix_cost_p' then fI.cct_findfix_cost_p:=ix else


    if t.fID='pbaseph' then fI.pbaseph:=ix else
    if t.fID='pbasepo4' then fI.pbasepo4:=ix else
    if t.fID='pbasephpo4' then fI.pbasephpo4:=ix else
    if t.fID='baselinepo4dose' then fI.baselinepo4dose:=ix else
    if t.fID='baselineph_w' then fI.baselineph_w :=ix else
    if t.fID='baselineph_wo' then fI.baselineph_wo:=ix else
    if t.fID='targetph' then fI.targetph:=ix else
    if t.fID='targetpo4' then fI.targetpo4:=ix else

    if t.fID='hh_consumption' then fI.hh_consumption:=ix else

    if t.fID='b_adjust_cct_sal_p1' then fI.b_adjust_cct_sal_p1:=ix else
    if t.fID='b_adjust_cct_sal_p2' then fI.b_adjust_cct_sal_p2:=ix else
    if t.fID='b_adjust_cct_sal_p3' then fI.b_adjust_cct_sal_p3:=ix else
    if t.fID='b_install_cct_sal_p1' then fI.b_install_cct_sal_p1:=ix else
    if t.fID='b_install_cct_sal_p2' then fI.b_install_cct_sal_p2:=ix else
    if t.fID='b_install_cct_sal_p3' then fI.b_install_cct_sal_p3:=ix else

    if t.fID='p_lead_ale_one' then fI.p_lead_ale_one:=ix else
    if t.fID='p_lead_ale_two' then fI.p_lead_ale_two:=ix else
    if t.fID='p_lead_ale_three' then fI.p_lead_ale_three:=ix else

    if t.fID='b_adjust_cct_lead_plat_1' then fI.b_adjust_cct_lead_plat_1:=ix else
    if t.fID='b_adjust_cct_lead_plat_2' then fI.b_adjust_cct_lead_plat_2:=ix else
    if t.fID='b_adjust_cct_lead_plat_3' then fI.b_adjust_cct_lead_plat_3:=ix else
    if t.fID='b_copper_agg_adj_plat' then fI.b_copper_agg_adj_plat:=ix else
    if t.fID='b_adjust_cct_source_chng_plat' then
fI.b_adjust_cct_source_chng_plat:=ix else
    if t.fID='b_adjust_cct_treat_chng_plat' then fI.b_adjust_cct_treat_chng_plat:=ix
else
    if t.fID='b_install_cct_lead_plat_1' then fI.b_install_cct_lead_plat_1:=ix else
    if t.fID='b_install_cct_lead_plat_2' then fI.b_install_cct_lead_plat_2:=ix else
    if t.fID='b_install_cct_lead_plat_3' then fI.b_install_cct_lead_plat_3:=ix else
    if t.fID='b_copper_agg_inst_plat' then fI.b_copper_agg_inst_plat:=ix else

    if t.fID='pp_lslr_partial' then fI.pp_lslr_partial:=ix else
    if t.fID='pp_lslr_paper' then fI.pp_lslr_paper:=ix else

    if t.fID='p_green_cct_adjust' then fI.p_green_cct_adjust:=ix else
    if t.fID='p_green_cct_install' then fI.p_green_cct_install:=ix else
    if t.fID='b_modify_cct_50_lsl' then fI.b_modify_cct_50_lsl:=ix else
    if t.fID='b_install_cct_50_lsl' then fI.b_install_cct_50_lsl:=ix else
    if t.fID='b_adjust_cct_lead_green_1' then fI.b_adjust_cct_lead_green_1:=ix else
    if t.fID='b_adjust_cct_lead_green_2' then fI.b_adjust_cct_lead_green_2:=ix else
    if t.fID='b_adjust_cct_lead_green_3' then fI.b_adjust_cct_lead_green_3:=ix else
    if t.fID='b_install_cct_lead_green_1' then fI.b_install_cct_lead_green_1:=ix
else
    if t.fID='b_install_cct_lead_green_2' then fI.b_install_cct_lead_green_2:=ix
else
    if t.fID='b_install_cct_lead_green_3' then fI.b_install_cct_lead_green_3:=ix
else
```

```
    if t.fID='adjust_cct' then fI.adjust_cct:=ix else
    if t.fID='install_cct' then fI.install_cct:=ix else
    if t.fID='b_copper_agg_adj_green' then fI.b_copper_agg_adj_green:=ix else
    if t.fID='b_copper_agg_inst_green' then fI.b_copper_agg_inst_green:=ix else
    if t.fID='b_install_cct_source_chng_green' then 
fI.b_install_cct_source_chng_green:=ix else
    if t.fID='b_adjust_cct_source_chng_green' then 
fI.b_adjust_cct_source_chng_green:=ix else
    if t.fID='b_install_cct_treat_chng_green' then 
fI.b_install_cct_treat_chng_green:=ix else
    if t.fID='b_adjust_cct_treat_chng_green' then 
fI.b_adjust_cct_treat_chng_green:=ix else
    if t.fID='num_hh_per_connect' then fI.num_hh_per_connect:=ix else
    if t.fID='pws_fail' then fI.pws_fail:=ix else
    if t.fID='lslr_green_rate' then fI.lslr_green_rate:=ix else
    if t.fID='b_cct_guid_chng_five' then fI.b_cct_guid_chng_five:=ix else
    if t.fID='b_install_cct_lead_ale_one' then fI.b_install_cct_lead_ale_one:=ix 
else
    if t.fID='b_install_cct_lead_ale_two' then fI.b_install_cct_lead_ale_two:=ix 
else
    if t.fID='b_install_cct_lead_ale_three' then fI.b_install_cct_lead_ale_three:=ix 
else
    if t.fID='p_sanit_surv_chng' then fI.p_sanit_surv_chng:=ix else
    if t.fID='b_cct_sanitary_survey' then fI.b_cct_sanitary_survey:=ix else
    if t.fID='p_ss_cct_guide_apply' then fI.p_ss_cct_guide_apply:=ix else
    if t.fID='pp_lcr_test' then fI.pp_lcr_test:=ix else
    if t.fID='pp_lcr_test_yes' then fI.pp_lcr_test_yes:=ix else
    if t.fID='num_lsl_paper' then fI.num_lsl_paper:=ix else

    if t.fID='dist_lead_base_bin1' then fI.dist_lead_base_bin1:=ix else
    if t.fID='dist_lead_base_bin2' then fI.dist_lead_base_bin2:=ix else
    if t.fID='dist_lead_base_bin3' then fI.dist_lead_base_bin3:=ix else
    if t.fID='bin_distr' then fI.bin_distr:=ix else
    if t.fID='p_bin3_nonzero' then fI.p_bin3_nonzero:=ix else

    if t.fID='pp_lsl_replaced_vol_goal' then fI.pp_lsl_replaced_vol_goal:=ix else
    if t.fID='pp_lsl_replaced_vol_pct' then fI.pp_lsl_replaced_vol_pct:=ix else
    if t.fID='rnd_p90_error' then fI.rnd_p90_error:=ix else
    if t.fID='b_modify_cct' then fI.b_modify_cct:=ix else
    if t.fID='b_install_cct' then fI.b_install_cct:=ix else
    if t.fID='b_modify_cct_mc' then fI.b_modify_cct_mc:=ix else
    if t.fID='b_install_cct_mc' then fI.b_install_cct_mc:=ix else
    if t.fID='b_install_pou' then fI.b_install_pou:=ix else
    if t.fID='system_pou' then fI.system_pou:=ix else
    if t.fID='numb_reduced_tap' then fI.numb_reduced_tap:=ix else
    if t.fID='numb_samp_customer' then fI.numb_samp_customer:=ix else
    if t.fID='pp_above_al_bin_one' then fI.pp_above_al_bin_one:=ix else
    if t.fID='pp_above_al_bin_two' then fI.pp_above_al_bin_two:=ix else
    if t.fID='pp_above_al_bin_three' then fI.pp_above_al_bin_three:=ix else
    if t.fID='b_findfix' then fI.b_findfix:=ix else
    if t.fID='b_lslr_study' then fI.b_lslr_study:=ix else
    if t.fID='b_pou_study' then fI.b_pou_study:=ix else
    if t.fID='b_lslr_mand' then fI.b_lslr_mand:=ix else
    if t.fID='b_lslr_vol' then fI.b_lslr_vol:=ix else
    if t.fID='b_lslr_requested' then fI.b_lslr_requested:=ix else
    if t.fID='school_1a' then fI.school_1a:=ix else
    if t.fID='school_1b' then fI.school_1b:=ix else
    if t.fID='school_3a' then fI.school_3a:=ix else
    if t.fID='school_3b' then fI.school_3b:=ix else
    if t.fID='school_3c' then fI.school_3c:=ix else
    if t.fID='school_3d' then fI.school_3d:=ix else
    if t.fID='school_5a' then fI.school_5a:=ix else
    if t.fID='school_5b' then fI.school_5b:=ix else

    if t.fID='post_cct_p90_bin1' then fI.post_cct_p90_bin1:=ix else
    if t.fID='post_cct_p90_bin2' then fI.post_cct_p90_bin2:=ix else
    if t.fID='post_ff_p90_bin1' then fI.post_ff_p90_bin1:=ix else
    if t.fID='post_ff_p90_bin2' then fI.post_ff_p90_bin2:=ix else

    if t.fID='p_cct_study' then fI.p_cct_study:=ix else
    if t.fID='b_cct_study_rec_install' then fI.b_cct_study_rec_install:=ix else
    if t.fID='b_cct_study_install' then fI.b_cct_study_install:=ix else
    if t.fID='b_state_cct_treatment_install' then
fI.b_state_cct_treatment_install:=ix else
    if t.fID='b_cct_study_rec_mod' then fI.b_cct_study_rec_mod:=ix else
    if t.fID='b_cct_study_mod' then fI.b_cct_study_mod:=ix else
    if t.fID='b_state_cct_treatment_mod' then fI.b_state_cct_treatment_mod:=ix else
    if t.fID='fail_nm1' then fI.fail_nm1:=ix else
    if t.fID='fail_nm2' then fI.fail_nm2:=ix else

    if t.fID='num_vol_leadtap_samples_per_k' then
fI.num_vol_leadtap_samples_per_k:=ix else
    if t.fID='p_vol_leadtap_prog' then fI.p_vol_leadtap_prog:=ix else
    if t.fID='hrs_act_wqp_op' then fI.hrs_act_wqp_op:=ix else
    if t.fID='cost_act_wqp' then fI.cost_act_wqp:=ix else
    if t.fID='rate_op' then fI.rate_op:=ix else
    if t.fID='b_cct_study_rec_mod_tl' then fI.b_cct_study_rec_mod_tl:=ix else
    if t.fID='b_cct_study_mod_tl' then fI.b_cct_study_mod_tl:=ix else
    if t.fID='b_modify_cct_tl' then fI.b_modify_cct_tl:=ix else
    if t.fID='b_state_cct_treatment_mod_tl' then fI.b_state_cct_treatment_mod_tl:=ix
else
    if t.fID='b_cct_study_rec_mod_al' then fI.b_cct_study_rec_mod_al:=ix else
    if t.fID='b_cct_study_mod_al' then fI.b_cct_study_mod_al:=ix else
    if t.fID='b_modify_cct_al' then fI.b_modify_cct_al:=ix else
    if t.fID='b_state_cct_treatment_mod_al' then fI.b_state_cct_treatment_mod_al:=ix
else
```

```
    if t.fID='numb_wqp_sites_added' then fI.numb_wqp_sites_added:=ix else
    if t.fID='numb_wqp_sites_added_prev' then fI.numb_wqp_sites_added_prev:=ix else
    if t.fID='pp_overlap_find_fix' then fI.pp_overlap_find_fix:=ix else
    if t.fID='numb_reduced_wqp' then fI.numb_reduced_wqp:=ix else
    if t.fID='numb_enhance_wqp' then fI.numb_enhance_wqp:=ix else
    if t.fID='pp_cust_init_lslr' then fI.pp_cust_init_lslr:=ix else
    if t.fID='num_lsl_requested' then fI.num_lsl_requested:=ix else
    if t.fID='annual_pou_cost_hh' then fI.annual_pou_cost_hh:=ix else
    if t.fID='numb_second_schools_pub' then fI.numb_second_schools_pub:=ix else
    if t.fID='numb_elem_schools_pub' then fI.numb_elem_schools_pub:=ix else
    if t.fID='numb_second_schools_priv' then fI.numb_second_schools_priv:=ix else
    if t.fID='numb_elem_schools_priv' then fI.numb_elem_schools_priv:=ix else
    if t.fID='numb_daycares' then fI.numb_daycares:=ix else
    if t.fID='p_grandfather_opt_pub' then fI.p_grandfather_opt_pub:=ix else
    if t.fID='p_grandfather_opt_priv' then fI.p_grandfather_opt_priv:=ix else
    if t.fID='p_grandfather_mand_pub' then fI.p_grandfather_mand_pub:=ix else
    if t.fID='p_grandfather_mand_priv' then fI.p_grandfather_mand_priv:=ix else
    if t.fID='p_grandfather_opt_child' then fI.p_grandfather_opt_child:=ix else
    if t.fID='p_grandfather_mand_child' then fI.p_grandfather_mand_child:=ix else
    if t.fID='b_state_one' then fI.b_state_one:=ix else
    if t.fID='b_state_two' then fI.b_state_two:=ix else
    if t.fID='num_paper_remain' then fI.num_paper_remain:=ix else
    if t.fID='num_lsl_remain' then fI.num_lsl_remain:=ix
    ;

    fParser.AddVariable(T.fID,Variables[ix]);
  end;

// The following values are also in PWSReplicator.pas TPWSReplicator.Create

  // arrPBinOptionBaseline[CCT, LSL, Probs]
  arrPBinBaseline[0,0,0] := 0.023;
  arrPBinBaseline[0,0,1] := 0.028;
  arrPBinBaseline[0,0,2] := 0.948;
  arrPBinBaseline[1,0,0] := 0.033;
  arrPBinBaseline[1,0,1] := 0.020;
  arrPBinBaseline[1,0,2] := 0.947;
  arrPBinBaseline[0,1,0] := 0.027;
  arrPBinBaseline[0,1,1] := 0.060;
  arrPBinBaseline[0,1,2] := 0.913;
  arrPBinBaseline[1,1,0] := 0.014;
  arrPBinBaseline[1,1,1] := 0.081;
  arrPBinBaseline[1,1,2] := 0.905;

  // arrPBinOption[CCT, LSL, Probs]
  arrPBinOption[0,0,0] := 0.023;
  arrPBinOption[0,0,1] := 0.028;
  arrPBinOption[0,0,2] := 0.948;
  arrPBinOption[1,0,0] := 0.033;
  arrPBinOption[1,0,1] := 0.020;
  arrPBinOption[1,0,2] := 0.947;
  arrPBinOption[0,1,0] := 0.07;
  arrPBinOption[0,1,1] := 0.06;
  arrPBinOption[0,1,2] := 0.87;
  arrPBinOption[1,1,0] := 0.07;
  arrPBinOption[1,1,1] := 0.15;
  arrPBinOption[1,1,2] := 0.78;

  // arrPBinOption5L[CCT, LSL, Probs]
  arrPBinOption5L[0,0,0] := 0.023;
  arrPBinOption5L[0,0,1] := 0.028;
  arrPBinOption5L[0,0,2] := 0.948;
  arrPBinOption5L[1,0,0] := 0.033;
  arrPBinOption5L[1,0,1] := 0.020;
  arrPBinOption5L[1,0,2] := 0.947;
  arrPBinOption5L[0,1,0] := 0.13;
  arrPBinOption5L[0,1,1] := 0.15;
  arrPBinOption5L[0,1,2] := 0.72;
  arrPBinOption5L[1,1,0] := 0.22;
  arrPBinOption5L[1,1,1] := 0.19;
  arrPBinOption5L[1,1,2] := 0.59;

  RowNo:=0;
  ReadSteps(Filename,Filter);

  fillchar(Values2,SizeOf(Values2),0);
  fillchar(Values1,SizeOf(Values1),0);
  fillchar(Values2p,SizeOf(Values2p),0);
  fillchar(Values1p,SizeOf(Values1p),0);
  fillchar(Values2Y,SizeOf(Values2Y),0);

end;

destructor TCostingSteps.Destroy;
begin
  SetLength(Variables, 0);
  SetLength(RawVariables, 0);
  CC.Free;

  slVariables.Free;
  slCosts.Free;
  slCalcs.Free;
  slCCTCosts.Free;

  fErrors.Free;
  CostSteps.Free;
  fParser.Free;
  inherited;
end;

procedure TCostingSteps.DetermineSystemPValues(const aSz, aSrc, aLSL,
  aCCT, aType : integer; const SetProbsTo01: boolean; const PWSId: string;
GetBaseCurValue : boolean=false);
begin
  if PWSId.Substring(1,9) = 'OH2504412' then
    PWS_p_values.p_lsl := 1
  else
    PWS_p_values.p_lsl := Round(fcostvars.Calculate_p_lsl(aSz, aSrc, aLSL, aCCT,
aType, SetProbsTo01,GetBaseCurValue));
  fCostVars.Calculate_pws_p_values(aSz, aSrc, PWS_p_values.p_lsl, aCCT, aType,
SetProbsTo01, PWS_p_values,GetBaseCurValue);
end;

procedure TCostingSteps.ReadSteps(Filename, Filter: string);
var
  Xls: TExcelFile;
  r, ci: integer;
  sl2,sl1,sl3 : TStringList;
  C : TCostingStep;
  IsBaseline: boolean;
  cn : string;
begin
  if AnsiContainsStr(Filename, 'Baseline') then
    IsBaseline := true
  else
    IsBaseline := false;

  Xls := TXlsFile.Create(Filename, False);
  Xls.ActiveSheetByName := 'Steps';
  {
    CWS_Costing_Steps_logic.xlsx
    A 1 Cost Number
    B 2 Cost Name
    C 3 Cost Description
    D 4 Probability cost applies to PWS or state (blank=1)
    E 5 Total Cost per Event (expression)
    F 6 Hours (Reporting)
    G 7 Labor (Reporting)
    H 8 O&M (Reporting)
    I 9 Domain
    J 10 Output Cost Group
    K 11 Frequency
    P 16 ICR Variable?
    Q 17 Intermediate Agglomerator
    R 18 Agglomerator
    S 19 Year
    U 21 Agglomerator ICR
    V 22 Include Cost
    W 23 VLS EP Level Analysis ?
    X 24 Bin 1
    Y 25 Bin 2
    Z 26 Bin 3
    AA 27 Bin 4
  }

  // sl2: Agglomerator values (18)
  sl2 := TStringList.Create;
  sl2.Sorted := true;
  sl2.Duplicates := dupIgnore;
  sl1 := TStringList.Create;
  sl1.Sorted := true;
  sl1.Duplicates := dupIgnore;

  // sl3 Agglomerator ICR values (21)
  sl3 := TStringList.Create;
  sl3.Sorted := true;
  sl3.Duplicates := dupIgnore;

  ci := -1;
  for r := 2 to Xls.RowCount do
  begin
    if (Xls.GetStringFromCell(r, 1) <> '') and
       (Xls.GetStringFromCell(r, 2) <> '') then
    begin
      cn:=Xls.GetStringFromCell(r, 2);
      if cn[1]='#' then continue;
      inc(ci);
      if Filter = '' then
      begin
        Add(IsBaseline, Xls.GetStringFromCell(r, 1),
            Xls.GetStringFromCell(r, 2),
            Xls.GetStringFromCell(r, 4),
            Xls.GetStringFromCell(r, 5),
            Xls.GetStringFromCell(r, 6),
            Xls.GetStringFromCell(r, 7),
            Xls.GetStringFromCell(r, 8),
            Xls.GetStringFromCell(r, 9),
            Xls.GetStringFromCell(r, 10),
            Xls.GetStringFromCell(r, 11),
            Xls.GetStringFromCell(r, 17),
            Xls.GetStringFromCell(r, 18),
            Xls.GetStringFromCell(r, 19),
            Xls.GetStringFromCell(r, 16),
            Xls.GetStringFromCell(r, 21),
            Xls.GetStringFromCell(r, 22),
            Xls.GetStringFromCell(r, 23),
            Xls.GetStringFromCell(r, 24),
            Xls.GetStringFromCell(r, 25),
            Xls.GetStringFromCell(r, 26),
            Xls.GetStringFromCell(r, 27),
            ci
            );
        // agglomerator metrics
        if Xls.GetStringFromCell(r, 18) <> '' then
        begin
          sl2.Add(Trim(Xls.GetStringFromCell(r, 18)));
          sl2.Add(Trim(Xls.GetStringFromCell(r, 18)) + '_hours');
          sl2.Add(Trim(Xls.GetStringFromCell(r, 18)) + '_labor');
          sl2.Add(Trim(Xls.GetStringFromCell(r, 18)) + '_om');
        end;
        sl1.Add(Trim(Xls.GetStringFromCell(r, 17)));
        sl1.Add(Trim(Xls.GetStringFromCell(r, 17)) + '_hours');
        sl1.Add(Trim(Xls.GetStringFromCell(r, 17)) + '_labor');
        sl1.Add(Trim(Xls.GetStringFromCell(r, 17)) + '_om');

        // Agglomerator ICR
        if Xls.GetStringFromCell(r, 21) <> '' then
        begin
          sl3.Add('ICR_C ' + Trim(Xls.GetStringFromCell(r, 21)) + '_1');
          sl3.Add('ICR_C ' + Trim(Xls.GetStringFromCell(r, 21)) + '_2');
          sl3.Add('ICR_C ' + Trim(Xls.GetStringFromCell(r, 21)) + '_3');
          sl3.Add('ICR_C ' + Trim(Xls.GetStringFromCell(r, 21)) + '_4');
          sl3.Add('ICR_C ' + Trim(Xls.GetStringFromCell(r, 21)) + '_10');
          sl3.Add('ICR_H ' + Trim(Xls.GetStringFromCell(r, 21)) + '_1');
          sl3.Add('ICR_H ' + Trim(Xls.GetStringFromCell(r, 21)) + '_2');
          sl3.Add('ICR_H ' + Trim(Xls.GetStringFromCell(r, 21)) + '_3');
          sl3.Add('ICR_H ' + Trim(Xls.GetStringFromCell(r, 21)) + '_4');
          sl3.Add('ICR_H ' + Trim(Xls.GetStringFromCell(r, 21)) + '_10');
        end;
      end
      else
      if Filter = Xls.GetStringFromCell(r, 9) then
      begin
        Add(IsBaseline, Xls.GetStringFromCell(r, 1),
            Xls.GetStringFromCell(r, 2),
            Xls.GetStringFromCell(r, 4),
            Xls.GetStringFromCell(r, 5),
            Xls.GetStringFromCell(r, 6),
            Xls.GetStringFromCell(r, 7),
            Xls.GetStringFromCell(r, 8),
            Xls.GetStringFromCell(r, 9),
            Xls.GetStringFromCell(r, 10),
            Xls.GetStringFromCell(r, 11),
            Xls.GetStringFromCell(r, 17),
            Xls.GetStringFromCell(r, 18),
            Xls.GetStringFromCell(r, 19),
            Xls.GetStringFromCell(r, 16),
            Xls.GetStringFromCell(r, 21),
            Xls.GetStringFromCell(r, 22),
            Xls.GetStringFromCell(r, 23),
            Xls.GetStringFromCell(r, 24),
            Xls.GetStringFromCell(r, 25),
            Xls.GetStringFromCell(r, 26),
            Xls.GetStringFromCell(r, 27),
            ci
            );
        // agglomerator metrics
        if Xls.GetStringFromCell(r, 18) <> '' then
        begin
          sl2.Add(Trim(Xls.GetStringFromCell(r, 18)));
          sl2.Add(Trim(Xls.GetStringFromCell(r, 18)) + '_hours');
          sl2.Add(Trim(Xls.GetStringFromCell(r, 18)) + '_labor');
          sl2.Add(Trim(Xls.GetStringFromCell(r, 18)) + '_om');
        end;
        sl1.Add(Trim(Xls.GetStringFromCell(r, 17)));
        sl1.Add(Trim(Xls.GetStringFromCell(r, 17)) + '_hours');
        sl1.Add(Trim(Xls.GetStringFromCell(r, 17)) + '_labor');
        sl1.Add(Trim(Xls.GetStringFromCell(r, 17)) + '_om');

        // Agglomerator ICR
        if Xls.GetStringFromCell(r, 21) <> '' then
        begin
          sl3.Add('ICR_C ' + Trim(Xls.GetStringFromCell(r, 21)) + '_1');
          sl3.Add('ICR_C ' + Trim(Xls.GetStringFromCell(r, 21)) + '_2');
          sl3.Add('ICR_C ' + Trim(Xls.GetStringFromCell(r, 21)) + '_3');
          sl3.Add('ICR_C ' + Trim(Xls.GetStringFromCell(r, 21)) + '_4');
          sl3.Add('ICR_C ' + Trim(Xls.GetStringFromCell(r, 21)) + '_10');
          sl3.Add('ICR_H ' + Trim(Xls.GetStringFromCell(r, 21)) + '_1');
          sl3.Add('ICR_H ' + Trim(Xls.GetStringFromCell(r, 21)) + '_2');
          sl3.Add('ICR_H ' + Trim(Xls.GetStringFromCell(r, 21)) + '_3');
          sl3.Add('ICR_H ' + Trim(Xls.GetStringFromCell(r, 21)) + '_4');
          sl3.Add('ICR_H ' + Trim(Xls.GetStringFromCell(r, 21)) + '_10');
        end;
      end;
    end;
  end;
  FreeAndNil(Xls);
```

```
//set AggID for each costing step...
fillchar(HourValue1,SizeOf(HourValue1),False);
fillchar(HourValue2,SizeOf(HourValue2),False);
for C in CostSteps do begin
  C.fAgg2ID:=sl2.IndexOf(C.fCostStepRec.Agglomerator2Xw);
  C.fAgg2IDH:=sl2.IndexOf(C.fCostStepRec.Agglomerator2Xw+ '_hours');
  HourValue2[C.fAgg2IDH]:=True;
  C.fAgg2IDL:=sl2.IndexOf(C.fCostStepRec.Agglomerator2Xw+ '_labor');
  C.fAgg2IDO:=sl2.IndexOf(C.fCostStepRec.Agglomerator2Xw+ '_om');

  C.fAgg1ID:=sl1.IndexOf(C.fCostStepRec.Agglomerator1Xw);
  C.fAgg1IDH:=sl1.IndexOf(C.fCostStepRec.Agglomerator1Xw+ '_hours');
  HourValue1[C.fAgg1IDH]:=True;
  C.fAgg1IDL:=sl1.IndexOf(C.fCostStepRec.Agglomerator1Xw+ '_labor');
  C.fAgg1IDO:=sl1.IndexOf(C.fCostStepRec.Agglomerator1Xw+ '_om');

  C.fAggICR_IDC1:=sl3.IndexOf('ICR_C ' + C.fCostStepRec.AgglomeratorICR + '_1');
  C.fAggICR_IDC2:=sl3.IndexOf('ICR_C ' + C.fCostStepRec.AgglomeratorICR + '_2');
  C.fAggICR_IDC3:=sl3.IndexOf('ICR_C ' + C.fCostStepRec.AgglomeratorICR + '_3');
  C.fAggICR_IDC4:=sl3.IndexOf('ICR_C ' + C.fCostStepRec.AgglomeratorICR + '_4');
  C.fAggICR_IDC10:=sl3.IndexOf('ICR_C ' + C.fCostStepRec.AgglomeratorICR + '_10');

  C.fAggICR_IDH1:=sl3.IndexOf('ICR_H ' + C.fCostStepRec.AgglomeratorICR + '_1');
  C.fAggICR_IDH2:=sl3.IndexOf('ICR_H ' + C.fCostStepRec.AgglomeratorICR + '_2');
  C.fAggICR_IDH3:=sl3.IndexOf('ICR_H ' + C.fCostStepRec.AgglomeratorICR + '_3');
  C.fAggICR_IDH4:=sl3.IndexOf('ICR_H ' + C.fCostStepRec.AgglomeratorICR + '_4');
  C.fAggICR_IDH10:=sl3.IndexOf('ICR_H ' + C.fCostStepRec.AgglomeratorICR + '_10');
end;

// agglomerator metric names
SetLength(AggInputs2,sl2.Count);
for r := 0 to sl2.Count - 1 do
  AggInputs2[r] := sl2[r];

SetLength(AggInputs1,sl1.Count);
for r := 0 to sl1.Count - 1 do
  AggInputs1[r] := sl1[r];

SetLength(AggICR,sl3.Count);
for r := 0 to sl3.Count - 1 do
  AggICR[r] := sl3[r];

sl2.Free;
sl1.Free;
sl3.Free;

if AnsiContainsStr(Filename, 'Baseline') then
  fErrors.SaveToFile('ss_errors_baseline.txt')
  else
    fErrors.SaveToFile('ss_errors.txt');
end;

// this procedure is called by TPWSReplicator.WriteOption
procedure TCostingSteps.ResetRandomSeeds(PWSSeed: integer);
begin
  fCostVars.ResetRandomSeeds(PWSSeed);
end;

procedure TCostingSteps.ResolveDatabaseVariables(const aSz, aSrc, aLSL,
  aCCT, aType, aPWS90PctBp1, aPWS90PctBp2: integer; const BaselineCols,
BaselineData: TStringList);
begin
  fCostVars.FillValueArray2(Variables, RawVariables, aSz, aSrc, aLSL, aCCT, aType,
1, aPWS90PctBp1, aPWS90PctBp2, true, true,
                            BaselineCols, BaselineData);
end;

// this procedure is called by TPWSReplicator.WriteBaseline3
procedure TCostingSteps.ResolveDatabaseVariables(const aSz, aSrc, aLSL, aCCT, aType,
aPWS90PctBp1, aPWS90PctBp2: integer);
begin
  fCostVars.FillValueArray(Variables, RawVariables, aSz, aSrc, aLSL, aCCT, aType, 1,
aPWS90PctBp1, aPWS90PctBp2, true, true, nil, true);
end;

procedure TCostingSteps.SetRandomYears;
var C : TCostingStep;
begin
  for C in CostSteps do
    C.ResetArrCalculateYr(fIsBaseline);
end;

procedure TCostingSteps.SetVariablesAndCalculate(const aSz, aSrc, aLSL, aCCT, aType,
aEP, aNC, aFirstAle : integer;
                                                 const aPop: integer; const
CostCapital: double;
                                                 const SetProbsTo01: boolean; const
pwsid, option: string;
                                                 const CCTCostEquations:
TCCTCostEquations; const num_lsl, pwswgt: double;
                                                 const O :
TDictionary<string,double>; const prtDebug, VLSystem: boolean;
                                                 const Small_Correct: integer; const
Bin: integer; const P90_base: double;
                                                 const Num_Proxies: integer;
slProxies: TStringList; const SchoolSampData: TSchoolSampDataRec);
```

```
var C : TCostingStep;
    Cost,Labor,OM,Hours,DoIt,V : double;
    Y : integer;
    NewDraw : boolean;
    CCT, LSL, POU : integer;
    UsefulLifeInstall,UsefulLifeMod,UsefulLifeFF : integer;
    ExistingCCT, NewCCT, FindAndFix, InstallPOU: boolean;
    ExistingCCTCostOM, AdjustCCTCostOM, InstallCCTCostOM, InstallCCTCostCap, 
InstallCCTCostCapDisc,
    InstallCCTCostCapDisc_p, FindAndFixCostOM : double;
    AdjustCCTCost, AdjustCCTOM, NewCCTCost, NewCCTOM: double;
    FindAndFixCostCap, FindAndFixCostCapDisc, AdjustCCTCostCap, 
AdjustCCTCostCapDisc: double;
    AdjustCCTCostCapDisc_p, FindAndFixCostCapDisc_p: double;

    Num_LSL_base: double;
    pp_lsl_replaced, num_lsl_replace, lslr_missed_goal : array[1..100] of double;
    Num_lsl_replace_ale: array[1..100] of double;
       // go out 51 years to avoid array overrun in year 49
    Num_lsl_remain: double;
    Meet_LSLR_Goal: integer;
    num_replace, num_remain, num_requested: double;
    num_lsl_requested: array[0..100] of double;
    NM: integer;
    num_hh_per_connect: double;
    hh_remain_lsl: double;
    lslr_conducted: boolean;
    pp_lsl_replacement_rates: array[0..100] of double;
    failCost1, failCost4, failCost5, failCost6, failCost7, failCost8: boolean;

    CV: TCostVar;
    i, iC: integer;
    sLine, sLine2: string;
    yy: integer;

    isBaseline: boolean;

    cct_adjust_yr, cct_install_yr, pou_install_yr: integer;

    pp_lslr_paper, num_lsl_base_adjust, num_replace_paper, num_replace_paper2: 
double;
    num_paper_remain: double;
    partial_cct_level: integer;
    hp_lslr_paper: array [1..3] of double;
    hp_lslr_partial: array [1..3] of double;

    prob_downstream_P_limit: integer;
    PDose, FlowLossP, ConnectionLossP: double;
    replace_rate: double;

ttLSL:double;

    owBin, tBin, owBin_tmp: integer;
    pws90pct, tpws90pct, ttpws90pct: double;
    proxy1_pws90, proxy2_pws90, proxy4_pws90: double;
    CCT_Change, bCCT_Change: boolean;
    bp1, bp2: integer;

    tmp_double: double;
    p_source_chng_yr: array[0..100] of integer;
    p_source_sig_yr: array[0..100] of integer;
    p_treat_change_yr: array[0..100] of integer;
    pp_lsl_replaced_vol_pct_yr: array[0..100] of double;
    pp_lsl_replaced_vol_actual: double;

    Num_tap_ge_al: double;
    fnf: boolean;
    ff_cct: array[0..100] of integer;
    sumff_cct, hff_cct: integer;
    hffY2, hFFY3 : integer;
    CCTB, LSLB: integer;

    SystemType: integer;
    BinChgLsl, BinChgNoLsl: array[0..100] of integer;

    b_cct_study_rec_install: array[0..100] of integer;
    b_cct_study_install: array[0..100] of integer;
    b_state_cct_treatment_install: array[0..100] of integer;
    b_cct_study_rec_mod: array[0..100] of integer;
    b_cct_study_mod: array[0..100] of integer;
    b_state_cct_treatment_mod: array[0..100] of integer;
    b_install_cct: array[0..100] of integer;
    b_install_cct_mc: array[0..100] of integer;
    p_cct_study: integer;
    proxy3_pws90: array[0..100] of double;
    InstallCCT: array[0..100] of boolean;
    AdjustCCT: array[0..100] of boolean;
    cct_study_done_yr: integer;
    b_modify_cct: array[0..100] of integer;
    b_modify_cct_mc: array[0..100] of integer;
    ff_pws90pct: double;
    b_cct_study_rec_mod_tl: array[0..100] of integer;
    b_cct_study_mod_tl: array[0..100] of integer;
    b_modify_cct_tl: array[0..100] of integer;
    b_state_cct_treatment_mod_tl: array[0..100] of integer;
    b_cct_study_rec_mod_al: array[0..100] of integer;
    b_cct_study_mod_al: array[0..100] of integer;
    b_modify_cct_al: array[0..100] of integer;
    b_state_cct_treatment_mod_al: array[0..100] of integer;
    system_pou_arr: array[0..100] of integer;

    numb_wqp_add_sites: array[0..100] of double;
    numb_wqp_add_sites_total: array[0..100] of double;
    numb_wqp_sites_added: array[0..100] of double;
    numb_wqp_sites_added_prev: array[0..100] of double;
    num_lsl_paper: array[0..100] of double;

    SourceTreatChangeEver: boolean;
    lsl_start_yr: integer;
begin
{$R+}
  fnum_proxies := num_proxies;

  fillchar(Values2,SizeOf(Values2),0);
  fillchar(Values1,SizeOf(Values1),0);
  fillchar(Values2p,SizeOf(Values2p),0);
  fillchar(Values1p,SizeOf(Values1p),0);
  fillchar(Values2Y,SizeOf(Values2Y),0);

  fillchar(ValuesCapital,SizeOf(ValuesCapital),0);
  fillchar(ValuesICR,SizeOf(ValuesICR),0);

  fillchar(pws90pctCCT_yr, SizeOf(pws90pctCCT_yr), 0);
  fillchar(pws90pctLSL_yr, SizeOf(pws90pctLSL_yr), 0);

  SystemType := aType;
  LSL := aLSL;
  CCT := aCCT;
  POU := 0;

  CCTB := CCT;
  LSLB := LSL;

  for i := 0 to Config.YearsOfAnalysis do
  begin
    InstallCCT[i] := false;
    AdjustCCT[i] := false;
  end;

  ExistingCCT := false;
  NewCCT := false;
  InstallPOU := false;
  FindAndFix := false;
  NewDraw := true;

  ExistingCCTCostOM:=0;
  AdjustCCTCostOM:=0;
  InstallCCTCostOM:=0;
  InstallCCTCostCap:=0;
  InstallCCTCostCapDisc:=0;
  InstallCCTCostCapDisc_p:=0;
  FindAndFixCostOM:=0;
  FindAndFixCostCap:=0;
  FindAndFixCostCapDisc:=0;
  AdjustCCTCostCap:=0;
  AdjustCCTCostCapDisc:=0;
  AdjustCCTCostCapDisc_p:=0;
  FindAndFixCostCapDisc_p:=0;

  HasLSLRCost := false;
  HasCCTCost := false;
ttLSL:=0;
  LSLReplaced := 0;
  LSLReplacedMandatory := 0;
  LSLReplacedVoluntary := 0;
  CCTInstalled := false;
  CCTAdjusted := false;
  CCTAdjusted_ale := false;
  CCTAdjusted_tle := false;
  CCTExisting := false;
  HasFindAndFixCost := false;
  POUInstalled := false;

  AdjustCCTCost:=0;
  AdjustCCTOM:=0;
  NewCCTCost:=0;
  NewCCTOM:=0;

  cct_adjust_yr := 0;
  cct_install_yr := 0;
  pou_install_yr := 0;
  partial_cct_level := 0;

  prob_downstream_P_limit := -1;

  POTWCost := 0;
  prerule_ploading_lbs_5 := 0;
  prerule_ploading_lbs_15 := 0;
  prerule_ploading_lbs_25 := 0;
  prerule_ploading_lbs_35 := 0;
  postrule_ploading_lbs_5 := 0;
  postrule_ploading_lbs_15 := 0;
  postrule_ploading_lbs_25 := 0;
  postrule_ploading_lbs_35 := 0;
  incr_ploading_lbs_5 := 0;
  incr_ploading_lbs_15 := 0;
  incr_ploading_lbs_25 := 0;
  incr_ploading_lbs_35 := 0;
  count_incr_ploading_lbs_5 := 0;
  count_incr_ploading_lbs_15 := 0;
  count_incr_ploading_lbs_25 := 0;
  count_incr_ploading_lbs_35 := 0;

  fillchar(b_cct_study_rec_install,SizeOf(b_cct_study_rec_install),0);
  fillchar(b_cct_study_install,SizeOf(b_cct_study_install),0);
  fillchar(b_state_cct_treatment_install,SizeOf(b_state_cct_treatment_install),0);
  fillchar(b_cct_study_rec_mod,SizeOf(b_cct_study_rec_mod),0);
  fillchar(b_cct_study_mod,SizeOf(b_cct_study_mod),0);
  fillchar(b_state_cct_treatment_mod,SizeOf(b_state_cct_treatment_mod),0);
  fillchar(b_install_cct,SizeOf(b_install_cct),0);
  fillchar(b_install_cct_mc,SizeOf(b_install_cct_mc),0);
  fillchar(proxy3_pws90,SizeOf(proxy3_pws90),0);
  cct_study_done_yr := 0;
  fillchar(b_modify_cct,SizeOf(b_modify_cct),0);
  fillchar(b_modify_cct_mc,SizeOf(b_modify_cct_mc),0);
  ff_pws90pct := 0;
  fillchar(b_cct_study_rec_mod_tl, SizeOf(b_cct_study_rec_mod_tl), 0);
  fillchar(b_cct_study_mod_tl, SizeOf(b_cct_study_mod_tl), 0);
  fillchar(b_modify_cct_tl, SizeOf(b_modify_cct_tl), 0);
  fillchar(b_state_cct_treatment_mod_tl, SizeOf(b_state_cct_treatment_mod_tl), 0);
  fillchar(b_cct_study_rec_mod_al, SizeOf(b_cct_study_rec_mod_al), 0);
  fillchar(b_cct_study_mod_al, SizeOf(b_cct_study_mod_al), 0);
  fillchar(b_modify_cct_al, SizeOf(b_modify_cct_al), 0);
  fillchar(b_state_cct_treatment_mod_al, SizeOf(b_state_cct_treatment_mod_al), 0);
  fillchar(system_pou_arr, SizeOf(system_pou_arr), 0);

  fillchar(numb_wqp_add_sites, SizeOf(numb_wqp_add_sites), 0);
  fillchar(numb_wqp_add_sites_total, SizeOf(numb_wqp_add_sites_total), 0);
  fillchar(numb_wqp_sites_added, SizeOf(numb_wqp_sites_added), 0);
  fillchar(numb_wqp_sites_added_prev, SizeOf(numb_wqp_sites_added_prev), 0);
  fillchar(num_lsl_paper, SizeOf(num_lsl_paper), 0);

  SourceTreatChangeEver := false;
  lsl_start_yr := 0;

  // if non-blank pwsid is passed to function, set up column headings for output 
debug files
  if prtDebug then
  begin
```

```
    slVariables.Clear;
    sLine2 := 'VarName' + chr(9) + 'Year' + chr(9) + 'CurValue' + chr(9) + 
'RawValue';
    slVariables.Add(sLine2);

    slCosts.Clear;
    sLine2 := 'owBin' + chr(9) + 'CostName' + chr(9) + 'Year' + chr(9) + 'Type' + 
chr(9) +
              'Agglomerator2' + chr(9) + 'Expression' + chr(9) +
              'Amount1' + chr(9) + 'Amount2' + chr(9) +
              'Bin1' + chr(9) +'Bin2' + chr(9) +'Bin3' + chr(9);
    slCosts.Add(sLine2);

    slCalcs.Clear;
    sLine2 := 'LSL' + chr(9) + 'Num_LSL_base'
                    + chr(9) + 'Year'
                    + chr(9) + 'pws_lslr'
                    + chr(9) + 'pp_lsl_replaced'
                    + chr(9) + 'num_replace'
                    + chr(9) + 'num_lsl_remain'
                    + chr(9) + 'hh_remain_lsl'
                    + chr(9) + 'Num_lsl_replace'
                    + chr(9) + 'Meet_LSLR_Goal'
                    + chr(9) + 'NM'
                    + chr(9) + 'Num_lsl_replace_ale'
                    + chr(9) + 'pp_lslr_paper'
                    + chr(9) + 'replace_rate'
                    ;
    slCalcs.Add(sLine2);

    slCCTCosts.Clear;
    sLine2 := 'Year' + chr(9) + 'aPop' + chr(9) +
                'ExistingCCTCostOM' + chr(9) +
                'AdjustCCTCostOM' + chr(9) +
                'InstallCCTCostOM' + chr(9) + 'cctCostCap' + chr(9) +
                'FindAndFixCostOM' + chr(9) +
                'iBaselinepo4dose' + chr(9) +
                'iBaselineph_w' + chr(9) +
                'iBaselineph_wo' + chr(9) +
                'pbaseph' + chr(9) +
                'pbasepo4' + chr(9) +
                'pbasephpo4' + chr(9) +
                'DFlowEP' + chr(9) +
                'AFlowEP' + chr(9) +
                'EntryPoints';
    slCCTCosts.Add(sLine2);
  end;
  if option = 'Baseline' then isBaseline := true
  else isBaseline := false;

  // read data request data values from database
  fCostVars.FillValueArray(Variables, RawVariables, aSz, aSrc, LSL, CCT, aType, 1,
Config.PWS90PctBp1, Config.PWS90PctBp2, SetProbsTo01, NewDraw,
                           O, isBaseline);
  NewDraw := false;

  Variables[fI.p_lsl] := LSL;

  // load external variables values
  Variables[fI.EP] := aEP;
  Variables[fI.Pws_Cct] := CCT;
  Variables[fI.Pws_first_ale] := aFirstAle;

  Variables[fI.Pws_sw] := 0;
  Variables[fI.Pws_gw] := 0;
  if aSrc = 2 then
    Variables[fI.Pws_sw] := 1
  else if aSrc = 1 then
    Variables[fI.Pws_gw] := 1;

  Variables[fI.Pws_pop] := aPop;
  if aNC > 0 then
    num_hh_per_connect := (aPop / Variables[fI.Numb_hh]) / aNC
  else
    num_hh_per_connect := 0;

  Variables[fI.meet_lslr_goal] := 1;
  Variables[fI.fail_nm1] := 0;
  Variables[fI.fail_nm2] := 0;

  failCost1 := false;
  failCost4 := false;
  failCost5 := false;
  failCost6 := false;
  failCost7 := false;
  failCost8 := false;

  InitCCTBVarsToZero(option);

  Variables[fI.cct_existing_cost] := 0;
  Variables[fI.cct_modify_cost] := 0;
  Variables[fI.cct_install_cost] := 0;
  Variables[fI.cct_findfix_cost] := 0;

  fillchar(pp_lsl_replaced,SizeOf(pp_lsl_replaced),0);
  fillchar(num_lsl_replace,SizeOf(num_lsl_replace),0);
  fillchar(Num_lsl_replace_ale,SizeOf(Num_lsl_replace_ale),0);
  fillchar(lslr_missed_goal,SizeOf(lslr_missed_goal),0);
  fillchar(num_lsl_requested,SizeOf(num_lsl_requested),0);
  fillchar(ff_cct, SizeOf(ff_cct), 0);
  hff_cct := 0;

  p_cct_study := trunc(Variables[fI.p_cct_study]);

  Num_LSL_base := 0;
  num_lsl_remain := 0;
  num_paper_remain := 0;
  LSLRequested := 0;

  if LSL = 1 then
  begin
    //Num_LSL_base := Variables[fI.perc_lsl] * aNC;
    Num_LSL_base := num_lsl;
    num_lsl_remain := Num_LSL_base;
    hh_remain_lsl := max(0, (num_hh_per_connect * num_lsl_remain));
  end;

  NM := 0;

  for i := 4 to Config.YearsOfAnalysis do
  begin
    O.TryGetValue('pp_lsl_replacement_' + i.ToString, pp_lsl_replacement_rates[i])
  end;

  for i := 1 to Config.YearsOfAnalysis do
  begin
    if i <= Config.YearsOfOutput then
    begin
      O.TryGetValue('p_source_chng_' + i.ToString, tmp_double);
      p_source_chng_yr[i] := trunc(tmp_double);
    end
    else
      p_source_chng_yr[i] := 0;
  end;

  for i := 1 to Config.YearsOfAnalysis do
  begin
    if i <= Config.YearsOfOutput then
    begin
      O.TryGetValue('p_source_sig_' + i.ToString, tmp_double);
      p_source_sig_yr[i] := trunc(tmp_double);
    end
    else
```

```
      p_source_sig_yr[i] := 0;
  end;

  for i := 1 to Config.YearsOfAnalysis do
  begin
    if i <= Config.YearsOfOutput then
    begin
      O.TryGetValue('p_treat_change_' + i.ToString, tmp_double);
      p_treat_change_yr[i] := trunc(tmp_double);
    end
    else
      p_treat_change_yr[i] := 0;
  end;

  for i := 1 to Config.YearsOfAnalysis do
  begin
    O.TryGetValue('pp_lsl_replaced_vol_pct_' + i.ToString,
pp_lsl_replaced_vol_pct_yr[i]);
  end;

  for i := 1 to Config.YearsOfAnalysis do
  begin
    O.TryGetValue('BinChgLSL_' + i.ToString, tmp_double);
    BinChgLsl[i] := trunc(tmp_double);

    O.TryGetValue('BinChgNoLSL_' + i.ToString, tmp_double);
    BinChgNoLsl[i] := trunc(tmp_double);
  end;

  // Compute costs from CCTCostEquations
  if not VLSystem then begin
    if CCT = 1 then
    begin
      CCTCostEquations.ExistingCCT;
      UsefulLifeMod:=round(CCTCostEquations.UsefulLifeOM);
      ExistingCCTCostOM := CCTCostEquations.ComputeOMCost;
      CCTCostEquations.AdjustCCT(Variables[fI.targetph], Variables[fI.targetpo4]);
      AdjustCCTCostOM := CCTCostEquations.ComputeOMCost;
      AdjustCCTCostCap := CCTCostEquations.ComputeCapitalCost;
      AdjustCCTCostCapDisc:= AdjustCCTCostCap * (DiscRate / (1 - Power((1 +
DiscRate),-CCTCostEquations.UsefulLifeCap)));
      AdjustCCTCostCapDisc_p:= AdjustCCTCostCap * (CostCapital / (1 - Power((1 +
CostCapital),-CCTCostEquations.UsefulLifeCap)));
    end
    else
    begin
      CCTCostEquations.NewCCT(Variables[fI.targetph], Variables[fI.targetpo4]);
      UsefulLifeInstall:=round(CCTCostEquations.UsefulLifeCap);
      InstallCCTCostOM := CCTCostEquations.ComputeOMCost;
      InstallCCTCostCap := CCTCostEquations.ComputeCapitalCost;
      InstallCCTCostCapDisc:= InstallCCTCostCap * (DiscRate / (1 - Power((1 + 
DiscRate),-CCTCostEquations.UsefulLifeCap)));
      InstallCCTCostCapDisc_p:= InstallCCTCostCap * (CostCapital / (1 - Power((1 + 
CostCapital),-CCTCostEquations.UsefulLifeCap)));
    end;

    if CCT = 0 then begin
      if 
CCTCostEquations.arrBaselineph_woCCT[aSrc,CCTCostEquations.iBaselineph_woCCT] < 7.5 
then begin
        CCTCostEquations.pbasepo4 := 0;
        CCTCostEquations.pbasephpo4 := 1;
      end;
    end;

    CCTCostEquations.FindAndFixCCT(CCTB);
    UsefulLifeFF:=round(CCTCostEquations.UsefulLifeCap);
    FindAndFixCostOM := CCTCostEquations.ComputeOMCost;
    FindAndFixCostCap := CCTCostEquations.ComputeCapitalCost;
    FindAndFixCostCapDisc:= FindAndFixCostCap * (DiscRate / (1 - Power((1 + 
DiscRate),-CCTCostEquations.UsefulLifeCap)));
    FindAndFixCostCapDisc_p:= FindAndFixCostCap * (CostCapital / (1 - Power((1 + 
CostCapital),-CCTCostEquations.UsefulLifeCap)));


  end;

  fAdjust_CCT := 0;
  fInstall_CCT := 0;

  lslr_conducted := false;

  if isBaseline then
  begin
    hp_lslr_partial[1] := Variables[fI.pp_lslr_partial];
  end
  else
  if  (option = 'OW') or (option='OW5L') then
  begin
    hp_lslr_paper[1] := Variables[fI.pp_lslr_paper];
    hp_lslr_partial[1] := Variables[fI.pp_lslr_partial];

    if Num_Proxies = 0 then
    begin
      if Round(Config.DiscountRate*100)/100 = 0.03 then
        Variables[fI.annual_pou_cost_hh] := 111
      else if Round(Config.DiscountRate*100)/100 = 0.07 then
        Variables[fI.annual_pou_cost_hh] := 114
      else
        Variables[fI.annual_pou_cost_hh] := -1;
    end
    else
      Variables[fI.annual_pou_cost_hh] := 114;
  end;

  CCT_Change := false;
  bCCT_Change := false;

  if not VLSystem then
  begin
    if isBaseline then
      LeadConcentrationBins(pwsid, option, 0, aSz, aSrc, LSL, CCT, POU, aPop, aNC,
                            fAdjust_CCT, fInstall_CCT, cct_adjust_yr,
cct_install_yr, pou_install_yr,
                            bCCT_Change, pwswgt, 0, 0, 0, Num_Proxies,
partial_cct_level,
                            hp_lslr_paper, hp_lslr_partial,
pp_lsl_replacement_rates,true)
    else
      LeadConcentrationBins(pwsid, option, 0, aSz, aSrc, LSL, CCT, POU, aPop, aNC,
                            fAdjust_CCT, fInstall_CCT, cct_adjust_yr,
cct_install_yr, pou_install_yr,
                            bCCT_Change, pwswgt, 0, 0, 0, Num_Proxies,
partial_cct_level,
                            hp_lslr_paper, hp_lslr_partial,
pp_lsl_replaced_vol_pct_yr,true)
  end;

  if option = 'Baseline' then begin
    bp1 := 10;
    bp2 := 15;
  end
  else begin
    bp1 := Config.PWS90PctBp1;
    bp2 := Config.PWS90PctBp2;
  end;

  if not VLSystem then
  begin
    owBin := Bin;

    if owBin = 1 then pws90pct := bp2 + 5
    else if owBin = 2 then pws90pct := bp1 + ((bp2-bp1)/2)
    else if owBin = 3 then
    begin
      if Variables[fI.p_bin3_nonzero] = 1 then pws90pct := bp1/2
      else pws90pct := 0;
    end;
  end;

  owBin_tmp := 0;

  if option <> 'Baseline' then
  begin
    if Config.VolLeadProg = 0 then
      Variables[fI.p_vol_leadtap_prog] := 0;
  end;

  // Year loop
  for Y:=1 to fYears do begin
    if (num_proxies = 0) and (y > Config.YearsOfOutput) then
      continue;

    Variables[fI.school_1a] := 0;
    Variables[fI.school_1b] := 0;
    Variables[fI.school_3a] := 0;
    Variables[fI.school_3b] := 0;
    Variables[fI.school_3c] := 0;
    Variables[fI.school_3d] := 0;
    Variables[fI.school_5a] := 0;
    Variables[fI.school_5b] := 0;

    if Config.SchoolOption = 'school_1a' then
      Variables[fI.school_1a] := 1
    else if Config.SchoolOption = 'school_1b' then
      Variables[fI.school_1b] := 1
    else if Config.SchoolOption = 'school_3a' then
      Variables[fI.school_3a] := 1
    else if Config.SchoolOption = 'school_3b' then
      Variables[fI.school_3b] := 1
    else if Config.SchoolOption = 'school_3c' then
      Variables[fI.school_3c] := 1
    else if Config.SchoolOption = 'school_3d' then
      Variables[fI.school_3d] := 1
    else if Config.SchoolOption = 'school_5a' then
      Variables[fI.school_5a] := 1
    else if Config.SchoolOption = 'school_5b' then
      Variables[fI.school_5b] := 1;

    if option <> 'Baseline' then
    begin
      Variables[fI.numb_second_schools_pub] :=
SchoolSampData.numb_second_schools_pub;
      Variables[fI.numb_elem_schools_pub] := SchoolSampData.numb_elem_schools_pub;
      Variables[fI.numb_second_schools_priv] :=
SchoolSampData.numb_second_schools_priv;
      Variables[fI.numb_elem_schools_priv] := SchoolSampData.numb_elem_schools_priv;
      Variables[fI.numb_daycares] := SchoolSampData.numb_daycares;
      Variables[fI.p_grandfather_opt_pub] := SchoolSampData.p_grandfather_opt_pub;
      Variables[fI.p_grandfather_opt_priv] := SchoolSampData.p_grandfather_opt_priv;
      Variables[fI.p_grandfather_mand_pub] := SchoolSampData.p_grandfather_mand_pub;
      Variables[fI.p_grandfather_mand_priv] :=
SchoolSampData.p_grandfather_mand_priv;
      Variables[fI.p_grandfather_opt_child] :=
SchoolSampData.p_grandfather_opt_child;
      Variables[fI.p_grandfather_mand_child] :=
SchoolSampData.p_grandfather_mand_child;
    end;

    Variables[fI.b_state_one] := 0;
    Variables[fI.b_state_two] := 0;

    if (SchoolSampData.stateabb = 'AR') or (SchoolSampData.stateabb = 'LA') or
       (SchoolSampData.stateabb = 'MS') or (SchoolSampData.stateabb = 'MO') or
       (SchoolSampData.stateabb = 'SC') or (SchoolSampData.stateabb = 'ND') then
      Variables[fI.b_state_one] := 1;

    if (SchoolSampData.stateabb = 'AR') or (SchoolSampData.stateabb = 'LA') or
       (SchoolSampData.stateabb = 'MS') or (SchoolSampData.stateabb = 'MO') or
       (SchoolSampData.stateabb = 'SC') then
      Variables[fI.b_state_two] := 1;

//Added 2/27/19 - s
NEWDRAW:=FALSE;


    // Tracking PWS LSLR and LSLR Goal Failure Costs
    replace_rate := 0;

    tpws90pct := 999999;
    ttpws90pct := 999999;
    if (option = 'Baseline') and not VLSystem then
    begin
      Variables[fI.p_source_chng] := p_source_chng_yr[y];
      Variables[fI.p_source_sig] := p_source_sig_yr[y];
      Variables[fI.p_treat_change] := p_treat_change_yr[y];

      // Calculate Annual CCT and Find & Fix Micro Costs
      if CCTB = 1 then
      begin
```

```
        Variables[fI.cct_existing_cost] := ExistingCCTCostOM;
        CCTExisting := true;
      end
      else
        Variables[fI.cct_existing_cost] := 0;

      if y >= 4 then
      begin
        // Random change in PWS_90 due to sampling variation
        //proxy1_pws90 := pws90pct * (1 + Variables[fI.rnd_p90_error]);
        proxy1_pws90 := pws90pct * (1 + 0);

        // Change in water source or treatment technology
        if (b_install_cct[y] + b_modify_cct[y] > 0) or (POU = 1) then
        begin
          if ((p_source_chng_yr[y]*p_source_sig_yr[y]) = 0) and
(p_treat_change_yr[y] = 0) then proxy2_pws90 := proxy1_pws90
          else
          begin
            if LSL = 1 then
              tBin := BinChgLsl[y]
            else
              tBin := BinChgNoLsl[y];

            if tBin = 1 then
              tpws90pct := bp2 + 5
            else
            if tBin = 2 then
              tpws90pct := bp1 + ((bp2-bp1)/2)
            else
            if tBin = 3 then
              tpws90pct := bp1/2;

            if tpws90pct < proxy1_pws90 then
            begin
              proxy2_pws90 := tpws90pct;
            end
            else
              proxy2_pws90 := proxy1_pws90;

            SourceTreatChangeEver := true;
          end;
        end
        else
        begin
          if ((p_source_chng_yr[y]*p_source_sig_yr[y]) = 0) and
(p_treat_change_yr[y] = 0) then proxy2_pws90 := proxy1_pws90
          else
          begin
            if LSL = 1 then
              tBin := BinChgLsl[y]
            else
              tBin := BinChgNoLsl[y];

            if tBin = 1 then
              tpws90pct := bp2 + 5
            else
            if tBin = 2 then
              tpws90pct := bp1 + ((bp2-bp1)/2)
            else
            if tBin = 3 then
              tpws90pct := bp1/2;

            proxy2_pws90 := tpws90pct;

            SourceTreatChangeEver := true;
          end;
        end;

        // CCT and POU not VLS
        if not VLSystem then
        begin
          if (proxy2_pws90 > bp2) and (CCT = 0) and (not CCT_Change) and
(p_cct_study = 1) then
          begin
            for i := y + 5 to Config.YearsOfAnalysis do
              InstallCCT[i] := true;
            //225change
            proxy3_pws90[y+7] := Variables[fI.post_cct_p90_bin1];
            CCT_Change := true;
            if cct_install_yr = 0 then cct_install_yr := y + 5;

            b_cct_study_rec_install[y+1] := 1;
            b_cct_study_install[y+3] := 1;
            for i := y + 5 to Config.YearsOfAnalysis do
              b_install_cct[i] := 1;
            b_install_cct_mc[y+6] := 1;

            // System Capital Cost undiscounted
            ValuesCapital[2] := ValuesCapital[2] + InstallCCTCostCap;
          end
          else
          if (proxy2_pws90 > bp2) and (CCT = 0) and (not CCT_Change) and
(p_cct_study = 0) then
          begin
            for i := y + 4 to Config.YearsOfAnalysis do
              InstallCCT[i] := true;
            //225change
            proxy3_pws90[y+6] := Variables[fI.post_cct_p90_bin1];
            CCT_Change := true;
            if cct_install_yr = 0 then cct_install_yr := y + 4;

            b_cct_study_rec_install[y+1] := 1;
            b_state_cct_treatment_install[y+2] := 1;
            for i := y + 4 to Config.YearsOfAnalysis do
              b_install_cct[i] := 1;
            b_install_cct_mc[y+5] := 1;

            // System Capital Cost undiscounted
            ValuesCapital[2] := ValuesCapital[2] + InstallCCTCostCap;
          end
          else
          if (proxy2_pws90 > bp2) and (CCT = 1) and (not CCT_Change) and
(p_cct_study = 1) then
          begin
            for i := y + 4 to Config.YearsOfAnalysis do
            begin
              AdjustCCT[i] := true;
              b_modify_cct[i] := 1;
            end;
            //225change
            proxy3_pws90[y+6] := Variables[fI.post_cct_p90_bin1];
            CCT_Change := true;
            if cct_adjust_yr = 0 then cct_adjust_yr := y + 4;

            b_cct_study_rec_mod[y+1] := 1;
            b_cct_study_mod[y+3] := 1;
            b_modify_cct_mc[y+5] := 1;
          end
          else
          if (proxy2_pws90 > bp2) and (CCT = 1) and (not CCT_Change) and
(p_cct_study = 0) then
          begin
            for i := y + 3 to Config.YearsOfAnalysis do
              AdjustCCT[i] := true;
            //225change
            proxy3_pws90[y+5] := Variables[fI.post_cct_p90_bin1];
            CCT_Change := true;
            if cct_adjust_yr = 0 then cct_adjust_yr := y + 3;

            b_cct_study_rec_mod[y+1] := 1;
            b_state_cct_treatment_mod[y+2] := 1;
            for i := y + 3 to Config.YearsOfAnalysis do
              b_modify_cct[i] := 1;
            b_modify_cct_mc[y+4] := 1;
          end;

          if proxy3_pws90[y] = 0 then
            proxy3_pws90[y] := proxy2_pws90;
        end; // end if not VLS
      end; // y >= 4
    end // if option = 'Baseline'
    else
    if ((option = 'OW') or (option='OW5L')) and not VLSystem then
    begin
      Variables[fI.p_source_chng] := p_source_chng_yr[y];
      Variables[fI.p_source_sig] := p_source_sig_yr[y];
      Variables[fI.p_treat_change] := p_treat_change_yr[y];

      // Calculate Annual CCT and Find & Fix Micro Costs
      if CCTB = 1 then
      begin
        Variables[fI.cct_existing_cost] := ExistingCCTCostOM;
        CCTExisting := true;
      end
      else
        Variables[fI.cct_existing_cost] := 0;

      if y >= 4 then
      begin
        //calculate the additional WQP sites in year y without regard to maximum
        if owBin = 3 then begin
          if Variables[fI.p_tap_annual] = 1 then
            numb_wqp_add_sites[y] := Variables[fI.pp_overlap_find_fix] *
Variables[fI.pp_above_al_bin_three] * Variables[fI.numb_reduced_tap]
          else if (Variables[fI.p_tap_triennial] = 1) and
                  ((y=4) or (y=7) or (y=10) or (y=13) or (y=16) or (y=19) or
                   (y=22) or (y=25) or (y=28) or (y=31) or (y=34)) then
            numb_wqp_add_sites[y] := Variables[fI.pp_overlap_find_fix] *
Variables[fI.pp_above_al_bin_three] * Variables[fI.numb_reduced_tap]
          else if (Variables[fI.p_tap_nine] = 1) and
                  ((y=4) or (y=13) or (y=22) or (y=32)) then
            numb_wqp_add_sites[y] := Variables[fI.pp_overlap_find_fix] *
Variables[fI.pp_above_al_bin_three] * Variables[fI.numb_reduced_tap]
          else numb_wqp_add_sites[y] := Variables[fI.pp_overlap_find_fix] *
Variables[fI.pp_above_al_bin_three] * (2 * Variables[fI.numb_samp_customer]);
        end
        else if owBin = 2 then
          numb_wqp_add_sites[y] := Variables[fI.pp_overlap_find_fix] *
Variables[fI.pp_above_al_bin_two] * Variables[fI.numb_samp_customer]
        else if owBin = 1 then
          numb_wqp_add_sites[y] := Variables[fI.pp_overlap_find_fix] *
```

```
Variables[fI.pp_above_al_bin_one] * (2 * Variables[fI.numb_samp_customer]);

        //calculate the total number of additional wqp samples in by year y with max 
applied
        numb_wqp_add_sites_total[y] := min((numb_wqp_add_sites_total[y-1] + 
numb_wqp_add_sites[y]), Variables[fI.numb_enhance_wqp]);

        //numb_reduced_wqp (which are samples) = 2* number of baseline reduced wqp 
sites
        //numb_enhance_wqp (which are samples)= 2* number of baseline enhanced sites

        //calculates the added number of sites in year y

        numb_wqp_sites_added[y] := numb_wqp_add_sites_total[y] - 
numb_wqp_add_sites_total[y-1];
        numb_wqp_sites_added_prev[y] := numb_wqp_add_sites_total[y] - 
numb_wqp_sites_added[y];

        // Random change in PWS_90 due to sampling variation
        proxy1_pws90 := pws90pct * (1 + 0);

        // Change in water source or treatment technology
        if (b_install_cct[y] + b_modify_cct[y] > 0) or (POU = 1) then
        begin
          if ((p_source_chng_yr[y]*p_source_sig_yr[y]) = 0) and 
(p_treat_change_yr[y] = 0) then proxy2_pws90 := proxy1_pws90
          else
          begin
            if LSL = 1 then
              tBin := BinChgLsl[y]
            else
              tBin := BinChgNoLsl[y];

            if tBin = 1 then
              tpws90pct := bp2 + 5
            else
            if tBin = 2 then
              tpws90pct := bp1 + ((bp2-bp1)/2)
            else
            if tBin = 3 then
              tpws90pct := bp1/2;

            if tpws90pct < proxy1_pws90 then
            begin
              proxy2_pws90 := tpws90pct;
            end
            else
              proxy2_pws90 := proxy1_pws90;
            SourceTreatChangeEver := true;
          end;
        end
        else
        begin
          if ((p_source_chng_yr[y]*p_source_sig_yr[y]) = 0) and
(p_treat_change_yr[y] = 0) then proxy2_pws90 := proxy1_pws90
          else
          begin
            if LSL = 1 then
              tBin := BinChgLsl[y]
            else
              tBin := BinChgNoLsl[y];

            if tBin = 1 then
              tpws90pct := bp2 + 5
            else
            if tBin = 2 then
              tpws90pct := bp1 + ((bp2-bp1)/2)
            else
            if tBin = 3 then
              tpws90pct := bp1/2;

            proxy2_pws90 := tpws90pct;

            SourceTreatChangeEver := true;
          end;
        end;

        // CCT, LSLR, and POU Plans/Studies
        if Config.SmallSystemFlexibility then
        begin
          if (SystemType = 1) and (aPop > Config.SmallProxyPop) then
          begin
            if LSL = 1 then
              Variables[fI.b_lslr_study] := 1;

            if (proxy2_pws90 > bp1) and (CCT = 0) then begin
              b_cct_study_install[y+2] := 1;
              cct_study_done_yr := y + 2;
            end;
          end
          else
          begin
            if (LSL = 1) and (Small_Correct = 1) then
              Variables[fI.b_lslr_study] := 1;
            if (proxy2_pws90 > bp1) and (CCT = 0) and (Small_Correct = 2) then
            begin
              b_cct_study_install[y+2] := 1;
              cct_study_done_yr := y + 2;
            end;
          end;
        end
        else
        // SmallSystemFlexibity is false
        begin
          if LSL = 1 then
            Variables[fI.b_lslr_study] := 1;

          if (proxy2_pws90 > bp1) and (CCT = 0) then begin
            b_cct_study_install[y+2] := 1;
            cct_study_done_yr := y + 2;
          end;
        end;

        // CCT and POU not VLS
        if not VLSystem then
        begin
          if Config.SmallSystemFlexibility then
          begin
            if (SystemType = 1) and (aPop > Config.SmallProxyPop) then
            begin
{$IFNDEF NOTRIGCCT}
              if (proxy2_pws90 > bp1) and (proxy2_pws90 < bp2) and (CCT = 1) and
(not CCT_Change) then
              begin
                for i := y + 3 to Config.YearsOfAnalysis do
                begin
                  AdjustCCT[i] := true;
                  b_modify_cct[i] := 1;
                  b_modify_cct_tl[i] := 1;
                end;

                proxy3_pws90[y+5] := Variables[fi.post_cct_p90_bin2];
                CCT_Change := true;
                if cct_adjust_yr = 0 then cct_adjust_yr := y + 3;

                b_cct_study_rec_mod[y+1] := 1;
                b_cct_study_mod[y+1] := 1;

                b_cct_study_rec_mod_tl[y+1] := 1;
                b_cct_study_mod_tl[y+1] := 1;
              end
              else
```

```
{$ENDIF}
              if (proxy2_pws90 > bp2) and (CCT = 1) and (not CCT_Change) then
              begin
                for i := y + 4 to Config.YearsOfAnalysis do
                begin
                  AdjustCCT[i] := true;
                  b_modify_cct[i] := 1;
                  b_modify_cct_al[i] := 1;
                end;

                proxy3_pws90[y+6] := Variables[fi.post_cct_p90_bin1];
                CCT_Change := true;
                if cct_adjust_yr = 0 then cct_adjust_yr := y + 4;

                b_cct_study_rec_mod[y+1] := 1;
                b_cct_study_mod[y+3] := 1;

                b_cct_study_rec_mod_al[y+1] := 1;
                b_cct_study_mod_al[y+3] := 1;
              end
              else
              if (proxy2_pws90 > bp2) and (CCT = 0) and (not CCT_Change) then
              begin
                for i := max(cct_study_done_yr+2,y+4) to Config.YearsOfAnalysis do
                begin
                  InstallCCT[i] := true;
                  b_install_cct[i] := 1;
                end;
                if cct_install_yr = 0 then cct_install_yr :=
max(cct_study_done_yr+2,y+4);

                proxy3_pws90[max(cct_study_done_yr+4,y+6)] :=
Variables[fi.post_cct_p90_bin1];
                CCT_Change := true;

                // System Capital Cost undiscounted
                ValuesCapital[2] := ValuesCapital[2] + InstallCCTCostCap;
              end;

              if proxy3_pws90[y] = 0 then
                proxy3_pws90[y] := proxy2_pws90;
            end  //if (SystemType = 1) and (aPop > Config.SmallProxyPop) then
            else
            begin
{$IFNDEF NOTRIGCCT}
              if (proxy2_pws90 > bp1) and (proxy2_pws90 < bp2) and (CCT = 1) and
(not CCT_Change) and (POU = 0) then
              begin
```

```
                for i := y + 3 to Config.YearsOfAnalysis do
                begin
                  AdjustCCT[i] := true;
                  b_modify_cct[i] := 1;
                  b_modify_cct_tl[i] := 1;
                end;

                proxy3_pws90[y+5] := Variables[fi.post_cct_p90_bin2];
                CCT_Change := true;
                if cct_adjust_yr = 0 then cct_adjust_yr := y + 3;

                b_cct_study_rec_mod[y+1] := 1;
                b_cct_study_mod[y+1] := 1;

                b_cct_study_rec_mod_tl[y+1] := 1;
                b_cct_study_mod_tl[y+1] := 1;
              end
              else
{$ENDIF}
              if (proxy2_pws90 > bp2) and (CCT = 1) and (not CCT_Change) and (POU =
0) and (Small_Correct = 2) then
              begin
                for i := y + 4 to Config.YearsOfAnalysis do
                begin
                  AdjustCCT[i] := true;
                  b_modify_cct[i] := 1;
                  b_modify_cct_al[i] := 1;
                end;

                proxy3_pws90[y+6] := Variables[fi.post_cct_p90_bin2];
                CCT_Change := true;
                if cct_adjust_yr = 0 then cct_adjust_yr := y + 4;

                b_cct_study_rec_mod[y+1] := 1;
                b_cct_study_mod[y+3] := 1;

                b_cct_study_rec_mod_al[y+1] := 1;
                b_cct_study_mod_al[y+3] := 1;
              end
              else
              if (proxy2_pws90 > bp2) and (CCT = 0) and (not CCT_Change) and (POU =
0) and (Small_Correct = 2) then
              begin
                for i := y + 4 to Config.YearsOfAnalysis do
                begin
                  InstallCCT[i] := true;
                  b_install_cct[i] := 1;
                end;
```

```
if (LSL = 1) and (num_proxies = 0) and (InstallCCT[y+4]) then
  Config.Log.Text := 'LSL installing: pwsid = ' + pwsid;

                b_cct_study_install[y+2] := 1;
                cct_study_done_yr := y + 2;

                proxy3_pws90[y+6] := Variables[fi.post_cct_p90_bin1];
                CCT_Change := true;
                if cct_install_yr = 0 then cct_install_yr := y + 4;

                // System Capital Cost undiscounted
                ValuesCapital[2] := ValuesCapital[2] + InstallCCTCostCap;
              end
              else
              if (proxy2_pws90 > bp2) and (Small_Correct = 3) and (POU = 0) then
              begin
                InstallPOU := true;
                for i := y to Config.YearsOfAnalysis do
                  system_pou_arr[i] := 1;

                proxy3_pws90[y+1] := bp1/2;
                Variables[fI.b_install_pou] := 1;
                POUInstalled := true;
                POU := 1;
                if pou_install_yr = 0 then pou_install_yr := y;
                Variables[fI.b_pou_study] := 1;
              end;

              if proxy3_pws90[y] = 0 then
                proxy3_pws90[y] := proxy2_pws90;
            end; // end CCT and POU
          end //if Config.SmallSystemFlexibility then
          else
          // SmallSystemFlexibity is false
          begin
{$IFNDEF NOTRIGCCT}
            if (proxy2_pws90 > bp1) and (proxy2_pws90 < bp2) and (CCT = 1) and (not
CCT_Change) and (p_cct_study = 1) then
            begin
              for i := y + 4 to Config.YearsOfAnalysis do
                AdjustCCT[i] := true;
              proxy3_pws90[y+6] := Variables[fi.post_cct_p90_bin2];
              CCT_Change := true;
              if cct_adjust_yr = 0 then cct_adjust_yr := y + 4;

              b_cct_study_rec_mod[y+1] := 1;
              b_cct_study_mod[y+3] := 1;
              b_cct_study_rec_mod_tl[y+1] := 1;
              b_cct_study_mod_tl[y+3] := 1;

              for i := y + 4 to Config.YearsOfAnalysis do
              begin
                b_modify_cct[i] := 1;
                b_modify_cct_tl[i] := 1;
              end;
            end
            else
            if (proxy2_pws90 > bp1) and (proxy2_pws90 < bp2) and (CCT = 1) and (not
CCT_Change) and (p_cct_study = 0) then
            begin
              for i := y + 3 to Config.YearsOfAnalysis do
                AdjustCCT[i] := true;
              proxy3_pws90[y+5] := Variables[fi.post_cct_p90_bin2];
              CCT_Change := true;
              if cct_adjust_yr = 0 then cct_adjust_yr := y + 3;

              b_cct_study_rec_mod[y+1] := 1;
              b_state_cct_treatment_mod[y+2] := 1;

              b_cct_study_rec_mod_tl[y+1] := 1;
              b_state_cct_treatment_mod_tl[y+2] := 1;

              for i := y + 3 to Config.YearsOfAnalysis do
              begin
                b_modify_cct[i] := 1;
                b_modify_cct_tl[i] := 1;
              end;
            end
            else
{$ENDIF}
            if (proxy2_pws90 > bp2) and (CCT = 1) and (not CCT_Change) and
(p_cct_study = 1) then
            begin
              for i := y + 4 to Config.YearsOfAnalysis do
                AdjustCCT[i] := true;
              proxy3_pws90[y+6] := Variables[fi.post_cct_p90_bin1];
              CCT_Change := true;
              if cct_adjust_yr = 0 then cct_adjust_yr := y + 4;

              b_cct_study_rec_mod[y+1] := 1;
              b_cct_study_mod[y+3] := 1;

              b_cct_study_rec_mod_al[y+1] := 1;
              b_cct_study_mod_al[y+3] := 1;
              for i := y + 4 to Config.YearsOfAnalysis do
              begin
                b_modify_cct[i] := 1;
                b_modify_cct_al[i] := 1;
              end;
            end
            else
            if (proxy2_pws90 > bp2) and (CCT = 1) and (not CCT_Change) and 
(p_cct_study = 0) then
            begin
              for i := y + 3 to Config.YearsOfAnalysis do
                AdjustCCT[i] := true;
              proxy3_pws90[y+5] := Variables[fi.post_cct_p90_bin1];
              CCT_Change := true;
              if cct_adjust_yr = 0 then cct_adjust_yr := y + 3;

              b_cct_study_rec_mod[y+1] := 1;
              b_state_cct_treatment_mod[y+2] := 1;

              b_cct_study_rec_mod_al[y+1] := 1;
              b_state_cct_treatment_mod_al[y+2] := 1;

              for i := y + 3 to Config.YearsOfAnalysis do
              begin
                b_modify_cct[i] := 1;
                b_modify_cct_al[i] := 1;
              end;
            end
            else
            if (proxy2_pws90 > bp2) and (CCT = 0) and (not CCT_Change) then
            begin
              for i := max(cct_study_done_yr+2,y+2) to Config.YearsOfAnalysis do
                InstallCCT[i] := true;
              proxy3_pws90[max(cct_study_done_yr+4,y+4)] := 
Variables[fi.post_cct_p90_bin1];
              CCT_Change := true;
              if cct_install_yr = 0 then cct_install_yr := 
max(cct_study_done_yr+2,y+2);

              for i := max(cct_study_done_yr+2,y+2) to Config.YearsOfAnalysis do
                b_install_cct[i] := 1;

              // System Capital Cost undiscounted
              ValuesCapital[2] := ValuesCapital[2] + InstallCCTCostCap;
            end;

            if proxy3_pws90[y] = 0 then
```

```
          proxy3_pws90[y] := proxy2_pws90;
        end;
      end; // end if not VLS
    end; // y >= 4
  end; // end option = 'OW'

  Variables[fI.b_cct_study_rec_install] := b_cct_study_rec_install[y];
  Variables[fI.b_cct_study_install] := b_cct_study_install[y];
  Variables[fI.b_state_cct_treatment_install] := b_state_cct_treatment_install[y];
  Variables[fI.b_cct_study_rec_mod] := b_cct_study_rec_mod[y];
  Variables[fI.b_cct_study_mod] := b_cct_study_mod[y];
  Variables[fI.b_state_cct_treatment_mod] := b_state_cct_treatment_mod[y];
  Variables[fI.b_install_cct] := b_install_cct[y];
  Variables[fI.b_install_cct_mc] := b_install_cct_mc[y];
  Variables[fI.b_modify_cct] := b_modify_cct[y];
  Variables[fI.b_modify_cct_mc] := b_modify_cct_mc[y];
  Variables[fI.b_cct_study_rec_mod_tl] := b_cct_study_rec_mod_tl[y];
  Variables[fI.b_cct_study_mod_tl] := b_cct_study_mod_tl[y];
  Variables[fI.b_modify_cct_tl] := b_modify_cct_tl[y];
  Variables[fI.b_state_cct_treatment_mod_tl] := b_state_cct_treatment_mod_tl[y];
  Variables[fI.b_cct_study_rec_mod_al] := b_cct_study_rec_mod_al[y];
  Variables[fI.b_cct_study_mod_al] := b_cct_study_mod_al[y];
  Variables[fI.b_modify_cct_al] := b_modify_cct_al[y];
  Variables[fI.b_state_cct_treatment_mod_al] := b_state_cct_treatment_mod_al[y];
  Variables[fI.system_pou] := system_pou_arr[y];

  Variables[fI.numb_wqp_sites_added] := numb_wqp_sites_added[y];
  Variables[fI.numb_wqp_sites_added_prev] := numb_wqp_sites_added_prev[y];

  Variables[fI.num_lsl_replace] := 0;
  Variables[fI.hh_remain_lsl] := 0;
  Variables[fI.num_lsl_paper] := 0;

  proxy4_pws90 := -1;
  if (LSL = 1) and not (VLSystem) then
  begin
    if option = 'Baseline' then
    begin
      if y >= 4 then
      begin
        if proxy3_pws90[y] > bp2 then
        begin
          if not lslr_conducted then begin

            // 7% replacement
            num_lsl_replace[y] := min(Num_lsl_remain,
                                      (Num_LSL_base*0.21)*
```

```
(1-(Variables[fI.pp_lcr_test]*Variables[fI.pp_lcr_test_yes]))) 
                                        / 3;
            num_lsl_replace[y+1] := min(Num_lsl_remain,
                                          (Num_LSL_base*0.21)*

(1-(Variables[fI.pp_lcr_test]*Variables[fI.pp_lcr_test_yes])))
                                        / 3;
            num_lsl_replace[y+2] := min(Num_lsl_remain,
                                        (Num_LSL_base*0.21)*

(1-(Variables[fI.pp_lcr_test]*Variables[fI.pp_lcr_test_yes])))
                                      / 3;
            lslr_conducted := true;

          end;
        end;
      end;

        if y >= 4 then
        begin
          if y = 4 then
            hh_remain_lsl := max(0, (num_hh_per_connect * num_lsl_remain));

          if ((Num_lsl_remain > 0) and
             (num_replace >= (Num_LSL_base * 0.21 *
(1-(Variables[fI.pp_lcr_test]*Variables[fI.pp_lcr_test_yes]))))) then
          begin
            proxy4_pws90 := bp1 + ((bp2-bp1)/2);
          end;

          if num_lsl_replace[y] > 0 then
            Variables[fi.b_lslr_mand] := 1
          else
            Variables[fi.b_lslr_mand] := 0;

          if Num_lsl_remain <= 0 then
          begin
            LSL := 0;
            Variables[fi.p_lsl] := 0;
            NewDraw := true;
            Num_lsl_remain := 0;

            proxy4_pws90 := bp1/2;
          end;

          num_replace := 0;
          LSLReplacedMandatory := 0;
          for i := 1 to y do
```

```
            begin
               num_replace := num_replace + num_lsl_replace[i];
               LSLReplacedMandatory := LSLReplacedMandatory + num_lsl_replace[i];
            end;

            Num_lsl_remain := num_LSL_base - LSLReplacedMandatory;
            hh_remain_lsl := max(0, (num_hh_per_connect * num_lsl_remain));

ttLSL:=ttLSL+num_lsl_replace[y];

            Variables[fI.num_lsl_replace] := num_lsl_replace[y];
            Variables[fI.hh_remain_lsl] := hh_remain_lsl;
            num_lsl_paper[y] := ((Num_LSL_base*0.21) *
(Variables[fI.pp_lcr_test]))/3;
            Variables[fI.num_lsl_paper] := num_lsl_paper[y];
          end;
      end // end option = Baseline
      else
      if  (option = 'OW') or (option='OW5L') then begin
        if y >= 4 then
        begin
          if Num_lsl_remain > 0 then
          begin
            if Config.SmallSystemFlexibility then
            begin
              if (SystemType = 1) and (aPop > Config.SmallProxyPop) then
              begin
                if (proxy3_pws90[y] > bp1) and (proxy3_pws90[y] <= bp2) then
                begin
{$IFNDEF NOGOALLCR}
                  Num_lsl_replace[y] := min(pp_lsl_replaced_vol_pct_yr[y] *
(Num_LSL_base + num_paper_remain), Num_lsl_remain);
{$ELSE}
                  Num_lsl_replace[y] := 0;
{$ENDIF}
                  if num_lsl_replace[y] > 0 then
                    Variables[fi.b_lslr_vol] := 1
                  else
                    Variables[fi.b_lslr_vol] := 0;
                end
                else
                if proxy3_pws90[y] > bp2 then
                begin
                  Num_lsl_replace[y] := min(0.03 * (Num_LSL_base +
num_paper_remain), Num_lsl_remain);
                  if num_lsl_replace[y] > 0 then
                    Variables[fi.b_lslr_mand] := 1
                  else
                    Variables[fi.b_lslr_mand] := 0;
                end
                else
                begin
                  Num_lsl_replace[y] := 0;
                  num_lsl_requested[y] := Variables[fI.pp_cust_init_lslr] *
Num_lsl_remain;
                  Variables[fI.b_lslr_requested] := 1;
                end;
              end // if (SystemType = 1) and (aPop > Config.SmallProxyPop) then
              else
              begin
                if (proxy3_pws90[y] > bp2) and (Small_Correct = 1) then
                begin
                  Num_lsl_replace[y] := min(0.07 * (Num_LSL_base +
num_paper_remain), Num_lsl_remain);
                  if num_lsl_replace[y] > 0 then
                    Variables[fi.b_lslr_mand] := 1
                  else
                    Variables[fi.b_lslr_mand] := 0;

                  if lsl_start_yr = 0 then lsl_start_yr := y;
                end
                else
                  Num_lsl_replace[y] := 0;

                if (SystemType = 1) and (proxy3_pws90[y] <= bp2) then
                begin
                  num_lsl_requested[y] := Variables[fI.pp_cust_init_lslr] *
Num_lsl_remain;
                  Variables[fI.b_lslr_requested] := 1;
                end;
              end;
            end // if Config.SmallSystemFlexibility then
            else
            // SmallSystemFlexibity is false
            begin
              if (proxy3_pws90[y] > bp1) and (proxy3_pws90[y] <= bp2) then
              begin
{$IFNDEF NOGOALLCR}
                Num_lsl_replace[y] := min(pp_lsl_replaced_vol_pct_yr[y] *
(Num_LSL_base + num_paper_remain), Num_lsl_remain);
{$ELSE}
                Num_lsl_replace[y] := 0;
{$ENDIF}
                  if num_lsl_replace[y] > 0 then
                    Variables[fi.b_lslr_vol] := 1
                  else
```

```
                    Variables[fi.b_lslr_vol] := 0;
              end
              else
              if proxy3_pws90[y] > bp2 then
              begin
                Num_lsl_replace[y] := min(0.03 * (Num_LSL_base + num_paper_remain),
Num_lsl_remain);
                if num_lsl_replace[y] > 0 then
                  Variables[fi.b_lslr_mand] := 1
                else
                  Variables[fi.b_lslr_mand] := 0;
              end
              else
              begin
                Num_lsl_replace[y] := 0;
                if (SystemType = 1) then
                begin
                  num_lsl_requested[y] := Variables[fI.pp_cust_init_lslr] *
Num_lsl_remain;
                  Variables[fI.b_lslr_requested] := 1;
                end;
              end;
            end;
            proxy4_pws90 := proxy3_pws90[y];
          end
          else
          begin
            LSL := 0;
            Variables[fI.p_lsl] := LSL;
            NewDraw := true;
            Num_lsl_remain := 0;

            ttpws90pct := bp1/2;

            if proxy3_pws90[y] < ttpws90pct then
              proxy4_pws90 := proxy3_pws90[y]
            else
              proxy4_pws90 := ttpws90pct;
          end;

          num_replace := 0;
          num_remain := 0;
          num_requested := 0;
          for i := 1 to y-1 do
          begin
            num_replace := num_replace + Num_lsl_replace[i];
            num_remain := num_remain + num_lsl_paper[i];
            num_requested := num_requested + Num_lsl_requested[i];
          end;

          Num_lsl_remain := Num_LSL_Base - (num_replace + num_requested);
          hh_remain_lsl := max(0, (num_hh_per_connect * num_lsl_remain));

          num_paper_remain := (Num_LSL_Base * Variables[fI.pp_lslr_paper]) - 
num_remain;

ttLSL:=ttLSL+num_lsl_replace[y];

          Variables[fI.num_lsl_replace] := num_lsl_replace[y];
          Variables[fI.hh_remain_lsl] := hh_remain_lsl;
          num_lsl_paper[y] :=  Num_lsl_replace[y] * Variables[fI.pp_lslr_paper];
          Variables[fI.num_lsl_paper] := num_lsl_paper[y];

          if proxy3_pws90[y] > bp2 then
            LSLReplacedMandatory := LSLReplacedMandatory + num_lsl_replace[y]
          else
          if (proxy3_pws90[y] > bp1) and (proxy3_pws90[y] <= bp2) then
            LSLReplacedVoluntary := LSLReplacedVoluntary + num_lsl_replace[y];

          // Failure to meet LSLR Voluntary Program in Bin 2
          if Config.SmallSystemFlexibility then
          begin
            if (SystemType = 1) and (aPop > Config.SmallProxyPop) then
            begin
              if (proxy3_pws90[y] > bp1) and (proxy3_pws90[y] <= bp2) then
              begin
                Meet_LSLR_Goal := 0;
                if Num_LSL_base > 0 then
                begin
                  if (Num_lsl_replace[y]/Num_LSL_base) >= 
Variables[fI.pp_lsl_replaced_vol_goal] then
                    Meet_LSLR_Goal := 1;
                end;
              end;

              if (proxy3_pws90[y] <= bp1) or ((proxy3_pws90[y] > bp1) and 
(proxy3_pws90[y] <= bp2) and (Meet_LSLR_Goal = 1)) then
                NM := 0
              else
                NM := NM + 1;

              Variables[fI.Meet_Lslr_Goal] := Meet_LSLR_Goal;

              if NM = 0 then
              begin
                Variables[fI.fail_nm1] := 0;
```

```
        Variables[fI.fail_nm2] := 0;
       end;

       if NM >= 1 then Variables[fI.fail_nm1] := 1;
       if NM >= 2 then Variables[fI.fail_nm2] := 1;
      end
      else
      begin
       NM := 0;
       Variables[fI.Meet_Lslr_Goal] := 1;
       Variables[fI.fail_nm1] := 0;
       Variables[fI.fail_nm2] := 0;
      end;
     end
     else
     // SmallSystemFlexibity is false
     begin
      if (proxy3_pws90[y] > bp1) and (proxy3_pws90[y] <= bp2) then
      begin
       Meet_LSLR_Goal := 0;
       if Num_LSL_base > 0 then
       begin
        if (Num_lsl_replace[y]/Num_LSL_base) >= 
Variables[fI.pp_lsl_replaced_vol_goal] then
         Meet_LSLR_Goal := 1;
       end;
      end;

      if (proxy3_pws90[y] <= bp1) or ((proxy3_pws90[y] > bp1) and 
(proxy3_pws90[y] <= bp2) and (Meet_LSLR_Goal = 1)) then
       NM := 0
      else
       NM := NM + 1;

      Variables[fI.Meet_Lslr_Goal] := Meet_LSLR_Goal;

      if NM = 0 then
      begin
       Variables[fI.fail_nm1] := 0;
       Variables[fI.fail_nm2] := 0;
      end;

      if NM >= 1 then Variables[fI.fail_nm1] := 1;
      if NM >= 2 then Variables[fI.fail_nm2] := 1;
     end;
    end;
   end; // end option = OW
  end // end has LSL
```

```
    else
    begin
      Variables[fI.fail_nm1] := 0;
      Variables[fI.fail_nm2] := 0;
      Variables[fI.Meet_Lslr_Goal] := 1;
    end;

    if proxy4_pws90 = -1 then
      proxy4_pws90 := proxy3_pws90[y];

    Variables[fI.num_lsl_requested] := num_lsl_requested[y];
    LSLRequested := LSLRequested + num_lsl_requested[y];

    if option = 'Baseline' then
    begin
      if y >= 4 then
      begin
        // proxy4_pws90 > 0 when all LSL replaced
        if proxy4_pws90 > 0 then
          pws90pct := min(proxy4_pws90, proxy3_pws90[y])
        else
          pws90pct := proxy3_pws90[y];
      end;
    end;

    Variables[fI.num_lsl_remain] := num_lsl_remain;
    Variables[fI.num_paper_remain] := num_paper_remain;

    // find and fix
    if ( (option = 'OW') or (option='OW5L')) and (y >= 4) and (not VLSystem) then
    begin
      Num_tap_ge_al := 0;

      if Config.VolLeadProg = 1 then
      begin
        // bin = 3
        if pws90pct <= bp1 then
        begin
          if (Variables[fI.p_tap_nine] = 1) and (y-4 mod 9 = 0) then
            Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al(Variables[fI.numb_reduced_tap],
Variables[fI.pp_above_al_bin_three])
          else
          if (Variables[fI.p_tap_triennial] = 1) and (y-4 mod 3 = 0) then
            Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al(Variables[fI.numb_reduced_tap],
Variables[fI.pp_above_al_bin_three])
          else
          if (Variables[fI.p_tap_annual] = 1) then
            Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al(Variables[fI.numb_reduced_tap],
Variables[fI.pp_above_al_bin_three])
          else
          if (1 - Variables[fI.p_tap_nine] - Variables[fI.p_tap_annual] -
Variables[fI.p_tap_triennial] = 1) then
            Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al((Variables[fI.numb_samp_customer] * 2),
Variables[fI.pp_above_al_bin_three]);
        end
        // bin = 2
        else
        if (pws90pct > bp1) and (pws90pct <= bp2) then
          Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al(Variables[fI.numb_samp_customer],
Variables[fI.pp_above_al_bin_two])
        // bin = 1
        else
          Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al((Variables[fI.numb_samp_customer] * 2),
Variables[fI.pp_above_al_bin_one]);
      end
      else
      begin
        if pws90pct <= bp1 then
        begin
          if (Variables[fI.p_tap_nine] = 1) and (y-4 mod 9 = 0) then
            Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al(Variables[fI.numb_reduced_tap],
Variables[fI.pp_above_al_bin_three])
          else
          if (Variables[fI.p_tap_triennial] = 1) and (y-4 mod 3 = 0) then
            Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al(Variables[fI.numb_reduced_tap],
Variables[fI.pp_above_al_bin_three])
          else
          if (Variables[fI.p_tap_annual] = 1) then
            Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al(Variables[fI.numb_reduced_tap],
Variables[fI.pp_above_al_bin_three])
          else
          if (Variables[fI.p_tap_nine] = 0) and (Variables[fI.p_tap_triennial] = 0)
and (Variables[fI.p_tap_annual] = 0) then
            Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al(Variables[fI.numb_samp_customer]*2,
Variables[fI.pp_above_al_bin_three]);
        end
        else
        if (pws90pct > bp1) and (pws90pct <= bp2) then
          Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al(Variables[fI.numb_samp_customer],
Variables[fI.pp_above_al_bin_two])
        else
          Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al(Variables[fI.numb_samp_customer]*2,
Variables[fI.pp_above_al_bin_one]);
      end;

      if Num_tap_ge_al > 0 then fnf := true;

      if (Num_tap_ge_al = 0) or (b_install_cct[y] + b_modify_cct[y] = 0) then
ff_cct[y] := 0
      else if (Num_tap_ge_al > 0) and (b_install_cct[y] + b_modify_cct[y] > 0) then
      begin
        sumff_cct := 0;
        for i := 1 to y-1 do
           sumff_cct := sumff_cct + ff_cct[i];

        if sumff_cct = 0 then ff_cct[y] := 1
        else if sumff_cct = 1 then ff_cct[y] := 2
        else if sumff_cct = 3 then ff_cct[y] := 3
        else if sumff_cct >= 6 then ff_cct[y] := 4;

        if ff_cct[y] >= 2 then begin
          FindAndFix := true;
          Variables[fI.b_findfix] := 1;
          hFF_CCT := ff_cct[y];
          if hFF_CCt=2 then hFFY2:=Y else
          if hFF_CCt=3 then hFFY3:=Y;
        end;

        if (ff_cct[y] = 4) and (pws90pct > bp2) then
          ff_pws90pct:=Variables[fI.post_ff_p90_bin1] else
        if (ff_cct[y] = 4) and ((pws90pct > bp1) and (pws90pct <= bp2)) then
          ff_pws90pct:=Variables[fI.post_ff_p90_bin2];
      end;
    end;

    if (option = 'OW') or (option = 'OW5L') then
    begin
      if y >= 4 then
      begin
        if ff_pws90pct > 0 then
          pws90pct := min(proxy4_pws90, ff_pws90pct)
        else
```

```
          pws90pct := proxy4_pws90;
        end;
      end;

        if (CCTB = 1) and (b_install_cct[y] + b_modify_cct[y] > 0) then begin
          Variables[fI.cct_modify_cost] := AdjustCCTCostOM + AdjustCCTCostCapDisc -
ExistingCCTCostOM;
          if (Y-cct_adjust_yr) MOD UsefulLifeMod = 0 then
            Variables[fI.cct_modify_cost_umra] := AdjustCCTCostCap
          else
            Variables[fI.cct_modify_cost_umra] := 0;
          Variables[fI.cct_modify_cost_umra_om] := AdjustCCTCostOM -
ExistingCCTCostOM;
          Variables[fI.cct_modify_cost_p] := AdjustCCTCostOM + AdjustCCTCostCapDisc_p
- ExistingCCTCostOM;
        end else begin
          Variables[fI.cct_modify_cost] := 0;
          Variables[fI.cct_modify_cost_umra] := 0;
          Variables[fI.cct_modify_cost_umra_om] := 0;
          Variables[fI.cct_modify_cost_p] := 0;
        end;

        if (CCTB = 0) and (b_install_cct[y] + b_modify_cct[y] > 0) then begin
          Variables[fI.cct_install_cost] := InstallCCTCostOM + InstallCCTCostCapDisc;
          if (Y-cct_install_yr) MOD UsefulLIfeInstall = 0 then
            Variables[fI.cct_install_cost_umra] := InstallCCTCostCap
          else
            Variables[fI.cct_install_cost_umra] := 0;
          Variables[fI.cct_install_cost_umra_om] := InstallCCTCostOM;
          Variables[fI.cct_install_cost_p] := InstallCCTCostOM +
InstallCCTCostCapDisc_p;
        end else begin
          Variables[fI.cct_install_cost] := 0;
          Variables[fI.cct_install_cost_umra] := 0;
          Variables[fI.cct_install_cost_umra_om] := 0;
          Variables[fI.cct_install_cost_p] := 0;
        end;

        Variables[fI.cct_findfix_cost]:=0;
        Variables[fI.cct_findfix_cost_umra]:=0;
        Variables[fI.cct_findfix_cost_umra_om]:=0;
        Variables[fI.cct_findfix_cost_p]:=0;

{the *_p below is the annualizing capital cost for CCT and Find and Fix }

        if fnf and (hFF_CCT = 2) then
        begin
          Variables[fI.cct_findfix_cost] := (Variables[fI.hrs_act_wqp_op] *
  Variables[fI.rate_op]) + Variables[fI.cost_act_wqp];
          Variables[fI.cct_findfix_cost_umra] := (Variables[fI.hrs_act_wqp_op] *
  Variables[fI.rate_op]) + Variables[fI.cost_act_wqp];
          Variables[fI.cct_findfix_cost_umra_om] := 0;
          Variables[fI.cct_findfix_cost_p] := (Variables[fI.hrs_act_wqp_op] *
  Variables[fI.rate_op]) + Variables[fI.cost_act_wqp];
        end
        else
        if (CCTB = 1) and (hFF_CCT = 3) then begin
          Variables[fI.cct_findfix_cost] := (FindAndFixCostOM + FindAndFixCostCapDisc
  - AdjustCCTCostOM) * (1/aEP);
          if (Y-hFFY2) MOD UsefulLIfeFF = 0 then
            Variables[fI.cct_findfix_cost_umra] := (FindAndFixCostCap) * (1/aEP)
          else
            Variables[fI.cct_findfix_cost_umra] := 0;
          Variables[fI.cct_findfix_cost_umra_om] := (FindAndFixCostOM -
  AdjustCCTCostOM) * (1/aEP);
          Variables[fI.cct_findfix_cost_p] := (FindAndFixCostOM +
  FindAndFixCostCapDisc_p - AdjustCCTCostOM) * (1/aEP);
        end else if (CCTB = 0) and (hFF_CCT = 3) then begin
          Variables[fI.cct_findfix_cost] := (FindAndFixCostOM + FindAndFixCostCapDisc
  - InstallCCTCostOM) * (1/aEP);
          if (Y-hFFY2) MOD UsefulLIfeFF = 0 then
            Variables[fI.cct_findfix_cost_umra] := (FindAndFixCostCap) * (1/aEP)
          else
            Variables[fI.cct_findfix_cost_umra] := 0;
          Variables[fI.cct_findfix_cost_umra_om] := (FindAndFixCostOM -
  InstallCCTCostOM) * (1/aEP);
          Variables[fI.cct_findfix_cost_p] := (FindAndFixCostOM +
  FindAndFixCostCapDisc_p - InstallCCTCostOM) * (1/aEP);
        end else if (CCTB = 1) and (hFF_CCT >= 4) then begin
          Variables[fI.cct_findfix_cost] := (FindAndFixCostOM + FindAndFixCostCapDisc
  - AdjustCCTCostOM);
          if (Y-hFFY3) MOD UsefulLIfeFF = 0 then
            Variables[fI.cct_findfix_cost_umra] := (FindAndFixCostCap)
          else
            Variables[fI.cct_findfix_cost_umra] := 0;
          Variables[fI.cct_findfix_cost_umra_om] := (FindAndFixCostOM -
  AdjustCCTCostOM);
          Variables[fI.cct_findfix_cost_p] := (FindAndFixCostOM +
  FindAndFixCostCapDisc_p - AdjustCCTCostOM);
        end else if (CCTB = 0) and (hFF_CCT >= 4) then begin
          Variables[fI.cct_findfix_cost] := (FindAndFixCostOM + FindAndFixCostCapDisc
  - InstallCCTCostOM);
          if (Y-hFFY3) MOD UsefulLIfeFF = 0 then
            Variables[fI.cct_findfix_cost_umra] := (FindAndFixCostCap)
          else
            Variables[fI.cct_findfix_cost_umra] := 0;
          Variables[fI.cct_findfix_cost_umra_om] := (FindAndFixCostOM -
  InstallCCTCostOM);
          Variables[fI.cct_findfix_cost_p] := (FindAndFixCostOM +
  FindAndFixCostCapDisc_p - InstallCCTCostOM);
        end;

        if hFF_CCT >= 2 then
          HasFindAndFixCost := true;

      Variables[fI.pbaseph] := CCTCostEquations.pbaseph;
      Variables[fI.pbasepo4] := CCTCostEquations.pbasepo4;
      Variables[fI.pbasephpo4] := CCTCostEquations.pbasephpo4;

      // if non-blank pwsid is passed to function, collect data for output debug files
      if prtDebug then
      begin
        i := 0;
        for CV in fCostVars.Values do
        begin
          sLine := CV.fID + chr(9) + Y.ToString + chr(9) + Variables[i].ToString +
  chr(9) + RawVariables[i].ToString;
          slVariables.Add(sLine);
          Inc(i);
        end;

        if y >= 4 then
        begin
          sLine := 'pp_lsl_replacement_rates' + chr(9) + Y.ToString + chr(9) +
  pp_lsl_replacement_rates[y].ToString;
          slVariables.Add(sLine);
        end;

        sLine := 'p_source_chng_yr' + chr(9) + Y.ToString + chr(9) +
  p_source_chng_yr[y].ToString;
        slVariables.Add(sLine);
        sLine := 'p_treat_change_yr' + chr(9) + Y.ToString + chr(9) +
  p_treat_change_yr[y].ToString;
        slVariables.Add(sLine);
        sLine := 'pp_lsl_replaced_vol_pct_yr' + chr(9) + Y.ToString + chr(9) +
  pp_lsl_replaced_vol_pct_yr[y].ToString;
        slVariables.Add(sLine);
        sLine := 'owBin' + chr(9) + Y.ToString + chr(9) + owBin.ToString;
        slVariables.Add(sLine);
        sLine := 'pws90pct' + chr(9) + Y.ToString + chr(9) + pws90pct.ToString;
        slVariables.Add(sLine);

        sLine2 := LSL.ToString + chr(9) + Num_LSL_base.ToString
                               + chr(9) + Y.ToString
```

```
                              + chr(9) + pp_lsl_replacement_rates[y].ToString
                              + chr(9) + num_replace.ToString
                              + chr(9) + num_lsl_remain.ToString
                              + chr(9) + hh_remain_lsl.ToString
                              + chr(9) + Num_lsl_replace[y].ToString
                              + chr(9) + Meet_LSLR_Goal.ToString
                              + chr(9) + NM.ToString
                              + chr(9) + Num_lsl_replace_ale[y].ToString
                              + chr(9) + pp_lslr_paper.ToString
                              + chr(9) + replace_rate.ToString
                              ;
      slCalcs.Add(sLine2);


      sLine2 := Y.ToString + chr(9) + aPop.ToString + chr(9) +
                ExistingCCTCostOM.ToString + chr(9) +
                AdjustCCTCostOM.ToString + chr(9) +
                InstallCCTCostOM.ToString + chr(9) + InstallCCTCostCapDisc.ToString
+ chr(9) +
                FindAndFixCostOM.ToString + chr(9) +
                CCTCostEquations.iBaselinepo4dose.ToString + chr(9) +
                CCTCostEquations.iBaselineph_wph.ToString + chr(9) +
                CCTCostEquations.iBaselineph_woph.ToString + chr(9) +
                CCTCostEquations.pbaseph.ToString + chr(9) +
                CCTCostEquations.pbasepo4.ToString + chr(9) +
                CCTCostEquations.pbasephpo4.ToString + chr(9) +
                CCTCostEquations.DFlowEP.ToString + chr(9) +
                CCTCostEquations.AFlowEP.ToString + chr(9) +
                CCTCostEquations.EntryPoints.ToString;
      slCCTCosts.Add(sLine2);
    end;




    if pws90pct > bp2 then owBin_tmp := 1
    else if (pws90pct > bp1) and (pws90pct <= bp2) then owBin_tmp := 2
    else if pws90pct <= bp1 then owBin_tmp := 3;

    pws90pctCCT_yr[y] := proxy2_pws90;
    pws90pctLSL_yr[y] := proxy3_pws90[y];

    // tpws90pct is set above if there was a source water or treatment change
    if tpws90pct < proxy1_pws90 then
    begin
      owbin := owBin_tmp;
    end
```

```
else
if owBin_tmp > owbin then
begin
  owbin := owBin_tmp;
end
else
begin
  owBin := owBin;
end;

if _UseCompiled then begin
  CC._Evaluate(y);
end;

// Compute Costs
for ic := 0 to CostSteps.Count - 1 do begin
  C := CostSteps.Items[ic];
//for C in CostSteps do begin
  // only calculate cost in appropriate year
  if not C.DirArrCalculateYr[Y] then continue;
  // if very large system don't calculate ep level costs in this loop
  if VLSystem and C.fCostStepRec.VLSEpLevel then continue;

  if not (((owBin = 1) and (C.fCostStepRec.Bin1 = 1)) or
          ((owBin = 2) and (C.fCostStepRec.Bin2 = 1)) or
          ((owBin = 3) and (C.fCostStepRec.Bin3 = 1))) then continue;

  if SystemType = 1 then
  begin
    if C.fCostStepRec.IncludeCost = 'NTNCWS' then
      continue;
  end
  else
  if SystemType = 2 then
  begin
    if C.fCostStepRec.IncludeCost = 'CWS' then
      continue;
  end;


  if _UseCompiled then begin
    Cost := CC._Cost[C.fCostStepRec.ID];
    Labor := CC._Labor[C.fCostStepRec.ID];
    OM := CC._OM[C.fCostStepRec.ID];
    Hours := CC._Hours[C.fCostStepRec.ID];
  end else
    C.Evaluate(Cost,Labor,OM,Hours,DoIt);
```

```
// only do the following for CCT cost variables in hopes of saving processing
// this is for determining CCT install or adjust event
if not VLSystem then
begin
    if Cost > 0 then
    begin
      HasCCTCost := true;

      if (C.fCostStepRec.Frequency = 'Once') then
      begin
        for yy := y+1 to fYears do C.arrCalculateYr[yy] := false;
      end;

      if option = 'Baseline' then
      begin
        if C.BaselineCCTModify then
        begin
          ExistingCCT := false;
          CCTAdjusted := true;
          CCTAdjusted_ale := true;
          CCTAdjusted_tle := false;
        end
        else if C.BaselineCCTInstall then
        begin
          CCT := 1;
          NewCCT := true;
          NewDraw := true;
          CCTInstalled := true;
        end;
      end
      else if  (option = 'OW') or (option='OW5L') then
      begin
        if C.OWCCTModify then
        begin
          ExistingCCT := false;
          CCTAdjusted := true;
          if C.OWCCTModify_ale then CCTAdjusted_ale := true;
          if C.OWCCTModify_tle then CCTAdjusted_tle := true;
        end
        else if C.OWCCTInstall then
        begin
          CCT := 1;
          NewCCT := true;
          NewDraw := true;
          CCTInstalled := true;
        end;
      end;
    end;
```

```
      end;

      if not HasLSLRCost then
      begin
        HASLSLRcost:= (Cost > 0) and (C.fCostStepRec.LSLRCost);
      end;

      if C.SysLSLRCapital then
        ValuesCapital[0] := ValuesCapital[0] + Cost;

      if C.HhLSLRCapital then
        ValuesCapital[1] := ValuesCapital[1] + Cost;

      // these costs should incur only once

      if c.fAgg2ID > -1 then begin
        Values2[c.fAgg2ID]:=Values2[c.fAgg2ID]+Discount(Cost,Y-1,-1);
        Values2p[c.fAgg2ID]:=Values2p[c.fAgg2ID]+Discount(Cost,Y-1,CostCapital);
        Values2Y[y,c.fAgg2ID] := Values2Y[y,c.fAgg2ID] + Cost;
      end;
      if c.fAgg2IDH > -1 then begin
        Values2[c.fAgg2IDH]:=Values2[c.fAgg2IDH]+Hours;
        Values2p[c.fAgg2IDH]:=Values2p[c.fAgg2IDH]+Hours;
      end;
      if c.fAgg2IDL > -1 then begin
        Values2[c.fAgg2IDL]:=Values2[c.fAgg2IDL]+Discount(Labor,Y-1,-1);
        Values2p[c.fAgg2IDL]:=Values2p[c.fAgg2IDL]+Discount(Labor,Y-1,CostCapital);
        Values2Y[y,c.fAgg2IDL] := Values2Y[y,c.fAgg2IDL] + Labor;
      end;
      if c.fAgg2IDO > -1 then begin
        Values2[c.fAgg2IDO]:=Values2[c.fAgg2IDO]+Discount(OM,Y-1,-1);
        Values2p[c.fAgg2IDO]:=Values2p[c.fAgg2IDO]+Discount(OM,Y-1,CostCapital);
        Values2Y[y,c.fAgg2IDO] := Values2Y[y,c.fAgg2IDO] + OM;
      end;

      // aggregate ICR categories
      if Y = 1 then begin
        if C.fAggICR_IDC1 > -1 then ValuesICR[C.fAggICR_IDC1] :=
ValuesICR[C.fAggICR_IDC1] + OM;
        if C.fAggICR_IDH1 > -1 then ValuesICR[C.fAggICR_IDH1] :=
ValuesICR[C.fAggICR_IDH1] + Hours;
      end
      else if Y = 2 then begin
        if C.fAggICR_IDC2 > -1 then ValuesICR[C.fAggICR_IDC2] :=
ValuesICR[C.fAggICR_IDC2] + OM;
        if C.fAggICR_IDH2 > -1 then ValuesICR[C.fAggICR_IDH2] :=
ValuesICR[C.fAggICR_IDH2] + Hours;
      end
      else if Y = 3 then begin
        if C.fAggICR_IDC3 > -1 then ValuesICR[C.fAggICR_IDC3] :=
ValuesICR[C.fAggICR_IDC3] + OM;
        if C.fAggICR_IDH3 > -1 then ValuesICR[C.fAggICR_IDH3] :=
ValuesICR[C.fAggICR_IDH3] + Hours;
      end
      else if ((Y >= 4) and (Y <= 9)) then begin
        if C.fAggICR_IDC4 > -1 then ValuesICR[C.fAggICR_IDC4] :=
ValuesICR[C.fAggICR_IDC4] + OM;
        if C.fAggICR_IDH4 > -1 then ValuesICR[C.fAggICR_IDH4] :=
ValuesICR[C.fAggICR_IDH4] + Hours;
      end
      else if (Y >= 10) and (Y <= fYearsOutput) then begin
        if C.fAggICR_IDC10 > -1 then ValuesICR[C.fAggICR_IDC10] :=
ValuesICR[C.fAggICR_IDC10] + OM;
        if C.fAggICR_IDH10 > -1 then ValuesICR[C.fAggICR_IDH10] :=
ValuesICR[C.fAggICR_IDH10] + Hours;
      end;

      // if non-blank pwsid is passed to function, collect data for output debug
files
      if prtDebug then
      begin
        if C.fCostStepRec.TotalCost <> '' then
        begin
          sLine := owBin.ToString + chr(9) +
                   C.fCostStepRec.CostName + chr(9) + Y.ToString + chr(9) + 'cost' +
chr(9) +
                   C.fCostStepRec.Agglomerator2Xw + chr(9) +
C.fCostStepRec.TotalCost + chr(9) +
                   Discount(Cost,Y-1,-1).ToString + chr(9) + Cost.ToString + chr(9)
+
                   C.fCostStepRec.Bin1.ToString + chr(9) +
C.fCostStepRec.Bin2.ToString + chr(9) +
                   C.fCostStepRec.Bin3.ToString;
          slCosts.Add(sLine);
        end;
        if C.fCostStepRec.Hours <> '' then
        begin
          sLine := owBin.ToString + chr(9) +
                   C.fCostStepRec.CostName + chr(9) + Y.ToString + chr(9) + 'hours'
+ chr(9) +
                   C.fCostStepRec.Agglomerator2Xw + chr(9) + C.fCostStepRec.Hours +
chr(9) +
                   Hours.ToString + chr(9) + '0' + chr(9) +
                   C.fCostStepRec.Bin1.ToString + chr(9) +
C.fCostStepRec.Bin2.ToString + chr(9) +
                   C.fCostStepRec.Bin3.ToString;
          slCosts.Add(sLine);
        end;
        if C.fCostStepRec.Labor <> '' then
        begin
          sLine := owBin.ToString + chr(9) +
                   C.fCostStepRec.CostName + chr(9) + Y.ToString + chr(9) + 'labor'
+ chr(9) +
                   C.fCostStepRec.Agglomerator2Xw + chr(9) + C.fCostStepRec.Labor +
chr(9) +
                   Discount(Labor,Y-1,-1).ToString + chr(9) +
Discount(Labor,Y-1,CostCapital).ToString + chr(9) +
                   C.fCostStepRec.Bin1.ToString + chr(9) +
C.fCostStepRec.Bin2.ToString + chr(9) +
                   C.fCostStepRec.Bin3.ToString;
          slCosts.Add(sLine);
        end;
        if C.fCostStepRec.OM <> '' then
        begin
          sLine := owBin.ToString + chr(9) +
                   C.fCostStepRec.CostName + chr(9) + Y.ToString + chr(9) + 'om' +
chr(9) +
                   C.fCostStepRec.Agglomerator2Xw + chr(9) + C.fCostStepRec.OM +
chr(9) +
                   Discount(OM,Y-1,-1).ToString + chr(9) + OM.ToString + chr(9) +
                   C.fCostStepRec.Bin1.ToString + chr(9) +
C.fCostStepRec.Bin2.ToString + chr(9) +
                   C.fCostStepRec.Bin3.ToString;
          slCosts.Add(sLine);
        end;
      end;
    end;  // end C in CostSteps loop
    // end Compute Costs

    // POTW cost
    if not VLSystem then
    begin
      if CCTCostEquations.pbasepo4 = 1 then
      begin
        if (AdjustCCT[y] and (cct_adjust_yr = Y)) or
           (InstallCCT[y] and (cct_install_yr = Y)) then
          prob_downstream_P_limit :=  calc_prob_downstream_P_limit(isBaseline, Y);

        if prob_downstream_P_limit = 1 then
        begin
          PDose := CCTCostEquations.arrBaselineP[CCTCostEquations.iBaselinepo4dose];
          FlowLossP := (CCTCostEquations.AFlowEP*CCTCostEquations.EntryPoints) *
PDose * 10893.71;
          ConnectionLossP := aNC * PDose * 0.86;
          POTWCost := POTWCost + Discount((FlowLossP - 
ConnectionLossP),Y-1,CostCapital);
        end;
      end;

      if (CCTB = 1) and (CCTCostEquations.pbasepo4 + CCTCostEquations.pbasephpo4 > 
0) then
      begin
        if Y = 5 then
          prerule_ploading_lbs_5 := (0.775 * 
CCTCostEquations.arrBaselinepo4dose[CCTCostEquations.iBaselinepo4dose] * 
CCTCostEquations.AFlowEP * 1000) -
                                    (0.061 * aNC * 
CCTCostEquations.arrBaselinepo4dose[CCTCostEquations.iBaselinepo4dose])
        else
        if Y = 15 then
          prerule_ploading_lbs_15 := (0.775 * 
CCTCostEquations.arrBaselinepo4dose[CCTCostEquations.iBaselinepo4dose] * 
CCTCostEquations.AFlowEP * 1000) -
                                     (0.061 * aNC * 
CCTCostEquations.arrBaselinepo4dose[CCTCostEquations.iBaselinepo4dose])
        else
        if Y = 25 then
          prerule_ploading_lbs_25 := (0.775 * 
CCTCostEquations.arrBaselinepo4dose[CCTCostEquations.iBaselinepo4dose] * 
CCTCostEquations.AFlowEP * 1000) -
                                     (0.061 * aNC * 
CCTCostEquations.arrBaselinepo4dose[CCTCostEquations.iBaselinepo4dose])
        else
        if Y = 35 then
          prerule_ploading_lbs_35 := (0.775 * 
CCTCostEquations.arrBaselinepo4dose[CCTCostEquations.iBaselinepo4dose] * 
CCTCostEquations.AFlowEP * 1000) -
                                     (0.061 * aNC * 
CCTCostEquations.arrBaselinepo4dose[CCTCostEquations.iBaselinepo4dose]);
      end;

      if (b_install_cct[y] + b_modify_cct[y] > 0) and (CCTCostEquations.pbasepo4 + 
CCTCostEquations.pbasephpo4 > 0) then
      begin
        if Y = 5 then
          postrule_ploading_lbs_5 := (0.775 * 3.2 * CCTCostEquations.AFlowEP * 1000) 
-
                                     (0.061 * aNC * 3.2)
        else
        if Y = 15 then
          postrule_ploading_lbs_15 := (0.775 * 3.2 * CCTCostEquations.AFlowEP * 
1000) -
```

```
                                (0.061 * aNC * 3.2)
        else
        if Y = 25 then
          postrule_ploading_lbs_25 := (0.775 * 3.2 * CCTCostEquations.AFlowEP *
1000) -
                                (0.061 * aNC * 3.2)
        else
        if Y = 35 then
          postrule_ploading_lbs_35 := (0.775 * 3.2 * CCTCostEquations.AFlowEP *
1000) -
                                (0.061 * aNC * 3.2);
      end
      else
      begin
        if Y = 5 then
          postrule_ploading_lbs_5 := prerule_ploading_lbs_5
        else
        if Y = 15 then
          postrule_ploading_lbs_15 := prerule_ploading_lbs_15
        else
        if Y = 25 then
          postrule_ploading_lbs_25 := prerule_ploading_lbs_25
        else
        if Y = 35 then
          postrule_ploading_lbs_35 := prerule_ploading_lbs_35;
      end;

      incr_ploading_lbs_5 := postrule_ploading_lbs_5 - prerule_ploading_lbs_5;
      incr_ploading_lbs_15 := postrule_ploading_lbs_15 - prerule_ploading_lbs_15;
      incr_ploading_lbs_25 := postrule_ploading_lbs_25 - prerule_ploading_lbs_25;
      incr_ploading_lbs_35 := postrule_ploading_lbs_35 - prerule_ploading_lbs_35;

      if incr_ploading_lbs_5 > 0 then
        count_incr_ploading_lbs_5 := 1;
      if incr_ploading_lbs_15 > 0 then
        count_incr_ploading_lbs_15 := 1;
      if incr_ploading_lbs_25 > 0 then
        count_incr_ploading_lbs_25 := 1;
      if incr_ploading_lbs_35 > 0 then
        count_incr_ploading_lbs_35 := 1;
    end;

    // read data request data values from database for next year
    fCostVars.FillValueArray(Variables, RawVariables, aSz, aSrc, LSL, CCT, aType,
Y+1, Config.PWS90PctBp1, Config.PWS90PctBp2, SetProbsTo01, NewDraw,
                             O, isBaseline);

    Variables[fI.p_lsl] := LSL;
```

```
// load external variables values
Variables[fI.EP] := aEP;
Variables[fI.Pws_Cct] := CCT;
Variables[fI.Pws_first_ale] := aFirstAle;

Variables[fI.Pws_sw] := 0;
Variables[fI.Pws_gw] := 0;
if aSrc = 2 then
  Variables[fI.Pws_sw] := 1
else if aSrc = 1 then
  Variables[fI.Pws_gw] := 1;

Variables[fI.Pws_pop] := aPop;

InitCCTBVarsToZero(option);

if FindAndFix then
  Variables[fI.b_findfix] := 1;

if  (option <> 'Baseline') then
begin
  if Num_Proxies = 0 then
  begin
    if Round(Config.DiscountRate*100)/100 = 0.03 then
      Variables[fI.annual_pou_cost_hh] := 111
    else if Round(Config.DiscountRate*100)/100 = 0.07 then
      Variables[fI.annual_pou_cost_hh] := 114
    else
      Variables[fI.annual_pou_cost_hh] := -1;
  end
  else
    Variables[fI.annual_pou_cost_hh] := 114;

  if Config.VolLeadProg = 0 then
    Variables[fI.p_vol_leadtap_prog] := 0;
end;

if not VLSystem then
begin
  if AdjustCCT[y] then fAdjust_CCT := 1;
  if InstallCCT[y] then fInstall_CCT := 1;

  if fAdjust_CCT = 0 then
    partial_cct_level := 0
  else
  begin
    if CCTCostEquations.pbaseph = 1 then
```

```
        begin
          if CCTCostEquations.iBaselineph_wph <= 1 then
            partial_cct_level := 1
          else
          if (CCTCostEquations.iBaselineph_wph >= 2) and
(CCTCostEquations.iBaselineph_wph <= 3) then
            partial_cct_level := 2
          else
            partial_cct_level := 3;
        end
        else
        begin
          if CCTCostEquations.iBaselinepo4dose = 1 then
            partial_cct_level := 1
          else
          if CCTCostEquations.iBaselinepo4dose = 2 then
            partial_cct_level := 2
          else
            partial_cct_level := 3;
        end;
      end;

      bCCT_Change := (b_install_cct[y] + b_modify_cct[y] > 0);
      if isBaseline then
        LeadConcentrationBins(pwsid, option, y, aSz, aSrc, LSL, CCT, POU, aPop, aNC,
                              fAdjust_CCT, fInstall_CCT, cct_adjust_yr,
cct_install_yr, pou_install_yr,
                              bCCT_Change, pwswgt, num_lsl_replace[y],
num_lsl_requested[y], num_lsl_remain, Num_Proxies, partial_cct_level,
                              hp_lslr_paper, hp_lslr_partial,
pp_lsl_replacement_rates,false)
      else
        LeadConcentrationBins(pwsid, option, y, aSz, aSrc, LSL, CCT, POU, aPop, aNC,
                              fAdjust_CCT, fInstall_CCT, cct_adjust_yr,
cct_install_yr, pou_install_yr,
                              bCCT_Change, pwswgt, num_lsl_replace[y],
num_lsl_requested[y], num_lsl_remain, Num_Proxies, partial_cct_level,
                              hp_lslr_paper, hp_lslr_partial,
pp_lsl_replaced_vol_pct_yr,false)
    end;

    if Num_Proxies = 0 then
    begin
      Config.PWSBinCount[aType, y, aSz, aSrc, owBin] := Config.PWSBinCount[aType, y,
aSz, aSrc, owBin] + pwswgt;
    end;

    //csl('pwsid, y: ' + pwsid + ' ' + y.ToString);
```

```
  end; // end year loop

  LSLReplaced := LSLReplacedMandatory + LSLReplacedVoluntary;
  if Round(ttLSL)<>Round(LSLReplaced) then
    CodeSite.SendFmtMsg('Mismatch LSLReplaced: %f,  Sum: %f',[LSLReplaced,ttLSL]);

  if not VLSystem then
    Annualize(CostCapital);

  // write debug files
  if prtDebug then
  begin
//    slVariables.SaveToFile(userpath + 'Vars_' + pwsid + '_' + option + '.tab');
//    slCosts.SaveToFile(userpath + 'Costs_' + pwsid + '_' + option + '.tab');;
//    slCalcs.SaveToFile(userpath + 'Calcs_' + pwsid + '_' + option + '.tab');;
//    slCCTCosts.SaveToFile(userpath + 'CCTCosts_' + pwsid + '_' + option +
'.tab');;
  end;
end;

procedure TCostingSteps.SetVariablesAndCalculateVLS(const CostingData: TCostGenRec;
  const AddCostingData: TAddCostGenRec;
  const VLSEpWorkbook: TVLSEpWorkbookRec; const SetProbsTo01: boolean; const option:
string;
  const CCTCostEquations: TCCTCostEquations;
  const O: TDictionary<string, double>; const SchoolSampData: TSchoolSampDataRec;
const ResetBin : boolean=false);
var C : TCostingStep;
    Cost,Labor,OM,Hours,DoIt,V : double;
    Y : integer;
    NewDraw : boolean;
    CCT, LSL, POU : integer;
    UsefulLifeInstall,UsefulLifeMod,UsefulLifeFF : integer;
    ExistingCCT, NewCCT, FindAndFix, InstallPOU: boolean;
    ExistingCCTCostOM, AdjustCCTCostOM, InstallCCTCostOM, InstallCCTCostCap,
InstallCCTCostCapDisc,
    InstallCCTCostCapDisc_p, FindAndFixCostOM : double;
    AdjustCCTCost, AdjustCCTOM, NewCCTCost, NewCCTOM: double;
    FindAndFixCostCap, FindAndFixCostCapDisc, AdjustCCTCostCap,
AdjustCCTCostCapDisc: double;
    AdjustCCTCostCapDisc_p, FindAndFixCostCapDisc_p: double;

    Num_LSL_base: double;
    pp_lsl_replaced, num_lsl_replace, lslr_missed_goal, num_lsl_requested :
array[1..100] of double;
    Num_lsl_replace_ale: array[1..100] of double;
       // go out 51 years to avoid array overrun in year 49
    Num_lsl_remain: double;
    Meet_LSLR_Goal: integer;
    num_replace, num_remain, num_requested: double;
    NM: integer;
    num_hh_per_connect: double;
    hh_remain_lsl: double;
    lslr_conducted: boolean;
    pp_lsl_replacement_rates: array[0..100] of double;
    failCost1, failCost4, failCost5, failCost6, failCost7, failCost8: boolean;

    CV: TCostVar;
    i: integer;
    sLine, sLine2: string;
    yy: integer;

    isBaseline: boolean;

    cct_adjust_yr, cct_install_yr, pou_install_yr: integer;

    pp_lslr_paper, num_lsl_base_adjust, num_replace_paper, num_replace_paper2:
double;
    num_paper_remain: double;
    partial_cct_level: integer;
    hp_lslr_paper: array [1..3] of double;
    hp_lslr_partial: array [1..3] of double;

    prob_downstream_P_limit: integer;
    PDose, FlowLossP, ConnectionLossP: double;

    replace_rate: double;

    owBin, tBin, owBin_tmp: integer;
    pws90pct, tpws90pct, ttpws90pct: double;
    proxy1_pws90, proxy2_pws90, proxy4_pws90: double;
    CCT_Change, bCCT_Change: boolean;
    bp1, bp2: integer;

    tmp_double: double;
    p_source_chng_yr: array[0..100] of integer;
    p_source_sig_yr: array[0..100] of integer;
    p_treat_change_yr: array[0..100] of integer;
    pp_lsl_replaced_vol_pct_yr: array[0..100] of double;
    pp_lsl_replaced_vol_actual: double;

    Num_tap_ge_al: double;
    fnf: boolean;
    ff_cct: array[0..100] of integer;
    sumff_cct, hff_cct: integer;
    hffY2, hFFY3 : integer;
    CCTB, LSLB: integer;
    BinChgLsl, BinChgNoLsl: array[0..100] of integer;

    b_cct_study_rec_install: array[0..100] of integer;
    b_cct_study_install: array[0..100] of integer;
    b_state_cct_treatment_install: array[0..100] of integer;
    b_cct_study_rec_mod: array[0..100] of integer;
    b_cct_study_mod: array[0..100] of integer;
    b_state_cct_treatment_mod: array[0..100] of integer;
    b_install_cct: array[0..100] of integer;
    b_install_cct_mc: array[0..100] of integer;
    p_cct_study: integer;
    proxy3_pws90: array[0..100] of double;
    InstallCCT: array[0..100] of boolean;
    AdjustCCT: array[0..100] of boolean;
    cct_study_done_yr: integer;
    b_modify_cct: array[0..100] of integer;
    b_modify_cct_mc: array[0..100] of integer;
    ff_pws90pct: double;
    b_cct_study_rec_mod_tl: array[0..100] of integer;
    b_cct_study_mod_tl: array[0..100] of integer;
    b_modify_cct_tl: array[0..100] of integer;
    b_state_cct_treatment_mod_tl: array[0..100] of integer;
    b_cct_study_rec_mod_al: array[0..100] of integer;
    b_cct_study_mod_al: array[0..100] of integer;
    b_modify_cct_al: array[0..100] of integer;
    b_state_cct_treatment_mod_al: array[0..100] of integer;
    system_pou_arr: array[0..100] of integer;

    numb_wqp_add_sites: array[0..100] of double;
    numb_wqp_add_sites_total: array[0..100] of double;
    numb_wqp_sites_added: array[0..100] of double;
    numb_wqp_sites_added_prev: array[0..100] of double;
    num_lsl_paper: array[0..100] of double;

    SourceTreatChangeEver: boolean;
begin
{$R+}

  fnum_proxies := 0;

  fillchar(pws90pctCCT_yr, SizeOf(pws90pctCCT_yr), 0);
  fillchar(pws90pctLSL_yr, SizeOf(pws90pctLSL_yr), 0);

  LSL := VLSEpWorkbook.LSL;
  CCT := VLSEpWorkbook.CCT;
  POU := 0;
```

```
CCTB := CCT;
LSLB := LSL;

for i := 0 to Config.YearsOfAnalysis do
begin
  InstallCCT[i] := false;
  AdjustCCT[i] := false;
end;

ExistingCCT := false;
NewCCT := false;
InstallPOU := false;
FindAndFix := false;
NewDraw := true;

ExistingCCTCostOM:=0;
AdjustCCTCostOM:=0;
InstallCCTCostOM:=0;
InstallCCTCostCap:=0;
InstallCCTCostCapDisc:=0;
InstallCCTCostCapDisc_p:=0;
FindAndFixCostOM:=0;
FindAndFixCostCap:=0;
FindAndFixCostCapDisc:=0;
AdjustCCTCostCap:=0;
AdjustCCTCostCapDisc:=0;
AdjustCCTCostCapDisc_p:=0;
FindAndFixCostCapDisc_p:=0;

CCTInstalled := false;
CCTAdjusted := false;
CCTAdjusted_ale := false;
CCTAdjusted_tle := false;
CCTExisting := false;
HasFindAndFixCost := false;
POUInstalled := false;

AdjustCCTCost:=0;
AdjustCCTOM:=0;
NewCCTCost:=0;
NewCCTOM:=0;

cct_adjust_yr := 0;
cct_install_yr := 0;
pou_install_yr := 0;
partial_cct_level := 0;
```

```
prob_downstream_P_limit := -1;

POTWCost := 0;
prerule_ploading_lbs_5 := 0;
prerule_ploading_lbs_15 := 0;
prerule_ploading_lbs_25 := 0;
prerule_ploading_lbs_35 := 0;
postrule_ploading_lbs_5 := 0;
postrule_ploading_lbs_15 := 0;
postrule_ploading_lbs_25 := 0;
postrule_ploading_lbs_35 := 0;
incr_ploading_lbs_5 := 0;
incr_ploading_lbs_15 := 0;
incr_ploading_lbs_25 := 0;
incr_ploading_lbs_35 := 0;
count_incr_ploading_lbs_5 := 0;
count_incr_ploading_lbs_15 := 0;
count_incr_ploading_lbs_25 := 0;
count_incr_ploading_lbs_35 := 0;

fillchar(b_cct_study_rec_install,SizeOf(b_cct_study_rec_install),0);
fillchar(b_cct_study_install,SizeOf(b_cct_study_install),0);
fillchar(b_state_cct_treatment_install,SizeOf(b_state_cct_treatment_install),0);
fillchar(b_cct_study_rec_mod,SizeOf(b_cct_study_rec_mod),0);
fillchar(b_cct_study_mod,SizeOf(b_cct_study_mod),0);
fillchar(b_state_cct_treatment_mod,SizeOf(b_state_cct_treatment_mod),0);
fillchar(b_install_cct,SizeOf(b_install_cct),0);
fillchar(b_install_cct_mc,SizeOf(b_install_cct_mc),0);
fillchar(proxy3_pws90,SizeOf(proxy3_pws90),0);
cct_study_done_yr := 0;
fillchar(b_modify_cct,SizeOf(b_modify_cct),0);
fillchar(b_modify_cct_mc,SizeOf(b_modify_cct_mc),0);
ff_pws90pct := 0;
fillchar(b_cct_study_rec_mod_tl, SizeOf(b_cct_study_rec_mod_tl), 0);
fillchar(b_cct_study_mod_tl, SizeOf(b_cct_study_mod_tl), 0);
fillchar(b_modify_cct_tl, SizeOf(b_modify_cct_tl), 0);
fillchar(b_state_cct_treatment_mod_tl, SizeOf(b_state_cct_treatment_mod_tl), 0);
fillchar(b_cct_study_rec_mod_al, SizeOf(b_cct_study_rec_mod_al), 0);
fillchar(b_cct_study_mod_al, SizeOf(b_cct_study_mod_al), 0);
fillchar(b_modify_cct_al, SizeOf(b_modify_cct_al), 0);
fillchar(b_state_cct_treatment_mod_al, SizeOf(b_state_cct_treatment_mod_al), 0);
fillchar(system_pou_arr, SizeOf(system_pou_arr), 0);

fillchar(numb_wqp_add_sites, SizeOf(numb_wqp_add_sites), 0);
fillchar(numb_wqp_add_sites_total, SizeOf(numb_wqp_add_sites_total), 0);
fillchar(numb_wqp_sites_added, SizeOf(numb_wqp_sites_added), 0);
fillchar(numb_wqp_sites_added_prev, SizeOf(numb_wqp_sites_added_prev), 0);
fillchar(num_lsl_paper, SizeOf(num_lsl_paper), 0);
```

```
  SourceTreatChangeEver := false;
  if option = 'Baseline' then isBaseline := true
  else isBaseline := false;

  // read data request data values from database
  fCostVars.FillValueArray(Variables, RawVariables,
                           CostingData.SystemSize, CostingData.SourceWater, LSL,
                           CCT, CostingData.SystemType, 1, Config.PWS90PctBp1,
Config.PWS90PctBp2, SetProbsTo01, NewDraw, nil, isBaseline);
  NewDraw := false;

  Variables[fI.p_lsl] := LSL;
  if isBaseline then
    if VLSEpWOrkbook.p_b3 > -1 then
      Variables[fI.p_b3] := VLSEpWorkbook.p_b3;

  // load external variables values
  Variables[fI.EP] := VLSEpWorkbook.NumberEPs;
  Variables[fI.Pws_Cct] := CCT;

  Variables[fI.Pws_sw] := 0;
  Variables[fI.Pws_gw] := 0;
  if CostingData.SourceWater = 2 then
    Variables[fI.Pws_sw] := 1
  else if CostingData.SourceWater = 1 then
    Variables[fI.Pws_gw] := 1;

  Variables[fI.Pws_pop] := VLSEpWorkbook.Population;
  if VLSEpWorkbook.Connections > 0 then
    num_hh_per_connect := (VLSEpWorkbook.Population / Variables[fI.Numb_hh]) /
VLSEpWorkbook.Connections
  else
    num_hh_per_connect := 0;

  Variables[fI.meet_lslr_goal] := 1;
  Variables[fI.fail_nm1] := 0;
  Variables[fI.fail_nm2] := 0;

  failCost1 := false;
  failCost4 := false;
  failCost5 := false;
  failCost6 := false;
  failCost7 := false;
  failCost8 := false;

  InitCCTBVarsToZero(option);
```

```
Variables[fI.cct_existing_cost] := 0;
Variables[fI.cct_modify_cost] := 0;
Variables[fI.cct_install_cost] := 0;
Variables[fI.cct_findfix_cost] := 0;

fillchar(pp_lsl_replaced,SizeOf(pp_lsl_replaced),0);
fillchar(num_lsl_replace,SizeOf(num_lsl_replace),0);
fillchar(num_lsl_requested,SizeOf(num_lsl_requested),0);
fillchar(Num_lsl_replace_ale,SizeOf(Num_lsl_replace_ale),0);
fillchar(lslr_missed_goal,SizeOf(lslr_missed_goal),0);
fillchar(ff_cct, SizeOf(ff_cct), 0);
hff_cct := 0;

p_cct_study := trunc(Variables[fI.p_cct_study]);

Num_LSL_base := 0;
num_lsl_remain := 0;
num_paper_remain := 0;
num_requested := 0;
LSLRequestedVLS := 0;

if LSL = 1 then
begin
  Num_LSL_base := VLSEpWorkbook.NumberLSLs;
  num_lsl_remain := Num_LSL_base;
  hh_remain_lsl := max(0, (num_hh_per_connect * num_lsl_remain));
end;

NM := 0;

for i := 4 to Config.YearsOfAnalysis do
begin
  if i <= 35 then
  O.TryGetValue('pp_lsl_replacement_' + i.ToString, pp_lsl_replacement_rates[i]);
end;

for i := 1 to Config.YearsOfAnalysis do
begin
  if i <= 35 then
  O.TryGetValue('p_source_chng_' + i.ToString, tmp_double);
  p_source_chng_yr[i] := trunc(tmp_double);
end;

for i := 1 to Config.YearsOfAnalysis do
begin
  if i <= 35 then
  O.TryGetValue('p_source_sig_' + i.ToString, tmp_double);
  p_source_sig_yr[i] := trunc(tmp_double);
  end;

  for i := 1 to Config.YearsOfAnalysis do
  begin
    if i <= 35 then
    O.TryGetValue('p_treat_change_' + i.ToString, tmp_double);
    p_treat_change_yr[i] := trunc(tmp_double);
  end;

  for i := 1 to Config.YearsOfAnalysis do
  begin
    if i <= 35 then
    O.TryGetValue('pp_lsl_replaced_vol_pct_' + i.ToString, 
pp_lsl_replaced_vol_pct_yr[i]);
  end;

  for i := 1 to Config.YearsOfAnalysis do
  begin
    if i <= 35 then
    O.TryGetValue('BinChgLSL_' + i.ToString, tmp_double);
    BinChgLsl[i] := trunc(tmp_double);
    if i <= 35 then
    O.TryGetValue('BinChgNoLSL_' + i.ToString, tmp_double);
    BinChgNoLsl[i] := trunc(tmp_double);
  end;

  // Compute costs from CCTCostEquations

  if CCT = 1 then
  begin
    CCTCostEquations.ExistingCCT;
    UsefulLifeMod:=round(CCTCostEquations.UsefulLifeOM);
    ExistingCCTCostOM := CCTCostEquations.ComputeOMCost;
    CCTCostEquations.AdjustCCT(Variables[fI.targetph], Variables[fI.targetpo4]);
    AdjustCCTCostOM := CCTCostEquations.ComputeOMCost;
    AdjustCCTCostCap := CCTCostEquations.ComputeCapitalCost;
    AdjustCCTCostCapDisc:= AdjustCCTCostCap * (DiscRate / (1 - Power((1 + 
DiscRate),-CCTCostEquations.UsefulLifeCap)));
    AdjustCCTCostCapDisc_p:= AdjustCCTCostCap * (CostingData.CostCapital / (1 - 
Power((1 + CostingData.CostCapital),-CCTCostEquations.UsefulLifeCap)));
  end
  else
  begin
    CCTCostEquations.NewCCT(Variables[fI.targetph], Variables[fI.targetpo4]);
    UsefulLifeInstall:=round(CCTCostEquations.UsefulLifeCap);
    InstallCCTCostOM := CCTCostEquations.ComputeOMCost;
    InstallCCTCostCap := CCTCostEquations.ComputeCapitalCost;
    InstallCCTCostCapDisc:= InstallCCTCostCap * (DiscRate / (1 - Power((1 + 
DiscRate),-CCTCostEquations.UsefulLifeCap)));
    InstallCCTCostCapDisc_p:= InstallCCTCostCap * (CostingData.CostCapital / (1 - 
Power((1 + CostingData.CostCapital),-CCTCostEquations.UsefulLifeCap)));
  end;

    if CCT = 0 then begin
      if 
CCTCostEquations.arrBaselineph_wocct[CostingData.SourceWater,CCTCostEquations.iBasel
ineph_woccct] < 7.5 then begin
        CCTCostEquations.pbasepo4 := 0;
        CCTCostEquations.pbasephpo4 := 1;
      end;
    end;

    CCTCostEquations.FindAndFixCCT(CCTB);
    UsefulLifeFF:=round(CCTCostEquations.UsefulLifeCap);
    FindAndFixCostOM := CCTCostEquations.ComputeOMCost;
    FindAndFixCostCap := CCTCostEquations.ComputeCapitalCost;
    FindAndFixCostCapDisc:= FindAndFixCostCap * (DiscRate / (1 - Power((1 + 
DiscRate),-CCTCostEquations.UsefulLifeCap)));
    FindAndFixCostCapDisc_p:= FindAndFixCostCap * (CostingData.CostCapital / (1 - 
Power((1 + CostingData.CostCapital),-CCTCostEquations.UsefulLifeCap)));

  fAdjust_CCT := 0;
  fInstall_CCT := 0;

  lslr_conducted := false;

  if isBaseline then
  begin
    hp_lslr_partial[1] := Variables[fI.pp_lslr_partial];
  end
  else
  if  (option = 'OW') or (option='OW5L') then
  begin
    hp_lslr_paper[1] := Variables[fI.pp_lslr_paper];
    hp_lslr_partial[1] := Variables[fI.pp_lslr_partial];

    if AddCostingData.Num_Proxies = 0 then
    begin
      if Round(Config.DiscountRate*100)/100 = 0.03 then
        Variables[fI.annual_pou_cost_hh] := 111
      else if Round(Config.DiscountRate*100)/100 = 0.07 then
        Variables[fI.annual_pou_cost_hh] := 114
      else
        Variables[fI.annual_pou_cost_hh] := -1;
    end
    else
```

```
      Variables[fI.annual_pou_cost_hh] := 114;
  end;

  CCT_Change := false;
  bCCT_Change := false;
    if isBaseline then
      LeadConcentrationBins(CostingData.pwsid, option, 0, CostingData.SystemSize, 
CostingData.SourceWater,
                            LSL, CCT, POU, VLSEpWorkbook.Population, 
VLSEpWorkbook.Connections,
                            fAdjust_CCT, fInstall_CCT, cct_adjust_yr, 
cct_install_yr, pou_install_yr,
                            bCCT_Change,
                            CostingData.SamplingWeight, 0, 0, 0, 
AddCostingData.Num_Proxies, partial_cct_level,
                            hp_lslr_paper, hp_lslr_partial, 
pp_lsl_replacement_rates, ResetBin)
    else
      LeadConcentrationBins(CostingData.pwsid, option, 0, CostingData.SystemSize, 
CostingData.SourceWater,
                            LSL, CCT, POU, VLSEpWorkbook.Population, 
VLSEpWorkbook.Connections,
                            fAdjust_CCT, fInstall_CCT, cct_adjust_yr, 
cct_install_yr, pou_install_yr,
                            bCCT_Change,
                            CostingData.SamplingWeight, 0, 0, 0, 
AddCostingData.Num_Proxies, partial_cct_level,
                            hp_lslr_paper, hp_lslr_partial, 
pp_lsl_replaced_vol_pct_yr, ResetBin);

  if option = 'Baseline' then begin
    bp1 := 10;
    bp2 := 15;
  end
  else begin
    bp1 := Config.PWS90PctBp1;
    bp2 := Config.PWS90PctBp2;
  end;

  pws90pct := CostingData.P90_base;

    if pws90pct > bp2 then owBin := 1
    else if (pws90pct > bp1) and (pws90pct <= bp2) then owBin := 2
    else if (pws90pct <= bp1) then owBin := 3;

  owBin_tmp := 0;

  if option <> 'Baseline' then
  begin
    if Config.VolLeadProg = 0 then
      Variables[fI.p_vol_leadtap_prog] := 0;
  end;

  // Year loop
  for Y:=1 to fYears do begin

    Variables[fI.school_1a] := 0;
    Variables[fI.school_1b] := 0;
    Variables[fI.school_3a] := 0;
    Variables[fI.school_3b] := 0;
    Variables[fI.school_3c] := 0;
    Variables[fI.school_3d] := 0;
    Variables[fI.school_5a] := 0;
    Variables[fI.school_5b] := 0;

    if Config.SchoolOption = 'school_1a' then
      Variables[fI.school_1a] := 1
    else if Config.SchoolOption = 'school_1b' then
      Variables[fI.school_1b] := 1
    else if Config.SchoolOption = 'school_3a' then
      Variables[fI.school_3a] := 1
    else if Config.SchoolOption = 'school_3b' then
      Variables[fI.school_3b] := 1
    else if Config.SchoolOption = 'school_3c' then
      Variables[fI.school_3c] := 1
    else if Config.SchoolOption = 'school_3d' then
      Variables[fI.school_3d] := 1
    else if Config.SchoolOption = 'school_5a' then
      Variables[fI.school_5a] := 1
    else if Config.SchoolOption = 'school_5b' then
      Variables[fI.school_5b] := 1;

    if option <> 'Baseline' then
    begin
      Variables[fI.numb_second_schools_pub] :=
SchoolSampData.numb_second_schools_pub;
      Variables[fI.numb_elem_schools_pub] := SchoolSampData.numb_elem_schools_pub;
      Variables[fI.numb_second_schools_priv] :=
SchoolSampData.numb_second_schools_priv;
      Variables[fI.numb_elem_schools_priv] := SchoolSampData.numb_elem_schools_priv;
      Variables[fI.numb_daycares] := SchoolSampData.numb_daycares;
      Variables[fI.p_grandfather_opt_pub] := SchoolSampData.p_grandfather_opt_pub;
      Variables[fI.p_grandfather_opt_priv] := SchoolSampData.p_grandfather_opt_priv;
      Variables[fI.p_grandfather_mand_pub] := SchoolSampData.p_grandfather_mand_pub;
      Variables[fI.p_grandfather_mand_priv] :=
SchoolSampData.p_grandfather_mand_priv;
```

```
      Variables[fI.p_grandfather_opt_child] :=
SchoolSampData.p_grandfather_opt_child;
      Variables[fI.p_grandfather_mand_child] :=
SchoolSampData.p_grandfather_mand_child;
    end;

    Variables[fI.b_state_one] := 0;
    Variables[fI.b_state_two] := 0;

    if (SchoolSampData.stateabb = 'AR') or (SchoolSampData.stateabb = 'LA') or
       (SchoolSampData.stateabb = 'MS') or (SchoolSampData.stateabb = 'MO') or
       (SchoolSampData.stateabb = 'SC') or (SchoolSampData.stateabb = 'ND') then
      Variables[fI.b_state_one] := 1;

    if (SchoolSampData.stateabb = 'AR') or (SchoolSampData.stateabb = 'LA') or
       (SchoolSampData.stateabb = 'MS') or (SchoolSampData.stateabb = 'MO') or
       (SchoolSampData.stateabb = 'SC') then
      Variables[fI.b_state_two] := 1;

//Added 2/27/19 - s
NEWDRAW:=FALSE;

    if (option = 'Baseline') then
    begin
      Variables[fI.p_source_chng] := p_source_chng_yr[y];
      Variables[fI.p_source_sig] := p_source_sig_yr[y];
      Variables[fI.p_treat_change] := p_treat_change_yr[y];

      // Calculate Annual CCT and Find & Fix Micro Costs
      if CCTB = 1 then
      begin
        Variables[fI.cct_existing_cost] := ExistingCCTCostOM;
        CCTExisting := true;
      end
      else
        Variables[fI.cct_existing_cost] := 0;

      if y >= 4 then
      begin
        // Random change in PWS_90 due to sampling variation
        proxy1_pws90 := pws90pct * (1 + 0);

        // Change in water source or treatment technology
        if (b_install_cct[y] + b_modify_cct[y] > 0) or (POU = 1) then
        begin
          if ((p_source_chng_yr[y]*p_source_sig_yr[y]) = 0) and
(p_treat_change_yr[y] = 0) then proxy2_pws90 := proxy1_pws90
          else
          begin
            if LSL = 1 then
              tBin := BinChgLsl[y]
            else
              tBin := BinChgNoLsl[y];

            if tBin = 1 then
              tpws90pct := bp2 + 5
            else
            if tBin = 2 then
              tpws90pct := bp1 + ((bp2-bp1)/2)
            else
            if tBin = 3 then
              tpws90pct := bp1/2;

            if tpws90pct < proxy1_pws90 then
            begin
              proxy2_pws90 := tpws90pct;
            end
            else
              proxy2_pws90 := proxy1_pws90;

            SourceTreatChangeEver := true;
          end;
        end
        else
        begin
          if ((p_source_chng_yr[y]*p_source_sig_yr[y]) = 0) and
(p_treat_change_yr[y] = 0) then proxy2_pws90 := proxy1_pws90
          else
          begin
            if LSL = 1 then
              tBin := BinChgLsl[y]
            else
              tBin := BinChgNoLsl[y];

            if tBin = 1 then
              tpws90pct := bp2 + 5
            else
            if tBin = 2 then
              tpws90pct := bp1 + ((bp2-bp1)/2)
            else
            if tBin = 3 then
              tpws90pct := bp1/2;

            proxy2_pws90 := tpws90pct;

            SourceTreatChangeEver := true;
          end;
        end;

        // CCT and POU
          if (proxy2_pws90 > bp2) and (CCT = 0) and (not CCT_Change) and
(p_cct_study = 1) then
          begin
            for i := y + 5 to Config.YearsOfAnalysis do
              InstallCCT[i] := true;
            proxy3_pws90[y+7] := Variables[fi.post_cct_p90_bin1];
            CCT_Change := true;
            if cct_install_yr = 0 then cct_install_yr := y + 5;

            b_cct_study_rec_install[y+1] := 1;
            b_cct_study_install[y+3] := 1;
            for i := y + 5 to Config.YearsOfAnalysis do
              b_install_cct[i] := 1;
            b_install_cct_mc[y+6] := 1;

            // System Capital Cost undiscounted
            ValuesCapital[2] := ValuesCapital[2] + InstallCCTCostCap;
          end
          else
          if (proxy2_pws90 > bp2) and (CCT = 0) and (not CCT_Change) and
(p_cct_study = 0) then
          begin
            for i := y + 4 to Config.YearsOfAnalysis do
              InstallCCT[i] := true;
            proxy3_pws90[y+6] := Variables[fi.post_cct_p90_bin1];
            CCT_Change := true;
            if cct_install_yr = 0 then cct_install_yr := y + 4;

            b_cct_study_rec_install[y+1] := 1;
            b_state_cct_treatment_install[y+2] := 1;
            for i := y + 4 to Config.YearsOfAnalysis do
              b_install_cct[i] := 1;
            b_install_cct_mc[y+5] := 1;

            // System Capital Cost undiscounted
            ValuesCapital[2] := ValuesCapital[2] + InstallCCTCostCap;
          end
          else
          if (proxy2_pws90 > bp2) and (CCT = 1) and (not CCT_Change) and
(p_cct_study = 1) then
          begin
            for i := y + 4 to Config.YearsOfAnalysis do
              AdjustCCT[i] := true;
            //225change
            proxy3_pws90[y+6] := Variables[fI.post_cct_p90_bin1];
            CCT_Change := true;
            if cct_adjust_yr = 0 then cct_adjust_yr := y + 4;

            b_cct_study_rec_mod[y+1] := 1;
            b_cct_study_mod[y+3] := 1;
            for i := y + 4 to Config.YearsOfAnalysis do
              b_modify_cct[i] := 1;
            b_modify_cct_mc[y+5] := 1;
          end
          else
          if (proxy2_pws90 > bp2) and (CCT = 1) and (not CCT_Change) and
(p_cct_study = 0) then
          begin
            for i := y + 3 to Config.YearsOfAnalysis do
              AdjustCCT[i] := true;
            //225change
            proxy3_pws90[y+5] := Variables[fI.post_cct_p90_bin1];
            CCT_Change := true;
            if cct_adjust_yr = 0 then cct_adjust_yr := y + 3;

            b_cct_study_rec_mod[y+1] := 1;
            b_state_cct_treatment_mod[y+2] := 1;
            for i := y + 3 to Config.YearsOfAnalysis do
              b_modify_cct[i] := 1;
            b_modify_cct_mc[y+4] := 1;
          end;

          if proxy3_pws90[y] = 0 then
            proxy3_pws90[y] := proxy2_pws90;
      end; // y >= 4
    end // if option = 'Baseline'
    else
    if  (option = 'OW') or (option='OW5L') then
    begin
      Variables[fI.p_source_chng] := p_source_chng_yr[y];
      Variables[fI.p_source_sig] := p_source_sig_yr[y];
      Variables[fI.p_treat_change] := p_treat_change_yr[y];

      // Calculate Annual CCT and Find & Fix Micro Costs
      if CCTB = 1 then
      begin
        Variables[fI.cct_existing_cost] := ExistingCCTCostOM;
        CCTExisting := true;
      end
      else
        Variables[fI.cct_existing_cost] := 0;
      if y >= 4 then
      begin
        if LSL = 1 then Variables[fI.b_lslr_study] := 1;

        if (proxy2_pws90 > BP1) and (CCT = 0) then
        begin
          b_cct_study_install[y+2] := 1;
          cct_study_done_yr := y+2;
        end;

        //calculate the additional WQP sites in year y without regard to maximum
        if owBin = 3 then begin
          if Variables[fI.p_tap_annual] = 1 then
            numb_wqp_add_sites[y] := Variables[fI.pp_overlap_find_fix] *
Variables[fI.pp_above_al_bin_three] * Variables[fI.numb_reduced_tap]
          else if (Variables[fI.p_tap_triennial] = 1) and
                  ((y=4) or (y=7) or (y=10) or (y=13) or (y=16) or (y=19) or
                   (y=22) or (y=25) or (y=28) or (y=31) or (y=34)) then
            numb_wqp_add_sites[y] := Variables[fI.pp_overlap_find_fix] *
Variables[fI.pp_above_al_bin_three] * Variables[fI.numb_reduced_tap]
          else if (Variables[fI.p_tap_nine] = 1) and
                  ((y=4) or (y=13) or (y=22) or (y=32)) then
            numb_wqp_add_sites[y] := Variables[fI.pp_overlap_find_fix] *
Variables[fI.pp_above_al_bin_three] * Variables[fI.numb_reduced_tap]
          else numb_wqp_add_sites[y] := Variables[fI.pp_overlap_find_fix] *
Variables[fI.pp_above_al_bin_three] * (2 * Variables[fI.numb_samp_customer]);
        end
        else if owBin = 2 then
          numb_wqp_add_sites[y] := Variables[fI.pp_overlap_find_fix] *
Variables[fI.pp_above_al_bin_two] * Variables[fI.numb_samp_customer]
        else if owBin = 1 then
          numb_wqp_add_sites[y] := Variables[fI.pp_overlap_find_fix] *
Variables[fI.pp_above_al_bin_one] * (2 * Variables[fI.numb_samp_customer]);

        //calculate the total number of additional wqp samples in by year y with max
applied
        numb_wqp_add_sites_total[y] := min((numb_wqp_add_sites_total[y-1] +
numb_wqp_add_sites[y]), Variables[fI.numb_enhance_wqp]);

        //numb_reduced_wqp (which are samples) = 2* number of baseline reduced wqp
sites
        //numb_enhance_wqp (which are samples)= 2* number of baseline enhanced sites

        //calculates the added number of sites in year y

        numb_wqp_sites_added[y] := numb_wqp_add_sites_total[y] -
numb_wqp_add_sites_total[y-1];
        numb_wqp_sites_added_prev[y] := numb_wqp_add_sites_total[y] -
```

```
numb_wqp_sites_added[y];

        // Random change in PWS_90 due to sampling variation
        proxy1_pws90 := pws90pct * (1 + 0);

        // Change in water source or treatment technology
        if (b_install_cct[y] + b_modify_cct[y] > 0) then
        begin
          if ((p_source_chng_yr[y]*p_source_sig_yr[y]) = 0) and
(p_treat_change_yr[y] = 0) then proxy2_pws90 := proxy1_pws90
          else
          begin
            if LSL = 1 then
              tBin := BinChgLsl[y]
            else
              tBin := BinChgNoLsl[y];

            if tBin = 1 then
              tpws90pct := bp2 + 5
            else
            if tBin = 2 then
              tpws90pct := bp1 + ((bp2-bp1)/2)
            else
            if tBin = 3 then
              tpws90pct := bp1/2;

            if tpws90pct < proxy1_pws90 then
            begin
              proxy2_pws90 := tpws90pct;
            end
            else
              proxy2_pws90 := proxy1_pws90;

            SourceTreatChangeEver := true;
          end;
        end
        else
        begin
          if ((p_source_chng_yr[y]*p_source_sig_yr[y]) = 0) and
(p_treat_change_yr[y] = 0) then proxy2_pws90 := proxy1_pws90
          else
          begin
            if LSL = 1 then
              tBin := BinChgLsl[y]
            else
              tBin := BinChgNoLsl[y];

            if tBin = 1 then
              tpws90pct := bp2 + 5
            else
            if tBin = 2 then
              tpws90pct := bp1 + ((bp2-bp1)/2)
            else
            if tBin = 3 then
              tpws90pct := bp1/2;

            proxy2_pws90 := tpws90pct;

            SourceTreatChangeEver := true;
          end;
        end;

        // CCT, LSLR, and POU Plans/Studies
        if (proxy2_pws90 > bp1) and (CCT = 0) then begin
          b_cct_study_install[y+2] := 1;
          cct_study_done_yr := y + 2;
        end;

        // CCT and POU not VLS
{$IFNDEF NOTRIGCCT}
          if (proxy2_pws90 > bp1) and (proxy2_pws90 < bp2) and (CCT = 1) and (not 
CCT_Change) then
          begin
            for i := y + 3 to Config.YearsOfAnalysis do
            begin
              AdjustCCT[i] := true;
              b_modify_cct[i] := 1;
              b_modify_cct_tl[i] := 1;
            end;

            proxy3_pws90[y+5] := Variables[fi.post_cct_p90_bin2];
            CCT_Change := true;
            if cct_adjust_yr = 0 then cct_adjust_yr := y + 3;

            b_cct_study_rec_mod[y+1] := 1;
            b_cct_study_mod[y+1] := 1;

            b_cct_study_rec_mod_tl[y+1] := 1;
            b_cct_study_mod_tl[y+1] := 1;
          end
          else
{$ENDIF}
          if (proxy2_pws90 > bp2) and (CCT = 1) and (not CCT_Change) then
          begin
            for i := y + 4 to Config.YearsOfAnalysis do
            begin
```

```
       AdjustCCT[i] := true;
       b_modify_cct[i] := 1;
       b_modify_cct_al[i] := 1;
      end;

      proxy3_pws90[y+6] := Variables[fi.post_cct_p90_bin1];
      CCT_Change := true;
      if cct_adjust_yr = 0 then cct_adjust_yr := y + 4;

      b_cct_study_rec_mod[y+1] := 1;
      b_cct_study_mod[y+3] := 1;

      b_cct_study_rec_mod_al[y+1] := 1;
      b_cct_study_mod_al[y+3] := 1;
     end
     else
     if (proxy2_pws90 > bp2) and (CCT = 0) and (not CCT_Change) then
     begin
      for i := max(cct_study_done_yr+2,y+4) to Config.YearsOfAnalysis do
      begin
       InstallCCT[i] := true;
       b_install_cct[i] := 1;
      end;

      proxy3_pws90[max(cct_study_done_yr+4,y+6)] :=
Variables[fi.post_cct_p90_bin1];
      CCT_Change := true;
      if cct_install_yr = 0 then cct_install_yr :=
max(cct_study_done_yr+2,y+4);

      // System Capital Cost undiscounted
      ValuesCapital[2] := ValuesCapital[2] + InstallCCTCostCap;
     end;

     if proxy3_pws90[y] = 0 then
      proxy3_pws90[y] := proxy2_pws90;
   end; // y >= 4
  end; // end option = 'OW'

  Variables[fI.b_cct_study_rec_install] := b_cct_study_rec_install[y];
  Variables[fI.b_cct_study_install] := b_cct_study_install[y];
  Variables[fI.b_state_cct_treatment_install] := b_state_cct_treatment_install[y];
  Variables[fI.b_cct_study_rec_mod] := b_cct_study_rec_mod[y];
  Variables[fI.b_cct_study_mod] := b_cct_study_mod[y];
  Variables[fI.b_state_cct_treatment_mod] := b_state_cct_treatment_mod[y];
  Variables[fI.b_install_cct] := b_install_cct[y];
  Variables[fI.b_install_cct_mc] := b_install_cct_mc[y];
  Variables[fI.b_modify_cct] := b_modify_cct[y];
```

```
    Variables[fI.b_modify_cct_mc] := b_modify_cct_mc[y];
    Variables[fI.b_cct_study_rec_mod_tl] := b_cct_study_rec_mod_tl[y];
    Variables[fI.b_cct_study_mod_tl] := b_cct_study_mod_tl[y];
    Variables[fI.b_modify_cct_tl] := b_modify_cct_tl[y];
    Variables[fI.b_state_cct_treatment_mod_tl] := b_state_cct_treatment_mod_tl[y];
    Variables[fI.b_cct_study_rec_mod_al] := b_cct_study_rec_mod_al[y];
    Variables[fI.b_cct_study_mod_al] := b_cct_study_mod_al[y];
    Variables[fI.b_modify_cct_al] := b_modify_cct_al[y];
    Variables[fI.b_state_cct_treatment_mod_al] := b_state_cct_treatment_mod_al[y];
    Variables[fI.system_pou] := system_pou_arr[y];

    Variables[fI.numb_wqp_sites_added] := numb_wqp_sites_added[y];
    Variables[fI.numb_wqp_sites_added_prev] := numb_wqp_sites_added_prev[y];

    Variables[fI.num_lsl_replace] := 0;
    Variables[fI.hh_remain_lsl] := 0;
    Variables[fI.num_lsl_paper] := 0;

    proxy4_pws90 := -1;
    if (LSL = 1) then
    begin
      if option = 'Baseline' then
      begin
        if y >= 4 then
        begin
          if proxy3_pws90[y] > bp2 then
          begin
            if not lslr_conducted then begin

              // 7% replacement
              num_lsl_replace[y] := min(Num_lsl_remain,
                                         (Num_LSL_base*0.21)*

(1-(Variables[fI.pp_lcr_test]*Variables[fI.pp_lcr_test_yes])))
                                        / 3;
              num_lsl_replace[y+1] := min(Num_lsl_remain,
                                             (Num_LSL_base*0.21)*

(1-(Variables[fI.pp_lcr_test]*Variables[fI.pp_lcr_test_yes])))
                                        / 3;
              num_lsl_replace[y+2] := min(Num_lsl_remain,
                                           (Num_LSL_base*0.21)*

(1-(Variables[fI.pp_lcr_test]*Variables[fI.pp_lcr_test_yes])))
                                      / 3;
              lslr_conducted := true;

            end;
          end;
        end;

          if y >= 4 then
          begin
            if y = 4 then
              hh_remain_lsl := max(0, (num_hh_per_connect * num_lsl_remain));

            if ((Num_lsl_remain > 0) and
               (num_replace >= (Num_LSL_base * 0.21 *
(1-(Variables[fI.pp_lcr_test]*Variables[fI.pp_lcr_test_yes]))))) then
            begin
              proxy4_pws90 := bp1 + ((bp2-bp1)/2);
            end;

            if num_lsl_replace[y] > 0 then
              Variables[fi.b_lslr_mand] := 1
            else
              Variables[fi.b_lslr_mand] := 0;

            if Num_lsl_remain <= 0 then
            begin
              LSL := 0;
              Variables[fi.p_lsl] := 0;
              NewDraw := true;
              Num_lsl_remain := 0;

              proxy4_pws90 := bp1/2;
            end;

            num_replace := 0;
            LSLReplacedMandatory := 0;
            for i := 1 to y do
            begin
               num_replace := num_replace + num_lsl_replace[i];
               LSLReplacedMandatory := LSLReplacedMandatory + num_lsl_replace[i];
            end;


            Num_lsl_remain := num_LSL_base - LSLReplacedMandatory;
            hh_remain_lsl := max(0, (num_hh_per_connect * num_lsl_remain));

            Variables[fI.num_lsl_replace] := num_lsl_replace[y];
            Variables[fI.hh_remain_lsl] := hh_remain_lsl;
            num_lsl_paper[y] := ((Num_LSL_base*0.21) *
(Variables[fI.pp_lcr_test]))/3;
            Variables[fI.num_lsl_paper] := num_lsl_paper[y];
          end;
      end // end option = Baseline
      else if  (option = 'OW') or (option='OW5L') then
      begin
        if y >= 4 then
        begin
          if Num_lsl_remain > 0 then
          begin
              if (pws90pct > bp1) and (pws90pct <= bp2) then
              begin
{$IFNDEF NOGOALLCR}
                  Num_lsl_replace[y] := min(pp_lsl_replaced_vol_pct_yr[y] *
(Num_LSL_base + num_paper_remain), Num_lsl_remain);
{$ELSE}
                  Num_lsl_replace[y] := 0;
{$ENDIF}
                if num_lsl_replace[y] > 0 then
                  Variables[fi.b_lslr_vol] := 1
                else
                  Variables[fi.b_lslr_vol] := 0;
              end
              else
              if pws90pct > bp2 then
              begin
                  Num_lsl_replace[y] := min(0.03 * (Num_LSL_base +
num_paper_remain), Num_lsl_remain);
                if num_lsl_replace[y] > 0 then
                  Variables[fi.b_lslr_mand] := 1
                else
                  Variables[fi.b_lslr_mand] := 0;
              end
              else
              begin
                Num_lsl_replace[y] := 0;
                num_lsl_requested[y] := Variables[fI.pp_cust_init_lslr] *
Num_lsl_remain;
                Variables[fI.b_lslr_requested] := 1;
              end;

            proxy4_pws90 := proxy3_pws90[y];
          end
          else
          begin
            LSL := 0;
            Variables[fI.p_lsl] := LSL;
            NewDraw := true;
            Num_lsl_remain := 0;

            ttpws90pct := bp1/2;
            if proxy3_pws90[y] < ttpws90pct then
              proxy4_pws90 := proxy3_pws90[y]
            else
              proxy4_pws90 := ttpws90pct;
          end;

          num_replace := 0;
          num_remain := 0;
          for i := 1 to y-1 do
          begin
            num_replace := num_replace + Num_lsl_replace[i];
            num_remain := num_remain + num_lsl_paper[i];
          end;

          Num_lsl_remain := Num_LSL_Base - num_replace;
          hh_remain_lsl := max(0, (num_hh_per_connect * num_lsl_remain));

          num_paper_remain := (Num_LSL_Base * Variables[fI.pp_lslr_paper]) -
num_remain;

          Variables[fI.num_lsl_replace] := Num_lsl_replace[y];
          Variables[fI.hh_remain_lsl] := hh_remain_lsl;
          num_lsl_paper[y] :=  Num_lsl_replace[y] * Variables[fI.pp_lslr_paper];
          Variables[fI.num_lsl_paper] := num_lsl_paper[y];

          if pws90pct > bp2 then
            LSLReplacedMandatory := LSLReplacedMandatory + num_lsl_replace[y]
          else
          if (pws90pct > bp1) and (pws90pct <= bp2) then
            LSLReplacedVoluntary := LSLReplacedVoluntary + num_lsl_replace[y];

          // Failure to meet LSLR Voluntary Program in Bin 2
            if (pws90pct > bp1) and (pws90pct <= bp2) then
            begin
              Meet_LSLR_Goal := 0;
              if Num_LSL_base > 0 then
              begin
                if (Num_lsl_replace[y]/Num_LSL_base) >=
Variables[fI.pp_lsl_replaced_vol_goal] then
                  Meet_LSLR_Goal := 1;
              end;
            end;

            if (pws90pct <= bp1) or ((pws90pct > bp1) and (pws90pct <= bp2) and
(Meet_LSLR_Goal = 1)) then
              NM := 0
            else
              NM := NM + 1;

            Variables[fI.Meet_Lslr_Goal] := Meet_LSLR_Goal;

            if NM = 0 then
            begin
              Variables[fI.fail_nm1] := 0;
              Variables[fI.fail_nm2] := 0;
            end;

            if NM >= 1 then Variables[fI.fail_nm1] := 1;
            if NM >= 2 then Variables[fI.fail_nm2] := 1;
        end;
      end; // end option = OW
    end // end has LSL
    else
    begin
      Variables[fI.fail_nm1] := 0;
      Variables[fI.fail_nm2] := 0;
      Variables[fI.Meet_Lslr_Goal] := 1;
    end;

    if proxy4_pws90 = -1 then
      proxy4_pws90 := proxy3_pws90[y];


    Variables[fI.num_lsl_requested] := num_lsl_requested[y];
    LSLRequestedVLS := LSLRequestedVLS + num_lsl_requested[y];

    if option = 'Baseline' then
    begin
      if y >= 4 then
      begin
        // proxy4_pws90 > 0 when all LSL replaced
        if proxy4_pws90 > 0 then
          pws90pct := min(proxy4_pws90, proxy3_pws90[y])
        else
          pws90pct := proxy3_pws90[y];
      end;
    end;

    Variables[fI.num_lsl_remain] := num_lsl_remain;
    Variables[fI.num_paper_remain] := num_paper_remain;

    // find and fix
    if ( (option = 'OW') or (option='OW5L')) and (y >= 4) then
    begin
      Num_tap_ge_al := 0;
```

```
      if Config.VolLeadProg = 1 then
      begin
        // bin = 3
        if pws90pct <= bp1 then
        begin
          if (Variables[fI.p_tap_nine] = 1) and
             ((y=4) or (y=13) or (y=22) or (y=31)) then
            Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al(Variables[fI.numb_reduced_tap] +

(Variables[fI.p_vol_leadtap_prog] *

(Variables[fI.num_vol_leadtap_samples_per_k] * (VLSEpWorkbook.Population /
100000))), Variables[fI.pp_above_al_bin_three])
          else
          if (Variables[fI.p_tap_triennial] = 1) and
             ((y=4) or (y=7) or (y=10) or (y=13) or (y=16) or (y=19) or (y=22) or
(y=25) or (y=28) or (y=31) or (y=34)) then
            Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al(Variables[fI.numb_reduced_tap] +

(Variables[fI.p_vol_leadtap_prog] *

(Variables[fI.num_vol_leadtap_samples_per_k] * (VLSEpWorkbook.Population /
100000))), Variables[fI.pp_above_al_bin_three])
          else
          if (Variables[fI.p_tap_annual] = 1) then
            Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al(Variables[fI.numb_reduced_tap] +

(Variables[fI.p_vol_leadtap_prog] *

(Variables[fI.num_vol_leadtap_samples_per_k] * (VLSEpWorkbook.Population /
100000))), Variables[fI.pp_above_al_bin_three])
          else
          if (1 - Variables[fI.p_tap_nine] - Variables[fI.p_tap_annual] -
Variables[fI.p_tap_triennial] = 1) then
            Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al((Variables[fI.numb_samp_customer] * 2) +

(Variables[fI.p_vol_leadtap_prog] *

(Variables[fI.num_vol_leadtap_samples_per_k] * (VLSEpWorkbook.Population /
100000))), Variables[fI.pp_above_al_bin_three]);
        end
        // bin = 2
        else
        if (pws90pct > bp1) and (pws90pct <= bp2) then
          Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al(Variables[fI.numb_samp_customer] +

(Variables[fI.p_vol_leadtap_prog] *

(Variables[fI.num_vol_leadtap_samples_per_k] * (VLSEpWorkbook.Population /
100000))), Variables[fI.pp_above_al_bin_two])
        // bin = 1
        else
          Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al((Variables[fI.numb_samp_customer] * 2) +

(Variables[fI.p_vol_leadtap_prog] *

(Variables[fI.num_vol_leadtap_samples_per_k] * (VLSEpWorkbook.Population /
100000))), Variables[fI.pp_above_al_bin_one]);
      end
      else
      begin
        if pws90pct <= bp1 then
        begin
          if (Variables[fI.p_tap_nine] = 1) and
             ((y=4) or (y=13) or (y=22) or (y=31)) then
            Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al(Variables[fI.numb_reduced_tap],
Variables[fI.pp_above_al_bin_three])
          else
          if (Variables[fI.p_tap_triennial] = 1) and
             ((y=4) or (y=7) or (y=10) or (y=13) or (y=16) or (y=19) or (y=22) or
(y=25) or (y=28) or (y=31) or (y=34)) then
            Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al(Variables[fI.numb_reduced_tap],
Variables[fI.pp_above_al_bin_three])
          else
          if (Variables[fI.p_tap_annual] = 1) then
            Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al(Variables[fI.numb_reduced_tap],
Variables[fI.pp_above_al_bin_three])
          else
          if (Variables[fI.p_tap_nine] = 0) and (Variables[fI.p_tap_triennial] = 0)
and (Variables[fI.p_tap_annual] = 0) then
            Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al(Variables[fI.numb_samp_customer]*2,
Variables[fI.pp_above_al_bin_three]);
        end
        else
        if (pws90pct > bp1) and (pws90pct <= bp2) then
```

```
          Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al(Variables[fI.numb_samp_customer],
Variables[fI.pp_above_al_bin_two])
        else
          Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al(Variables[fI.numb_samp_customer]*2,
Variables[fI.pp_above_al_bin_one]);
      end;

      if Num_tap_ge_al > 0 then fnf := true;

      if (Num_tap_ge_al = 0) or (b_install_cct[y] + b_modify_cct[y] = 0) then
ff_cct[y] := 0
      else if (Num_tap_ge_al > 0) and (b_install_cct[y] + b_modify_cct[y] > 0) then
      begin
        sumff_cct := 0;
        for i := 1 to y-1 do
           sumff_cct := sumff_cct + ff_cct[i];

        if sumff_cct = 0 then ff_cct[y] := 1
        else if sumff_cct = 1 then ff_cct[y] := 2
        else if sumff_cct = 3 then ff_cct[y] := 3
        else if sumff_cct >= 6 then ff_cct[y] := 4;

        if ff_cct[y] >= 2 then begin
          FindAndFix := true;
          Variables[fI.b_findfix] := 1;
          hFF_CCT := ff_cct[y];
          if hFF_CCt=2 then hFFY2:=Y else
          if hFF_CCt=3 then hFFY3:=Y;
        end;

        if (ff_cct[y] = 4) and (pws90pct > bp2) then
          ff_pws90pct:=Variables[fI.post_ff_p90_bin1] else
        if (ff_cct[y] = 4) and ((pws90pct > bp1) and (pws90pct <= bp2)) then
          ff_pws90pct:=Variables[fI.post_ff_p90_bin2];
      end;
    end;

    if (option = 'OW') or (option = 'OW5L') then
    begin
      if y >= 4 then
      begin
        if ff_pws90pct > 0 then
          pws90pct := min(proxy4_pws90, ff_pws90pct)
        else
          pws90pct := proxy4_pws90;
      end;
    end;

      if (CCTB = 1) and (b_install_cct[y] + b_modify_cct[y] > 0) then begin
        Variables[fI.cct_modify_cost] := AdjustCCTCostOM + AdjustCCTCostCapDisc -
ExistingCCTCostOM;
        if (Y-cct_adjust_yr) MOD UsefulLifeMod = 0 then
          Variables[fI.cct_modify_cost_umra] := AdjustCCTCostCap
        else
          Variables[fI.cct_modify_cost_umra] := 0;
        Variables[fI.cct_modify_cost_umra_om] := AdjustCCTCostOM -
ExistingCCTCostOM;
        Variables[fI.cct_modify_cost_p] := AdjustCCTCostOM + AdjustCCTCostCapDisc_p
- ExistingCCTCostOM;
      end else begin
        Variables[fI.cct_modify_cost] := 0;
        Variables[fI.cct_modify_cost_umra] := 0;
        Variables[fI.cct_modify_cost_umra_om] := 0;
        Variables[fI.cct_modify_cost_p] := 0;
      end;

      if (CCTB = 0) and (b_install_cct[y] + b_modify_cct[y] > 0) then begin
        Variables[fI.cct_install_cost] := InstallCCTCostOM + InstallCCTCostCapDisc;
        if (Y-cct_install_yr) MOD UsefulLIfeInstall = 0 then
          Variables[fI.cct_install_cost_umra] := InstallCCTCostCap
        else
          Variables[fI.cct_install_cost_umra] := 0;
        Variables[fI.cct_install_cost_umra_om] := InstallCCTCostOM;
        Variables[fI.cct_install_cost_p] := InstallCCTCostOM +
InstallCCTCostCapDisc_p;
      end else begin
        Variables[fI.cct_install_cost] := 0;
        Variables[fI.cct_install_cost_umra] := 0;
        Variables[fI.cct_install_cost_umra_om] := 0;
        Variables[fI.cct_install_cost_p] := 0;
      end;

      Variables[fI.cct_findfix_cost]:=0;
      Variables[fI.cct_findfix_cost_umra]:=0;
      Variables[fI.cct_findfix_cost_umra_om]:=0;
      Variables[fI.cct_findfix_cost_p]:=0;

{the *_p below is the annualizing capital cost for CCT and Find and Fix }

      if fnf and (hFF_CCT = 2) then
      begin
        Variables[fI.cct_findfix_cost] := (Variables[fI.hrs_act_wqp_op] *
Variables[fI.rate_op]) + Variables[fI.cost_act_wqp];
        Variables[fI.cct_findfix_cost_umra] := (Variables[fI.hrs_act_wqp_op] *
Variables[fI.rate_op]) + Variables[fI.cost_act_wqp];
        Variables[fI.cct_findfix_cost_umra_om] := 0;
        Variables[fI.cct_findfix_cost_p] := (Variables[fI.hrs_act_wqp_op] *
Variables[fI.rate_op]) + Variables[fI.cost_act_wqp];
      end
      else
      if (CCTB = 1) and (hFF_CCT = 3) then begin
        Variables[fI.cct_findfix_cost] := (FindAndFixCostOM + FindAndFixCostCapDisc
- AdjustCCTCostOM) * (1/CostingData.EntryPoints);
        if (Y-hFFY2) MOD UsefulLIfeFF = 0 then
          Variables[fI.cct_findfix_cost_umra] := (FindAndFixCostCap) *
(1/CostingData.EntryPoints)
        else
          Variables[fI.cct_findfix_cost_umra] := 0;
        Variables[fI.cct_findfix_cost_umra_om] := (FindAndFixCostOM -
AdjustCCTCostOM) * (1/CostingData.EntryPoints);
        Variables[fI.cct_findfix_cost_p] := (FindAndFixCostOM +
FindAndFixCostCapDisc_p - AdjustCCTCostOM) * (1/CostingData.EntryPoints);
      end else if (CCTB = 0) and (hFF_CCT = 3) then begin
        Variables[fI.cct_findfix_cost] := (FindAndFixCostOM + FindAndFixCostCapDisc
- InstallCCTCostOM) * (1/CostingData.EntryPoints);
        if (Y-hFFY2) MOD UsefulLIfeFF = 0 then
          Variables[fI.cct_findfix_cost_umra] := (FindAndFixCostCap) *
(1/CostingData.EntryPoints)
        else
          Variables[fI.cct_findfix_cost_umra] := 0;
        Variables[fI.cct_findfix_cost_umra_om] := (FindAndFixCostOM -
InstallCCTCostOM) * (1/CostingData.EntryPoints);
        Variables[fI.cct_findfix_cost_p] := (FindAndFixCostOM +
FindAndFixCostCapDisc_p - InstallCCTCostOM) * (1/CostingData.EntryPoints);
      end else if (CCTB = 1) and (hFF_CCT >= 4) then begin
        Variables[fI.cct_findfix_cost] := (FindAndFixCostOM + FindAndFixCostCapDisc
- AdjustCCTCostOM);
        if (Y-hFFY3) MOD UsefulLIfeFF = 0 then
          Variables[fI.cct_findfix_cost_umra] := (FindAndFixCostCap)
        else
          Variables[fI.cct_findfix_cost_umra] := 0;
        Variables[fI.cct_findfix_cost_umra_om] := (FindAndFixCostOM -
AdjustCCTCostOM);
        Variables[fI.cct_findfix_cost_p] := (FindAndFixCostOM +
FindAndFixCostCapDisc_p - AdjustCCTCostOM);
      end else if (CCTB = 0) and (hFF_CCT >= 4) then begin
        Variables[fI.cct_findfix_cost] := (FindAndFixCostOM + FindAndFixCostCapDisc
- InstallCCTCostOM);
        if (Y-hFFY3) MOD UsefulLIfeFF = 0 then
          Variables[fI.cct_findfix_cost_umra] := (FindAndFixCostCap)
        else
          Variables[fI.cct_findfix_cost_umra] := 0;
        Variables[fI.cct_findfix_cost_umra_om] := (FindAndFixCostOM -
InstallCCTCostOM);
        Variables[fI.cct_findfix_cost_p] := (FindAndFixCostOM +
FindAndFixCostCapDisc_p - InstallCCTCostOM);
      end;

      if hFF_CCT >= 2 then
        HasFindAndFixCost := true;

    Variables[fI.pbaseph] := CCTCostEquations.pbaseph;
    Variables[fI.pbasepo4] := CCTCostEquations.pbasepo4;
    Variables[fI.pbasephpo4] := CCTCostEquations.pbasephpo4;

{
    if (SourceTreatChangeEver) and
       (not ((b_install_cct[y] + b_modify_cct[y] > 0) or (InstallPOU))) then
    begin
      owBin := owBin;
    end
    else
    begin
      if pws90pct > bp2 then owBin := 1
      else if (pws90pct > bp1) and (pws90pct <= bp2) then owBin := 2
      else if pws90pct <= bp1 then owBin := 3;
    end;
}

    if pws90pct > bp2 then owBin_tmp := 1
    else if (pws90pct > bp1) and (pws90pct <= bp2) then owBin_tmp := 2
    else if pws90pct <= bp1 then owBin_tmp := 3;

    pws90pctCCT_yr[y] := proxy2_pws90;
    pws90pctLSL_yr[y] := proxy3_pws90[y];

    // tpws90pct is set above if there was a source water or treatment change
    if tpws90pct < proxy1_pws90 then
    begin
      owbin := owBin_tmp;
    end
    else
    if owBin_tmp > owbin then
    begin
      owbin := owBin_tmp;
    end
    else
    begin
      owBin := owBin;
    end;
```

```
// Compute Costs
for C in CostSteps do begin
  // only calculate cost in appropriate year
  if not C.arrCalculateYr[Y] then continue;
  // if very large system don't calculate ep level costs in this loop
  if not C.fCostStepRec.VLSEpLevel then continue;

  if not (((owBin = 1) and (C.fCostStepRec.Bin1 = 1)) or
          ((owBin = 2) and (C.fCostStepRec.Bin2 = 1)) or
          ((owBin = 3) and (C.fCostStepRec.Bin3 = 1))) then continue;

  C.Evaluate(Cost,Labor,OM,Hours,DoIt);

  // only do the following for CCT cost variables in hopes of saving processing
  // this is for determining CCT install or adjust event
      if Cost > 0 then
      begin
        HasCCTCostVLS := true;

        if (C.fCostStepRec.Frequency = 'Once') then
        begin
          for yy := y+1 to fYears do C.arrCalculateYr[yy] := false;
        end;

        if option = 'Baseline' then
        begin
          if C.BaselineCCTModify then
          begin
            ExistingCCT := false;
            CCTAdjusted := true;
            CCTAdjusted_ale := true;
            CCTAdjusted_tle := false;
          end
          else if C.BaselineCCTInstall then
          begin
            CCT := 1;
            NewCCT := true;
            NewDraw := true;
            CCTInstalled := true;
          end;
        end
        else if  (option = 'OW') or (option='OW5L') then
        begin
          if C.OWCCTModify then
          begin
            //AdjustCCT := true;
            ExistingCCT := false;
```

```
                CCTAdjusted := true;
                if C.OWCCTModify_ale then CCTAdjusted_ale := true;
                if C.OWCCTModify_tle then CCTAdjusted_tle := true;
              end
              else if C.OWCCTInstall then
              begin
                CCT := 1;
                NewCCT := true;
                NewDraw := true;
                CCTInstalled := true;
              end;
            end;
          end;

      if not HasLSLRCostVLS then
      begin
        HasLSLRCostVLS := (Cost > 0) and (C.fCostStepRec.LSLRCost);
      end;

      if C.SysLSLRCapital then
        ValuesCapital[0] := ValuesCapital[0] + Cost;

      if C.HhLSLRCapital then
        ValuesCapital[1] := ValuesCapital[1] + Cost;

      // these costs should incur only once
      if c.fAgg2ID > -1 then begin
        Values2[c.fAgg2ID]:=Values2[c.fAgg2ID]+Discount(Cost,Y-1,-1);

Values2p[c.fAgg2ID]:=Values2p[c.fAgg2ID]+Discount(Cost,Y-1,CostingData.CostCapital);
        Values2Y[y,c.fAgg2ID] := Values2Y[y,c.fAgg2ID] + Cost;
      end;
      if c.fAgg2IDH > -1 then begin
        Values2[c.fAgg2IDH]:=Values2[c.fAgg2IDH]+Hours;
        Values2p[c.fAgg2IDH]:=Values2p[c.fAgg2IDH]+Hours;
      end;
      if c.fAgg2IDL > -1 then begin
        Values2[c.fAgg2IDL]:=Values2[c.fAgg2IDL]+Discount(Labor,Y-1,-1);

Values2p[c.fAgg2IDL]:=Values2p[c.fAgg2IDL]+Discount(Labor,Y-1,CostingData.CostCapita
l);
        Values2Y[y,c.fAgg2IDL] := Values2Y[y,c.fAgg2IDL] + Labor;
      end;
      if c.fAgg2IDO > -1 then begin
        Values2[c.fAgg2IDO]:=Values2[c.fAgg2IDO]+Discount(OM,Y-1,-1);

Values2p[c.fAgg2IDO]:=Values2p[c.fAgg2IDO]+Discount(OM,Y-1,CostingData.CostCapital);
        Values2Y[y,c.fAgg2IDO] := Values2Y[y,c.fAgg2IDO] + OM;
      end;

      // aggregate ICR categories
      if Y = 1 then begin
        if C.fAggICR_IDC1 > -1 then ValuesICR[C.fAggICR_IDC1] :=
ValuesICR[C.fAggICR_IDC1] + OM;
        if C.fAggICR_IDH1 > -1 then ValuesICR[C.fAggICR_IDH1] :=
ValuesICR[C.fAggICR_IDH1] + Hours;
      end
      else if Y = 2 then begin
        if C.fAggICR_IDC2 > -1 then ValuesICR[C.fAggICR_IDC2] :=
ValuesICR[C.fAggICR_IDC2] + OM;
        if C.fAggICR_IDH2 > -1 then ValuesICR[C.fAggICR_IDH2] :=
ValuesICR[C.fAggICR_IDH2] + Hours;
      end
      else if Y = 3 then begin
        if C.fAggICR_IDC3 > -1 then ValuesICR[C.fAggICR_IDC3] :=
ValuesICR[C.fAggICR_IDC3] + OM;
        if C.fAggICR_IDH3 > -1 then ValuesICR[C.fAggICR_IDH3] :=
ValuesICR[C.fAggICR_IDH3] + Hours;
      end
      else if ((Y >= 4) and (Y <= 9)) then begin
        if C.fAggICR_IDC4 > -1 then ValuesICR[C.fAggICR_IDC4] :=
ValuesICR[C.fAggICR_IDC4] + OM;
        if C.fAggICR_IDH4 > -1 then ValuesICR[C.fAggICR_IDH4] :=
ValuesICR[C.fAggICR_IDH4] + Hours;
      end
      else if (Y >= 10) then begin
        if C.fAggICR_IDC10 > -1 then ValuesICR[C.fAggICR_IDC10] :=
ValuesICR[C.fAggICR_IDC10] + OM;
        if C.fAggICR_IDH10 > -1 then ValuesICR[C.fAggICR_IDH10] :=
ValuesICR[C.fAggICR_IDH10] + Hours;
      end;
    end;  // end C in CostSteps loop

    // POTW cost
    if CCTCostEquations.pbasepo4 = 1 then
    begin
      if (AdjustCCT[y] and (cct_adjust_yr = Y)) or
         (InstallCCT[y] and (cct_install_yr = Y)) then
        prob_downstream_P_limit :=  calc_prob_downstream_P_limit(isBaseline, Y);

      if prob_downstream_P_limit = 1 then
      begin
        PDose := CCTCostEquations.arrBaselineP[CCTCostEquations.iBaselinepo4dose];
        FlowLossP := (CCTCostEquations.AFlowEP*VLSEpWorkbook.NumberEPs) * PDose *
10893.71;
        ConnectionLossP := VLSEpWorkbook.Connections * PDose * 0.86;
        POTWCost := POTWCost + Discount((FlowLossP -
ConnectionLossP),Y-1,CostingData.CostCapital);
      end;

      if (CCTB = 1) and (CCTCostEquations.pbasepo4 + CCTCostEquations.pbasephpo4 >
0) then
      begin
        if Y = 5 then
          prerule_ploading_lbs_5 := (0.775 *
CCTCostEquations.arrBaselinepo4dose[CCTCostEquations.iBaselinepo4dose] *
CCTCostEquations.AFlowEP * 1000) -
                                    (0.061 * VLSEpWorkbook.Connections *
CCTCostEquations.arrBaselinepo4dose[CCTCostEquations.iBaselinepo4dose])
        else
        if Y = 15 then
          prerule_ploading_lbs_15 := (0.775 *
CCTCostEquations.arrBaselinepo4dose[CCTCostEquations.iBaselinepo4dose] *
CCTCostEquations.AFlowEP * 1000) -
                                     (0.061 * VLSEpWorkbook.Connections *
CCTCostEquations.arrBaselinepo4dose[CCTCostEquations.iBaselinepo4dose])
        else
        if Y = 25 then
          prerule_ploading_lbs_25 := (0.775 *
CCTCostEquations.arrBaselinepo4dose[CCTCostEquations.iBaselinepo4dose] *
CCTCostEquations.AFlowEP * 1000) -
                                     (0.061 * VLSEpWorkbook.Connections *
CCTCostEquations.arrBaselinepo4dose[CCTCostEquations.iBaselinepo4dose])
        else
        if Y = 35 then
          prerule_ploading_lbs_35 := (0.775 *
CCTCostEquations.arrBaselinepo4dose[CCTCostEquations.iBaselinepo4dose] *
CCTCostEquations.AFlowEP * 1000) -
                                     (0.061 * VLSEpWorkbook.Connections *
CCTCostEquations.arrBaselinepo4dose[CCTCostEquations.iBaselinepo4dose]);
      end;

      if (b_install_cct[y] + b_modify_cct[y] > 0) and (CCTCostEquations.pbasepo4 +
CCTCostEquations.pbasephpo4 > 0) then
      begin
        if Y = 5 then
          postrule_ploading_lbs_5 := (0.775 * 3.2 * CCTCostEquations.AFlowEP * 1000)
-
                                     (0.061 * VLSEpWorkbook.Connections * 3.2)
        else
        if Y = 15 then
          postrule_ploading_lbs_15 := (0.775 * 3.2 * CCTCostEquations.AFlowEP *
1000) -
                                      (0.061 * VLSEpWorkbook.Connections * 3.2)
        else
        if Y = 25 then
          postrule_ploading_lbs_25 := (0.775 * 3.2 * CCTCostEquations.AFlowEP *
1000) -
                                      (0.061 * VLSEpWorkbook.Connections * 3.2)
        else
        if Y = 35 then
          postrule_ploading_lbs_35 := (0.775 * 3.2 * CCTCostEquations.AFlowEP *
1000) -
                                      (0.061 * VLSEpWorkbook.Connections * 3.2);
      end
      else
      begin
        if Y = 5 then
          postrule_ploading_lbs_5 := prerule_ploading_lbs_5
        else
        if Y = 15 then
          postrule_ploading_lbs_15 := prerule_ploading_lbs_15
        else
        if Y = 25 then
          postrule_ploading_lbs_25 := prerule_ploading_lbs_25
        else
        if Y = 35 then
          postrule_ploading_lbs_35 := prerule_ploading_lbs_35;
      end;

      incr_ploading_lbs_5 := postrule_ploading_lbs_5 - prerule_ploading_lbs_5;
      incr_ploading_lbs_15 := postrule_ploading_lbs_15 - prerule_ploading_lbs_15;
      incr_ploading_lbs_25 := postrule_ploading_lbs_25 - prerule_ploading_lbs_25;
      incr_ploading_lbs_35 := postrule_ploading_lbs_35 - prerule_ploading_lbs_35;

      if incr_ploading_lbs_5 > 0 then
        count_incr_ploading_lbs_5 := 1;
      if incr_ploading_lbs_15 > 0 then
        count_incr_ploading_lbs_15 := 1;
      if incr_ploading_lbs_25 > 0 then
        count_incr_ploading_lbs_25 := 1;
      if incr_ploading_lbs_35 > 0 then
        count_incr_ploading_lbs_35 := 1;
    end;

    // read data request data values from database for next year
    fCostVars.FillValueArray(Variables, RawVariables,
                           CostingData.SystemSize, CostingData.SourceWater,
VLSEpWorkbook.LSL,
                           VLSEpWorkbook.CCT, CostingData.SystemType, Y+1,
Config.PWS90PctBp1, Config.PWS90PctBp2, SetProbsTo01, NewDraw, nil, isBaseline);
```

```
    Variables[fI.p_lsl] := LSL;

    // load external variables values
    Variables[fI.EP] := VLSEpWorkbook.NumberEPs;
    Variables[fI.Pws_Cct] := VLSEpWorkbook.CCT;

    Variables[fI.Pws_sw] := 0;
    Variables[fI.Pws_gw] := 0;
    if CostingData.SourceWater = 2 then
      Variables[fI.Pws_sw] := 1
    else if CostingData.SourceWater = 1 then
      Variables[fI.Pws_gw] := 1;

    Variables[fI.Pws_pop] := VLSEpWorkbook.Population;

    InitCCTBVarsToZero(option);

    if FindAndFix then
      Variables[fI.b_findfix] := 1;

    if  (option <> 'Baseline') then
    begin
      if AddCostingData.Num_Proxies = 0 then
      begin
        if Round(Config.DiscountRate*100)/100 = 0.03 then
          Variables[fI.annual_pou_cost_hh] := 111
        else if Round(Config.DiscountRate*100)/100 = 0.07 then
          Variables[fI.annual_pou_cost_hh] := 114
        else
          Variables[fI.annual_pou_cost_hh] := -1;
      end
      else
        Variables[fI.annual_pou_cost_hh] := 114;

      if Config.VolLeadProg = 0 then
        Variables[fI.p_vol_leadtap_prog] := 0;
    end;

    bCCT_Change := (b_install_cct[y] + b_modify_cct[y] > 0);
    if isBaseline then
      LeadConcentrationBins(CostingData.pwsid, option, y, CostingData.SystemSize,
CostingData.SourceWater,
                            LSL, CCT, POU, VLSEpWorkbook.Population,
VLSEpWorkbook.Connections,
                            fAdjust_CCT, fInstall_CCT, cct_adjust_yr,
cct_install_yr, pou_install_yr,
                            bCCT_Change,
                            CostingData.SamplingWeight, num_lsl_replace[y],
num_lsl_requested[y], num_lsl_remain, AddCostingData.Num_Proxies, partial_cct_level,
                            hp_lslr_paper, hp_lslr_partial,
pp_lsl_replacement_rates, false)
    else
      LeadConcentrationBins(CostingData.pwsid, option, y, CostingData.SystemSize,
CostingData.SourceWater,
                            LSL, CCT, POU, VLSEpWorkbook.Population,
VLSEpWorkbook.Connections,
                            fAdjust_CCT, fInstall_CCT, cct_adjust_yr,
cct_install_yr, pou_install_yr,
                            bCCT_Change,
                            CostingData.SamplingWeight, num_lsl_replace[y],
num_lsl_requested[y], num_lsl_remain, AddCostingData.Num_Proxies, partial_cct_level,
                            hp_lslr_paper, hp_lslr_partial,
pp_lsl_replaced_vol_pct_yr, false);

    if AddCostingData.Num_Proxies = 0 then
    begin
      Config.PWSBinCount[CostingData.SystemType, y, CostingData.SystemSize,
CostingData.SourceWater, owBin] :=
          Config.PWSBinCount[CostingData.SystemType, y, CostingData.SystemSize,
CostingData.SourceWater, owBin] + CostingData.SamplingWeight;
    end;
  end; // end year loop

  LSLReplaced := LSLReplacedMandatory + LSLReplacedVoluntary;

  //Annualize(CostingData.CostCapital);
end;

procedure TCostingSteps.StateCostsCalculate(const SetProbsTo01: boolean; option:
string);
var C: TCostingStep;
    Cost,Labor,OM,Hours,DoIt : double;
    Y : integer;
    NewDraw : boolean;
    IsBaseline: boolean;
    v: double;
begin
  StateCost := 0;
  StateICRHours1 := 0;
  StateICRHours2 := 0;
  StateICRHours3 := 0;
  StateICRHours4 := 0;
  StateICRHours10 := 0;
  StateICRCost1 := 0;
  StateICRCost2 := 0;
  StateICRCost3 := 0;
  StateICRCost4 := 0;
  StateICRCost10 := 0;
  NewDraw := false;

  if option = 'Baseline' then IsBaseline := true
  else IsBaseline := false;


  fCostVars.FillValueArray(Variables, RawVariables,
0,0,0,0,0,1,Config.PWS90PctBp1,Config.PWS90PctBp2, SetProbsTo01, NewDraw,nil,
IsBaseline);

  for Y := 1 to fYears do
  begin

    for C in CostSteps do begin
      if C.fCostStepRec.Domain <> 'State' then continue;
      if not C.arrCalculateYr[Y] then continue;
      C.EvaluateState(Cost,Labor,OM,Hours,DoIt);

      StateCost := StateCost + Discount(Cost,Y-1,-2) ;

      if Y = 1 then
      begin
        StateICRCost1 := StateICRCost1 + OM;
        StateICRHours1 := StateICRHours1 + Hours;
      end
      else if Y = 2 then
      begin
        StateICRCost2 := StateICRCost2 + OM;
        StateICRHours2 := StateICRHours2 + Hours;
      end
      else if Y = 3 then
      begin
        StateICRCost3 := StateICRCost3 + OM;
        StateICRHours3 := StateICRHours3 + Hours;
      end
      else if ((Y >= 4) and (Y <= 9)) then
      begin
        StateICRCost4 := StateICRCost4 + OM;
        StateICRHours4 := StateICRHours4 + Hours;
      end
      else if (Y >= 10) then
      begin
        StateICRCost10 := StateICRCost10 + OM;
        StateICRHours10 := StateICRHours10 + Hours;
      end;
    end;
  end;

  // Annualize
  v:=(DiscRate / (1 - IntPower((1 + DiscRate),-fYears)));
  StateCost := StateCost * v;
end;

{ TCostingStep }

procedure TCostingStep.AddVars(s: string);
var i : integer;
    ns : string;
    t : TStringList;
begin
  //dumb get variables out of string....
  ns:='';
  for i:=1 to length(s) do begin
    if CharInSet(s[i],['a'..'z','A'..'Z','0'..'9','_']) then
      ns:=ns+s[i]
    else
      ns:=ns+' ';
  end;
  while pos('  ',ns)>0 do
    ns:=StringReplace(ns,'  ',' ',[rfReplaceAll]);
  ns:=StringReplace(ns,' ',',',[rfReplaceAll]);
  t:=TStringLIst.Create;
  t.CommaText:=ns;
  for i:=0 to t.Count-1 do begin
    if ((length(t[i])>0) and (fMyVars.IndexOf(t[i])<0)) then begin
     if not CharInSet(t[i][1],['0'..'9']) then fMyVars.Add(t[i]);
    end;
  end;
  t.Free;
end;

constructor TCostingStep.Create(aCostStepRec: TCostStepRec; aCostVars : TCostVars;
aP : TParser;
                                IsBaseline: boolean; aCC : TLSRCompiledCost);
var i: integer;
begin
  fCC := aCC;
  fCostStepRec:=aCostStepRec;
  fImAStateCost := fCostStepRec.Domain='State';
  fCCTCost:=false;
  fCostStepRec.LSLRCost:=AnsiIndexText(fCostStepRec.CostName,
                         ['system_lslr_one','system_lslr_two','system_lslr_three',
                          'hh_lslr_one','hh_lslr_two','hh_lslr_three',
                          'system_lslr','hh_lslr','system_pou',
                          'system_lslr_vol','system_lslr_bin1',
                          'hh_lslr_vol','hh_lslr_bin1']) > -1;

  if (fCostStepRec.CostName='system_install_cct_copper_ale') or
     (fCostStepRec.CostName='system_install_cct_lead_ale') or
     (fCostStepRec.CostName='system_install_cct_source') or
     (fCostStepRec.CostName='system_install_cct_treat') or
     (fCostStepRec.CostName='system_adjust_cct_copper_ale') or
     (fCostStepRec.CostName='system_adjust_cct_guide') or
     (fCostStepRec.CostName='system_adjust_cct_source') or
     (fCostStepRec.CostName='system_adjust_cct_treat') or
     (fCostStepRec.CostName='system_adjust_cct_wqp') or

     (fCostStepRec.CostName='system_install_cct_leadagg') or
     (fCostStepRec.CostName='system_install_cct_sale_one') or
     (fCostStepRec.CostName='system_install_cct_sale_two') or
     (fCostStepRec.CostName='system_install_cct_sale_three') or
     (fCostStepRec.CostName='system_install_cct_ca') or
     (fCostStepRec.CostName='system_adjust_cct_sale_one') or
     (fCostStepRec.CostName='system_adjust_cct_sale_two') or
     (fCostStepRec.CostName='system_adjust_cct_sale_three') or
     (fCostStepRec.CostName='system_adjust_cct_ca') or

     (fCostStepRec.CostName='system_install_cct_sale_lrg_lsl') or
     (fCostStepRec.CostName='system_install_cct_sale_smmed_one') or
     (fCostStepRec.CostName='system_install_cct_sale_smmed_two') or
     (fCostStepRec.CostName='system_install_cct_sale_smmed_three') or
     (fCostStepRec.CostName='system_adjust_cct_sale_lrg_lsl') or
     (fCostStepRec.CostName='system_adjust_cct_sale_smmed_one') or
     (fCostStepRec.CostName='system_adjust_cct_sale_smmed_two') or
     (fCostStepRec.CostName='system_adjust_cct_sale_smmed_three') or

     (fCostStepRec.CostName='system_install_cct_lead_ale_one') or
     (fCostStepRec.CostName='system_install_cct_lead_ale_two') or
     (fCostStepRec.CostName='system_install_cct_lead_ale_three') or
     (fCostStepRec.CostName='system_adjust_cct_guide') or
     (fCostStepRec.CostName='system_adjust_cct_guide_five') or
     (fCostStepRec.CostName='system_adjust_cct_sanitary') or
     (fCostStepRec.CostName='system_adjust_cct_sanitary_five') or

     (fCostStepRec.CostName='system_install_cct') or
     (fCostStepRec.CostName='system_modify_cct') or
     (fCostStepRec.CostName='system_modify_cct_al') or
     (fCostStepRec.CostName='system_modify_cct_tl') or
     (fCostStepRec.CostName='system_mod_install_cct_lead_ale')

      then fCCTCost:=true;
```

```
  fBaselineCCTInstall := false;
  if (fCostStepRec.CostName='system_install_cct_lead_ale') or
     (fCostStepRec.CostName='system_install_cct_copper_ale') or
     (fCostStepRec.CostName='system_install_cct_source') or
     (fCostStepRec.CostName='system_install_cct_treat') then fBaselineCCTInstall :=
true;

  fBaselineCCTModify := false;
  if (fCostStepRec.CostName='system_adjust_cct_copper_ale') or
     (fCostStepRec.CostName='system_adjust_cct_source') or
     (fCostStepRec.CostName='system_adjust_cct_treat') or
     (fCostStepRec.CostName='system_mod_install_cct_lead_ale') then
fBaselineCCTModify := true;

  fBabyBlueCCTInstall := false;
  if (fCostStepRec.CostName='system_install_cct_lead_ale_one') or
     (fCostStepRec.CostName='system_install_cct_lead_ale_two') or
     (fCostStepRec.CostName='system_install_cct_lead_ale_three') or
     (fCostStepRec.CostName='system_install_cct_copper_ale') or
     (fCostStepRec.CostName='system_install_cct_source') or
     (fCostStepRec.CostName='system_install_cct_treat') then fBabyBlueCCTInstall :=
true;

  fBabyBlueCCTModify := false;
  if (fCostStepRec.CostName='system_adjust_cct_guide') or
     (fCostStepRec.CostName='system_adjust_cct_copper_ale') or
     (fCostStepRec.CostName='system_adjust_cct_source') or
     (fCostStepRec.CostName='system_adjust_cct_treat') or
     (fCostStepRec.CostName='system_adjust_cct_guide_five') or
     (fCostStepRec.CostName='system_adjust_cct_sanitary') or
     (fCostStepRec.CostName='system_adjust_cct_sanitary_five') then
fBabyBlueCCTModify := true;

  fSysLSLRCapital := false;
  if (fCostStepRec.CostName='system_lslr') or
     (fCostStepRec.CostName='system_lslr_one') or
     (fCostStepRec.CostName='system_lslr_two') or
     (fCostStepRec.CostName='system_lslr_three') or
     (fCostStepRec.CostName='system_lslr_vol') or
     (fCostStepRec.CostName='system_lslr_bin1') then fSysLSLRCapital := true;

  fHhLSLRCapital := false;
  if (fCostStepRec.CostName='hh_lslr') or
     (fCostStepRec.CostName='hh_lslr_one') or
     (fCostStepRec.CostName='hh_lslr_two') or
     (fCostStepRec.CostName='hh_lslr_three') or
     (fCostStepRec.CostName='hh_lslr_vol') or
     (fCostStepRec.CostName='hh_lslr_bin1') then fHhLSLRCapital := true;
```

```
fSysCCTCapital := false;

fOWCCTInstall := false;
if (fCostStepRec.CostName='system_install_cct') then fOWCCTInstall := true;

fOWCCTModify := false;
if (fCostStepRec.CostName='system_modify_cct_tl') or
   (fCostStepRec.CostName='system_modify_cct_al') then fOWCCTModify := true;

fOWCCTModify_ale := false;
if (fCostStepRec.CostName='system_modify_cct_al') then fOWCCTModify_ale := true;

fOWCCTModify_tle := false;
if (fCostStepRec.CostName='system_modify_cct_tl') then fOWCCTModify_tle := true;

fOneTimeCost := false;

if (fCostStepRec.CostName='system_plan_lslr') or
   (fCostStepRec.CostName='state_confer_lslr') or
   (fCostStepRec.CostName='state_temp_sop') or
   (fCostStepRec.CostName='system_develop_sop') or
   (fCostStepRec.CostName='state_temp_outreach_lslr') or
   (fCostStepRec.CostName='system_consult_outreach_lslr') or
   (fCostStepRec.CostName='state_review_lslr_pe') or
   (fCostStepRec.CostName='system_pe_lead_one') or
   (fCostStepRec.CostName='state_pe_lead_one') or
   (fCostStepRec.CostName='system_collect_sw_ale') or
   (fCostStepRec.CostName='system_analyze_sw_ale') or
   (fCostStepRec.CostName='system_report_sw_ale') or
   (fCostStepRec.CostName='state_review_sw_ale') or
   (fCostStepRec.CostName='system_collect_sw_ale_b1') or
   (fCostStepRec.CostName='system_analyze_sw_ale_b1') or
   (fCostStepRec.CostName='system_report_sw_ale_b1') or
   (fCostStepRec.CostName='state_review_sw_ale_b1') or
   (fCostStepRec.CostName='system_devel_pe_pou') or
   (fCostStepRec.CostName='system_collect_tap_pou_proactive') or
   (fCostStepRec.CostName='system_analyze_tap_pou_proactive') or
   (fCostStepRec.CostName='system_inform_tap_pou_proactive') or

   (fCostStepRec.CostName='system_cct_study_lead') or
   (fCostStepRec.CostName='state_cct_rec_lcr_lead') or
   (fCostStepRec.CostName='state_cct_rec_nostudy_lead') or
   (fCostStepRec.CostName='system_discuss_cct_lead') or
   (fCostStepRec.CostName='system_cct_monitor_lead') or
   (fCostStepRec.CostName='system_reject_cct_lead') or
   (fCostStepRec.CostName='system_cct_samp_lead') or
   (fCostStepRec.CostName='system_cct_invalid_lead') or
```

```
(fCostStepRec.CostName='state_cct_invalid_lead') or
(fCostStepRec.CostName='system_cct_samp_inform_lead') or
(fCostStepRec.CostName='system_collect_wqp_cct_lead') or
(fCostStepRec.CostName='system_analyze_wqp_cct_lead') or
(fCostStepRec.CostName='system_collect_wqp_ep_cct_lead') or
(fCostStepRec.CostName='system_analyze_wqp_ep_cct_lead') or
(fCostStepRec.CostName='system_report_ep_wqp_lead') or
(fCostStepRec.CostName='state_review_wqp_ep_cct_lead') or
(fCostStepRec.CostName='state_cct_owqps_lead') or
(fCostStepRec.CostName='system_submit_wq_lead_inst') or

(fCostStepRec.CostName='system_pe_devel_sal') or
(fCostStepRec.CostName='state_review_pe') or
(fCostStepRec.CostName='state_cct_study_lead') or
(fCostStepRec.CostName='system_collect_wqp_cct_lead_ph') or
(fCostStepRec.CostName='system_analyze_wqp_cct_lead_ph') or
(fCostStepRec.CostName='system_collect_wqp_ep_cct_lead_ph') or
(fCostStepRec.CostName='system_analyze_wqp_ep_cct_lead_ph') or
(fCostStepRec.CostName='system_collect_wqp_cct_lead_ortho') or
(fCostStepRec.CostName='system_analyze_wqp_cct_lead_ortho') or
(fCostStepRec.CostName='system_collect_wqp_ep_cct_lead_ortho') or
(fCostStepRec.CostName='system_analyze_wqp_ep_cct_lead_ortho') or
(fCostStepRec.CostName='system_devel_lslr') or
(fCostStepRec.CostName='system_devel_prior_lslr') or
(fCostStepRec.CostName='system_cct_samp_inform_lead_ntncws') or

(fCostStepRec.CostName='state_mod_cct_study_lead') or
(fCostStepRec.CostName='system_mod_cct_study_lead') or
(fCostStepRec.CostName='state_mod_cct_rec_lcr_lead') or
(fCostStepRec.CostName='state_mod_cct_rec_nostudy_lead') or
(fCostStepRec.CostName='system_mod_discuss_cct_lead') or
(fCostStepRec.CostName='system_mod_cct_monitor_lead') or
(fCostStepRec.CostName='system_mod_reject_cct_lead') or
(fCostStepRec.CostName='system_mod_cct_samp_lead') or
(fCostStepRec.CostName='system_mod_cct_invalid_lead') or
(fCostStepRec.CostName='state_mod_cct_invalid_lead') or
(fCostStepRec.CostName='system_mod_cct_samp_inform_lead') or
(fCostStepRec.CostName='system_mod_collect_wqp_cct_lead_ph') or
(fCostStepRec.CostName='system_mod_analyze_wqp_cct_lead_ph') or
(fCostStepRec.CostName='system_mod_collect_wqp_ep_cct_lead_ph') or
(fCostStepRec.CostName='system_mod_analyze_wqp_ep_cct_lead_ph') or
(fCostStepRec.CostName='system_mod_report_ep_wqp_lead') or
(fCostStepRec.CostName='state_mod_review_wqp_ep_cct_lead') or
(fCostStepRec.CostName='system_mod_collect_wqp_cct_lead_ortho') or
(fCostStepRec.CostName='system_mod_analyze_wqp_cct_lead_ortho') or
(fCostStepRec.CostName='system_mod_collect_wqp_ep_cct_lead_ortho') or
(fCostStepRec.CostName='system_mod_analyze_wqp_ep_cct_lead_ortho') or
(fCostStepRec.CostName='state_mod_cct_owqps_lead') or
```

```
     (fCostStepRec.CostName='system_mod_submit_wq_lead_inst') or
     (fCostStepRec.CostName='state_cct_study_revise') or
     (fCostStepRec.CostName='system_revise_cct_mod') or
     (fCostStepRec.CostName='state_review_cct_plan_mod') or
     (fCostStepRec.CostName='state_cct_rec_nostudy_modify') or
     (fCostStepRec.CostName='state_owqp_cct_modify') or
     (fCostStepRec.CostName='system_cct_study_source_inst') or
     (fCostStepRec.CostName='state_cct_rec_lcr_source_inst') or
     (fCostStepRec.CostName='state_owqp_cct_install_wocct') or
     (fCostStepRec.CostName='system_goal_lslr') or
     (fCostStepRec.CostName='state_goal_lslr') or
     (fCostStepRec.CostName='system_plan_pou') or
     (fCostStepRec.CostName='state_confer_pou') or
     (fCostStepRec.CostName='state_pe_pou_review') or

     (fCostStepRec.CostName='system_hrs_collect_sw_ale') or
     (fCostStepRec.CostName='state_collect_sw_ale') or
     (fCostStepRec.CostName='state_analyze_sw_ale') or
     (fCostStepRec.CostName='state_report_sw_ale') or
     (fCostStepRec.CostName='system_hrs_collect_sw_ale_b1') or
     (fCostStepRec.CostName='state_collect_sw_ale_b1') or
     (fCostStepRec.CostName='state_analyze_sw_ale_b1') or
     (fCostStepRec.CostName='state_report_sw_ale_b1') or
     (fCostStepRec.CostName='system_pe_devel_sal') or
     (fCostStepRec.CostName='state_review_pe') or

     (fCostStepRec.CostName='state_cct_study_revise_tl') or
     (fCostStepRec.CostName='state_cct_study_revise_al') or
     (fCostStepRec.CostName='system_submit_wq_lead_inst_tl') or
     (fCostStepRec.CostName='system_submit_wq_lead_inst_al') or
     (fCostStepRec.CostName='system_revise_cct_mod_tl') or
     (fCostStepRec.CostName='system_revise_cct_mod_al') or
     (fCostStepRec.CostName='state_review_cct_plan_mod_tl') or
     (fCostStepRec.CostName='state_review_cct_plan_mod_al') or
     (fCostStepRec.CostName='state_cct_rec_nostudy_modify_tl') or
     (fCostStepRec.CostName='state_cct_rec_nostudy_modify_al') or
     (fCostStepRec.CostName='state_owqp_cct_modify_tl') or
     (fCostStepRec.CostName='state_owqp_cct_modify_al') or
     (fCostStepRec.CostName='system_plan_lslr_vol') or
     (fCostStepRec.CostName='system_plan_lslr_mand') or
     (fCostStepRec.CostName='state_temp_pe_pou')

    then fOneTimeCost := true;

  if fCostStepRec.CostName='system_install_cct_leadagg' then
    fDEBUG:=true;
  fCostVars:=aCostVars;
  P := aP;
  fMyVars:=TStringList.create;
  fMyVars.Sorted:=true;
  fMyVars.Duplicates:=dupIgnore;
  //Remove equations for Non-ICR rows...
  if fCostStepRec.ICRRow <> 'Y' then begin
    fCostStepRec.Hours:='';
  end;

    fCostStepRec.Labor:='';

  fError:='';
  pSetup(pCost,fCostStepRec.TotalCost,fOKC);
  pSetup(pIn,fCostStepRec.ProbabilityCost,fOKP);
  pSetup(pHours,fCostStepRec.Hours,fOKH);
  pSetup(pOM,fCostStepRec.OM,fOKO);
  pSetup(pLabor,fCostStepRec.Labor,fOKL);
  fOkAll:=((fOKP) or (length(trim(fCostStepRec.ProbabilityCost))<1)) and
          ((fOKC) or (length(trim(fCostStepRec.TotalCost))<1)) and
          ((fOKH) or (length(trim(fCostStepRec.Hours))<1)) and
          ((fOKO) or (length(trim(fCostStepRec.OM))<1)) and
          ((fOKL) or (length(trim(fCostStepRec.Labor))<1));

  for i := 1 to high(farrCalculateYr) do
    arrCalculateYr[i] := false;

  InitArrCalculateYr(IsBaseline);
  fEvalCount:=0;
end;

destructor TCostingStep.Destroy;
begin
  fMyVars.Free;
  inherited;
end;

procedure TCostingStep.Evaluate(var Cost, Labor, OM, Hours, DoIt: double);
begin
  Cost:=0; Labor:=0; OM:=0; Hours:=0;DoIt:=1;
  if not fOKAll then exit;
  if fImAStateCost  then exit;
  if fOKP then begin
    DoIt:=Convert(p.Execute(pin)^, vtDouble).Float64;
    inc(fEvalCount);
    if DoIt<1 then exit;
  end;

  if fOKC then begin
    Cost:=Convert(p.Execute(pCost)^, vtDouble).Float64;
    inc(fEvalCount);
  end;
  if fOKL then begin
    Labor:=Convert(p.Execute(pLabor)^, vtDouble).Float64;
    inc(fEvalCount);
  end;
  if fOKO then begin
    OM:=Convert(p.Execute(pOM)^, vtDouble).Float64;
    inc(fEvalCount);
  end;
  if fOKH then begin
    Hours:=Convert(p.Execute(pHours)^, vtDouble).Float64;
    inc(fEvalCount);
  end;
end;

procedure TCostingStep.EvaluateState(var Cost, Labor, OM, Hours, DoIt: double);
begin
  Cost:=0; Labor:=0; OM:=0; Hours:=0;DoIt:=1;
  if not fOKAll then exit;
  if not fImAStateCost  then exit;
  if fOKP then begin
    DoIt:=Convert(p.Execute(pin)^, vtDouble).Float64;
    if DoIt<1 then exit;
  end;
  if fOKC then
    Cost:=Convert(p.Execute(pCost)^, vtDouble).Float64;
  if fOKL then
    Labor:=Convert(p.Execute(pLabor)^, vtDouble).Float64;
  if fOKO then
    OM:=Convert(p.Execute(pOM)^, vtDouble).Float64;
  if fOKH then
    Hours:=Convert(p.Execute(pHours)^, vtDouble).Float64;
end;

procedure TCostingStep.InitArrCalculateYr(IsBaseline: boolean);
var
  i, j, k: integer;
  s: string;
  iYr, jYr, rYr: integer;
  iSep: integer;
  iYrs: array[1..100] of integer;
begin
  { year examples:
      3        set arrCalculateYr[3] = true
      3-7      set arrCalculateYr[3] - arrCalculateYr[7] = true
      3:7      choose a year between 3 and 7 and set to true
      3,6,9    set years 3, 7 and 9 to true
      6;12;18  choose one of these years and set it to true
  }
  if TryStrToInt(fCostStepRec.Year, iYr) then
    arrCalculateYr[iYr] := true
  else
  begin
    iSep := AnsiPos( '-', fCostStepRec.Year);
    if iSep > 0 then
    begin
      iYr := StrToInt(Copy(fCostStepRec.Year, 1, iSep-1));
      jYr := StrToInt(Copy(fCostStepRec.Year, iSep+1,
length(fCostStepRec.Year)-iSep));
      for i := iYr to jYr do
        arrCalculateYr[i] := true;
    end
    else
    begin
      iSep := AnsiPos(':', fCostStepRec.Year);
      if iSep > 0 then
      begin
        iYr := StrToInt(Copy(fCostStepRec.Year, 1, iSep-1));
        jYr := StrToInt(Copy(fCostStepRec.Year, iSep+1,
Length(fCostStepRec.Year)-iSep));
        // changed to iiRandomRange 4/6/18
        rYr := iiRandomRange(iYr, jYr, IsBaseline);
        arrCalculateYr[rYr] := true;
      end
      else
      begin
        if AnsiContainsStr(fCostStepRec.Year, ',') then
        begin
          s := '';
          for i := 1 to Length(fCostStepRec.Year) do
          begin
            if (fCostStepRec.Year[i] = ',') then
            begin
              arrCalculateYr[StrToInt(s)] := true;
              s := '';
              continue;
            end;
            s := s + fCostStepRec.Year[i];
          end;
          arrCalculateYr[StrToInt(s)] := true;
        end
        else
        begin
          if AnsiContainsStr(fCostStepRec.Year, ';') then
          begin
            s := '';
            j := 1;
            for i := 1 to Length(fCostStepRec.Year) do
            begin
              if (fCostStepRec.Year[i] = ';') then
              begin
                iYrs[j] := StrToInt(s);
                s := '';
                inc(j);
                continue;
              end;
              s := s + fCostStepRec.Year[i];
            end;
            iYrs[j] := StrToInt(s);
            // changed to iiRandomRange 4/6/18
            k := iiRandomRange(1,j, IsBaseline);
            arrCalculateYr[iYrs[k]] := true;
          end;
        end;
      end;
    end;
  end;
end;

procedure TCostingStep.pSetup(var TS : TScript; s: string; var fOK : boolean);
var ss : string;
begin
  ss:=trim(s);
  if ss = ' ' then ss := '';
  fOK := false;
  if length(ss)<1 then
    exit;
  try
    P.StringToScript(ss,TS);
    fOK:=true;
  except
    on e:exception do begin
      fError:=fError + e.Message + '   Eq:'+s+#13#10;
      fOk:=False;
    end;
  end;
end;

procedure TCostingStep.ResetArrCalculateYr(IsBaseline : boolean);
var
  i, j, k: integer;
  s: string;
  iYr, jYr, rYr: integer;
  iSep: integer;
  iYrs: array[1..100] of integer;
begin
  { year examples:
      3        set arrCalculateYr[3] = true
      3-7      set arrCalculateYr[3] - arrCalculateYr[7] = true
      3:7      choose a year between 3 and 7 and set to true
      3,6,9    set years 3, 7 and 9 to true
      6;12;18  choose one of these years and set it to true
  }
  iSep := AnsiPos(':', fCostStepRec.Year);
  if iSep > 0 then begin
    for i := 1 to high(farrCalculateYr) do
      arrCalculateYr[i] := false;
    iYr := StrToInt(Copy(fCostStepRec.Year, 1, iSep-1));
    jYr := StrToInt(Copy(fCostStepRec.Year, iSep+1,
Length(fCostStepRec.Year)-iSep));
    rYr := iiRandomRange(iYr, jYr, isBaseline);
    arrCalculateYr[rYr] := true;
  end else
  if AnsiContainsStr(fCostStepRec.Year, ';') then begin
    for i := 1 to high(farrCalculateYr) do
      arrCalculateYr[i] := false;
    s := '';
    j := 1;
    for i := 1 to Length(fCostStepRec.Year) do begin
      if (fCostStepRec.Year[i] = ';') then begin
        iYrs[j] := StrToInt(s);
        s := '';
        inc(j);
        continue;
      end;
      s := s + fCostStepRec.Year[i];
    end;
    iYrs[j] := StrToInt(s);
    k := iiRandomRange(1,j, isBaseline);
    arrCalculateYr[iYrs[k]] := true;
  end;
end;

function TCostingStep.getArrCalculateYr(Index: integer): boolean;
begin
  result := fArrCalculateYr[Index];
end;

procedure TCostingStep.SetArrCalculateYr(Index: integer; const Value: boolean);
begin
  fArrCalculateYr[Index] := Value;
  DirArrCalculateYr[Index] := value;
  fCC._YearOK[Index,fCostStepRec.ID] := value;
end;

end.
```