```
unit CostingSteps;

interface

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

type
  tFilterHelp = record
    pwsid: string;
    aNC, num_sl_unknown,pop_per_connection,
    perc_unknown_nonlead_b,
    num_unknown_resolved,
    num_lsl_replaced,
    pp_lslr_lsl_adj
    : double;
  end;


  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: 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;
    fRandomStream: integer;

    fLastPWSID, fLastPWSID2: string;
    CostCounted: 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 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_dssa_cost: integer;
```

```
    cct_modify_cost_umra, cct_install_cost_umra, cct_dssa_cost_umra: integer;
    cct_modify_cost_umra_om, cct_install_cost_umra_om, cct_dssa_cost_umra_om:
integer;
    cct_modify_cost_p, cct_install_cost_p, cct_dssa_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_lsl, pp_lslr_partial, pp_lslr_gal_prev_lsl, pp_lslr_leadcon,
      pp_lslr_galprev_leadcon: 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: 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_dssa: 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_dssa,
numb_reduced_wqp,
      numb_enhance_wqp: integer;

    num_lsl_remain, 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, pp_grandfather_opt_pub_elem,
      pp_grandfather_opt_priv_elem, pp_grandfather_mand_pub_elem,
pp_grandfather_mand_priv_elem,
      pp_grandfather_opt_child, pp_grandfather_mand_child,
pp_grandfather_opt1_pub_second,
      pp_grandfather_opt2_pub_second: integer;
    pp_grandfather_opt1_priv_second, pp_grandfather_opt2_priv_second: integer;

    b_state_one, b_state_two: integer;
    num_lslr_lsl_replace, num_lslr_partial_replace, num_lslr_gal_prev_lsl_replace,
```

```
      num_lslr_galprev_leadcon_replace, num_lslr_leadcon_replace, num_lslr_replace:
integer;
    num_temp_pou: integer;
    p_90_concurrent_15, p_90_concurrent_12, p_90_concurrent_10, p_90_concurrent_5:
integer;
    pp90aboveal15_1, pp90aboveal15_2, pp90aboveal15_3, pp90aboveal15_4,
pp90aboveal15_5: integer;
    pp90aboveal12_1, pp90aboveal12_2, pp90aboveal12_3, pp90aboveal12_4,
pp90aboveal12_5: integer;
    pp90aboveal10_1, pp90aboveal10_2, pp90aboveal10_3, pp90aboveal10_4,
pp90aboveal10_5: integer;
    pp90aboveal5_1, pp90aboveal5_2, pp90aboveal5_3, pp90aboveal5_4, pp90aboveal5_5:
integer;
    pbin1, pbin2, pbin3, pbin4, pbin5: integer;
    padjbin1, padjbin2, padjbin3, padjbin4, padjbin5: integer;

    p_wqp_all, p_wqp_cond: integer;
    b_temp_pou_1, b_temp_pou_2, b_temp_pou_3, b_temp_pou_4: integer;
    pp_lsl, pp_lsl_unknown, pp_lsl_nolead_unknown, pws_lsl: integer;
    perc_lsl_known, perc_lsl_known_lead, perc_lsl_unknown, perc_lsl_unknown_lead,
      perc_lsl_nolead_unknown_unknown: integer;

    num_lsl_testout, num_unknown_resolved, num_lsl_validated, num_unknown_remain,
      hh_unknown_remain: integer;
    num_lsl_filter, pp_pou_adopt: integer;

    hrs_adopt_rule_js, hrs_modify_ds_js, hrs_initial_ta_js, hrs_train_imp_js,
hrs_coord_epa_js,
      hrs_ta_js, hrs_sdwis_js, hrs_train_ann_js, rate_js: integer;

    numb_ntncws_filters: integer;
  end;

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

    fPA: array of double;
    fPI: array of double;

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

    fAdjust_CCT, fInstall_CCT: integer;

    GBinA: array [1..32] of double;
```

```
    GBin: array [1..32] of double;
    FH: TFilterHelp;
    fRandomStream: integer;

    fnum_proxies: integer;
    TotalPWSCost: double;

    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: 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 RemoveFilters(yr: integer);
    procedure SetNewFilters(yr: integer);
    procedure SetInitFilters(yr: integer);

    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,
      perc_lsl_b: double; Num_Proxies, partial_cct_level: integer;
      pp_lslr_lsl, pp_lslr_lsl_adj: double; num_lsl_filters,
num_hh_per_connect,num_temp_pou : double;
      ResetBins: boolean = false);
    function calc_prob_downstream_P_limit(baseline: boolean; Year: integer):
integer;
    function get_category_value(category: string): double;
    procedure MakeLCRxBin(aOption: string; aALE: integer; aIBin: integer;
      var vLCRxBin, vLCRxP90: integer);
    procedure Up(ff, tt, yy: integer; pp: double);
    function GetTotalPWSCost(const sA: TArray<string>; const CostStepRec:
TCostStepRec; Cost: double): double;
  public
    CostSteps: TObjectList<TCostingStep>;
    AggInputs2, AggInputs1, AggICR: TArray<string>;

    // holds current values of all variables for steps..
```

```
    Variables: TDoubleArray;
    RawVariables: TDoubleArray;
    fI: TCustomVarIdxs;

    // 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, POTWCostVLS: 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;
    StateCostUndiscounted: 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;
    CCTInstalled: boolean;
    CCTAdjusted, CCTAdjusted_ale, CCTAdjusted_tle: boolean;
    CCTExisting: boolean;
    HasFindAndFixCost: boolean;
    POUInstalled: boolean;
    LSLRequested, LSLRequestedVLS: double;
```

```
// These variables are for the very large systems
HasLSLRCostVLS: boolean;
HasCCTCostVLS: boolean;

LSLReplacedVLS: double;
LSLReplacedPopVLS: double;
LSLReplacedMandatoryVLS: double;
LSLReplacedVoluntaryVLS: double;
LSLReplacedPopMandatoryVLS: double;
LSLReplacedPopVoluntaryVLS: double;
LSLRequestedPopVLS: double;

LSLReplaceFullMandatoryVLS: double;
LSLReplaceFullPopMandatoryVLS: double;
LSLReplacePartialMandatoryVLS: double;
LSLReplacePartialPopMandatoryVLS: double;
LSLGalPrevMandatoryVLS: double;
LSLGalPrevPopMandatoryVLS: double;
LSLLeadConMandatoryVLS: double;
LSLLeadConPopMandatoryVLS: double;
LSLGalPrevLeadConMandatoryVLS: double;
LSLGalPrevLeadConPopMandatoryVLS: double;

LSLReplaceFullVoluntaryVLS: double;
LSLReplaceFullPopVoluntaryVLS: double;
LSLReplacePartialVoluntaryVLS: double;
LSLReplacePartialPopVoluntaryVLS: double;
LSLGalPrevVoluntaryVLS: double;
LSLGalPrevPopVoluntaryVLS: double;
LSLLeadConVoluntaryVLS: double;
LSLLeadConPopVoluntaryVLS: double;
LSLGalPrevLeadConVoluntaryVLS: double;
LSLGalPrevLeadConPopVoluntaryVLS: double;

LSLReplacedNSVLS: double;
LSLReplacedPopNSVLS: double;
LSLReplacedMandatoryNSVLS: double;
LSLReplacedVoluntaryNSVLS: double;
LSLReplacedPopMandatoryNSVLS: double;
LSLReplacedPopVoluntaryNSVLS: double;
LSLRequestedNSVLS: double;
LSLRequestedPopNSVLS: double;

LSLReplaceFullMandatoryNSVLS: double;
LSLReplaceFullPopMandatoryNSVLS: double;
LSLReplacePartialMandatoryNSVLS: double;
LSLReplacePartialPopMandatoryNSVLS: double;
LSLGalPrevMandatoryNSVLS: double;
LSLGalPrevPopMandatoryNSVLS: double;
```

```
LSLLeadConMandatoryNSVLS: double;
LSLLeadConPopMandatoryNSVLS: double;
LSLGalPrevLeadConMandatoryNSVLS: double;
LSLGalPrevLeadConPopMandatoryNSVLS: double;

LSLReplaceFullVoluntaryNSVLS: double;
LSLReplaceFullPopVoluntaryNSVLS: double;
LSLReplacePartialVoluntaryNSVLS: double;
LSLReplacePartialPopVoluntaryNSVLS: double;
LSLGalPrevVoluntaryNSVLS: double;
LSLGalPrevPopVoluntaryNSVLS: double;
LSLLeadConVoluntaryNSVLS: double;
LSLLeadConPopVoluntaryNSVLS: double;
LSLGalPrevLeadConVoluntaryNSVLS: double;
LSLGalPrevLeadConPopVoluntaryNSVLS: 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;
ToBin1Count: double;

SumFedLSLReplace: double;
SumFedLSLReplaceMand: double;
SumFedLSLPartialReplaceMand: double;
SumFedLSLGalPrevMand: double;
SumFedLSLLeadConMand: double;
SumFedLSLGalPrevLeadConMand: double;
SumFedLSLReplaceVol: double;
SumFedLSLPartialReplaceVol: double;
SumFedLSLGalPrevVol: double;
SumFedLSLLeadConVol: double;
SumFedLSLGalPrevLeadConVol: double;

SumTotLSLReplace: double;
SumTotLSLReplaceMand: double;
SumTotLSLPartialReplaceMand: double;
SumTotLSLGalPrevMand: double;
SumTotLSLLeadConMand: double;
SumTotLSLGalPrevLeadConMand: double;
SumTotLSLReplaceVol: double;
SumTotLSLPartialReplaceVol: double;
SumTotLSLGalPrevVol: double;
```

```
    SumTotLSLLeadConVol: double;
    SumTotLSLGalPrevLeadConVol: 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, GMoveBinMicroTotal: TYearlyMovementMicro;
    GMoveBinTotal, GMoveBinTotalAllYears : TMovementMicro;
    CC: TLSRCompiledCost;

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

    deferral_any, deferral_cap, deferral_pct: double;
    deferral_anyvls, deferral_capvls, deferral_pctvls: double;
    pubNewPath: 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 CostingData: TCostGenRec; 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;
      const pws_state: string);

    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 pws_state: string; 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; const IsBaseline: boolean;
      const have_lsl: boolean; 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: integer;
    function AssignNewBinOption(option: string): integer;

    procedure OutputBenefitMoveBins(aName : string);

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

implementation

uses VCL.FlexCel.Core, FlexCel.XlsAdapter;

{ 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: 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;

  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.GetTotalPWSCost(const sA: TArray<string>; const CostStepRec:
TCostStepRec;
  Cost: double): double;
var i : integer;
begin
  Result:=0;
```

```
  if MatchText(CostStepRec.Agglomerator2Xw, sA) then
      Result := Cost;
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 numbering 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.Up(ff, tt, yy: integer; pp: double);
begin
  var p := Round(pp * 10) / 10;
  GMoveBinMicro[yy, ff, tt] := GMoveBinMicro[yy, ff, tt] + p;
  GMoveBinTotal[ff, tt] := GMoveBinTotal[ff, tt] + p;
  GMoveBinTotalAllYears[ff, tt] := GMoveBinTotalAllYears[ff, tt] + p;
  GMoveBinMicroTotal[yy, ff, tt] := GMoveBinMicroTotal[yy, ff, tt] + p;
end;

procedure TCostingSteps.RemoveFilters(yr: integer);
begin
  if (Config.OptionName <> 'LCRI') or (fIsBaseLine) then exit;
  var t,n: double;
  var i: integer;
  if (Config.LSLFilters = 1) and (Config.TempPOUOption = 1) and
(Variables[fI.b_temp_pou_1] = 1) then begin
    for i:=1 to  16 do begin
      if i in [1,4,13] then begin
        t := GBinA[i+16];
        if t=0 then continue;
        n :=  min(t, ( FH.pop_per_connection * FH.num_lsl_replaced *
FH.pp_lslr_lsl_adj));
        Up(i+16,i, yr, n);
        GBinA[i] := GBinA[i] + n;
        GBinA[i+16] := GBinA[i+16] - n;
      end;
      if i in [2,7,14] then begin
        t := GBinA[i+16];
        if t=0 then continue;
        n :=  min(t, ( FH.pop_per_connection * FH.num_lsl_replaced * ( 1 -
FH.pp_lslr_lsl_adj)));
```

```
        Up(i+16,i, yr, n);
        GBinA[i] := GBinA[i] + n;
        GBinA[i+16] := GBinA[i+16] - n;
      end;
    end;
  end;

  if (Config.LSLFilters = 1) or ((Config.TempPOUOption = 2) and
(Variables[fI.b_temp_pou_2] = 1))
      or ((Config.TempPOUOption = 3) and (Variables[fI.b_temp_pou_3] = 1))then begin
    for i := 1 to 16 do begin
      if i in [1,4] then begin
        t:= GBinA[i+16];
        if t=0 then continue;
        n :=  min(t, ( FH.pop_per_connection * FH.num_lsl_replaced *
FH.pp_lslr_lsl_adj));
        Up(i+16,i, yr, n);
        GBinA[i] := GBinA[i] + n;
        GBinA[i+16] := GBinA[i+16] - n;
      end;
      if i in [2,7] then begin
        t := GBinA[i+16];
        if t=0 then continue;
        n :=  min(t, ( FH.pop_per_connection * FH.num_lsl_replaced * ( 1 -
FH.pp_lslr_lsl_adj)));
        Up(i+16,i, yr, n);
        GBinA[i] := GBinA[i] + n;
        GBinA[i+16] := GBinA[i+16] - n;
      end;
    end;
  end;

  //looks like same condition as above
  if (Config.LSLFilters = 1) or ((Config.TempPOUOption = 2) and
(Variables[fI.b_temp_pou_2] = 1))
      or ((Config.TempPOUOption = 3) and (Variables[fI.b_temp_pou_3] = 1)) then
begin
    for i := 1 to 16  do begin
      if not i in [3,10] then continue;
      t := GBinA[i+16];
      if t=0 then continue;
      n :=  min(t, FH.num_unknown_resolved * (1-
Variables[fI.perc_lsl_unknown_lead]) * FH.pop_per_connection);
      Up(i+16,i, yr, n);
      GBinA[i] := GBinA[i] + n;
      GBinA[i+16] := GBinA[i+16] - n;
    end;
  end;

  if (Config.LSLFilters = 0) and (
```

```
     ((Config.TempPOUOption = 2) and (Variables[fI.b_temp_pou_2] = 0)) or
     ((Config.TempPOUOption = 1) and (Variables[fI.b_temp_pou_1] = 0)) or
     ((Config.TempPOUOption = 3) and (Variables[fI.b_temp_pou_3] = 0))
     ) then begin

    for i:=1 to  16 do begin
      t := GBinA[i+16];
      if t=0 then continue;
      Up(i+16,i, yr, t);
      GBinA[i] := GBinA[i] + t;
      GBinA[i+16] := GBinA[i+16] - t;
    end;

  end;
end;

procedure TCostingSteps.SetNewFilters(yr: integer);
begin
  var t: double;
  if (Config.OptionName <> 'LCRI') or (fIsBaseLine) then exit;

  if (Config.LSLFilters = 1) then begin
    for var i := 1 to 16  do begin
      if i in [13,14] then begin
        Up(i, i+16, yr, GBinA[i]);
        GBinA[i+16] := GBinA[i+16] + GBinA[i];
        GBinA[i] := 0;
      end;

      if i in [15] then begin
        t:= GBinA[i] * FH.perc_unknown_nonlead_b;
        Up(i, i+16, yr, t);
        GBinA[i+16] := GBinA[i+16] + t;
        GBinA[i] := GBinA[i] - t;
      end;
    end;
  end;
end;

procedure TCostingSteps.SetInitFilters(yr: integer);
begin
  var t: double;
  var i: integer;
  if (Config.OptionName <> 'LCRI') or (fIsBaseLine) then exit;

  if (Config.LSLFilters = 1) or (Variables[fI.b_temp_pou_2] = 1) then begin
    for i := 1 to 16 do begin
      if i in [1,2,4,7] then begin
        Up(i, i+16, yr, GBinA[i]);
        GBinA[i+16] := GBinA[i+16] + GBinA[i];
```

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

      if i in [3,10] then begin
        t:= GBinA[i] * FH.perc_unknown_nonlead_b;
        Up(i, i+16, yr, t);
        GBinA[i+16] := GBinA[i+16] + t;
        GBinA[i] := GBinA[i] - t;
      end;
    end;
  end;  //else????

  if (Variables[fI.b_temp_pou_1] = 1) then begin
    for i:=1 to 16 do begin
      Up(i, i+16, yr, GBinA[i]);
      GBinA[i+16] := GBinA[i+16] + GBinA[i];
      GBinA[i] := 0;
    end;
  end;

  if (Config.LSLFilters = 0) and (Variables[fI.b_temp_pou_3] = 1) then begin
    for i := 1 to 16 do begin
      if i in [1,2,4,7] then begin
        t:= GBinA[i] * Variables[fI.pp_pou_adopt];
        Up(i, i+16, yr, t);
        GBinA[i+16] := GBinA[i+16] + t;
        GBinA[i] := GBinA[i] - t;
      end;

      if i in [3,10] then begin
        t:= GBinA[i] * FH.perc_unknown_nonlead_b * Variables[fI.pp_pou_adopt];
        Up(i, i+16, yr, t);
        GBinA[i+16] := GBinA[i+16] + t;
        GBinA[i] := GBinA[i] - t;
//lll.L('SetInit_'+Config.RunName, FH.pwsid + ',' +
FH.perc_unknown_nonlead_b.ToString + ',' + Variables[fI.pp_pou_adopt].ToString + ','
+GBinA[i].ToString + ',' + i.ToString );
      end;
    end;
  end;
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, perc_lsl_b: double;
  Num_Proxies, partial_cct_level: integer; pp_lslr_lsl, pp_lslr_lsl_adj: double;
  num_lsl_filters, num_hh_per_connect, num_temp_pou: double;
  ResetBins: boolean = false);
```

```
var
  pop_per_connection: double;
  pp_lsl_replaced: double;
  sLine: string;
  GB1, GB2, GB3, GB4, GB5, GB6, GB7, GB8, GB9, Gb10, GB11, GB12, GB13, GB14, GB15,
GB16,
  fGB1, fGB2, fGB3, fGB4, fGB5, fGB6, fGB7, fGB8, fGB9, fGb10, fGB11, fGB12, fGB13,
fGB14,
  fGB15, fGB16: double;

  i, y, f, T: integer;
  GBSum: double;
  num_lsl_replaced: double;

  cpp_lsl_replaced, hcpp_lsl_replaced, zpp_lsl_replaced: double;

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_replaced2 = num_lsl_replaced_fed[y]
  num_lsl_replaced := num_lsl_replaced2 + num_lsl_requested;

  //safe to set it here - needed for temp filter stuff
  FH.num_lsl_replaced := num_lsl_replaced;

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

  pp_lsl_replaced := 0;

  if yr = 0 then begin
    hcpp_lsl_replaced := 0;


    // Before the year loop, calculate for each PWS:
    // NOTE: Need to update benefits to differentiate between full and other types
```

of lead content

```
    // Bin 1: No CCT and Full LSL
    if CCT = 1 then
      GBinA[1] := 0
    else if aLSL = 0 then
      GBinA[1] := 0
    else
      GBinA[1] := aPop * perc_lsl_b * pp_lslr_lsl;

    // BIN 2: No CCT and Non-Full LSL
    if CCT = 1 then
      GBinA[2] := 0
    else if aLSL = 0 then
      GBinA[2] := 0
    else
      GBinA[2] := aPop * perc_lsl_b * (1 - pp_lslr_lsl);

    // 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_b));

    // BIN 4: Partial CCT and Full LSL
    if CCT = 0 then
      GBinA[4] := 0
    else if aLSL = 0 then
      GBinA[4] := 0
    else
      GBinA[4] := aPop * perc_lsl_b * pp_lslr_lsl;

    GBinA[5] := 0;

    GBinA[6] := 0;

    // BIN 7: Partial CCT and Non-Full LSL
    if CCT = 0 then
      GBinA[7] := 0
    else if aLSL = 0 then
      GBinA[7] := 0
    else
      GBinA[7] := aPop * perc_lsl_b * (1 - pp_lslr_lsl);

    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_b));

  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;

  //temp bins for Bins 1-16
  for i := 17 to 32 do GBinA[i] :=0;

  // move shadow bins into bins that the benefits calculation will use
  for i := 1 to 32 do begin
    GBin[i] := GBinA[i];
    if (GBin[i] > -0.01) and (GBin[i] < 0) then begin
      GBin[i] := 0;
      GBinA[i] := 0;
    end;
  end;
end else begin // Every year calculate
  if (yr=4) then SetInitFilters(yr);
  if (yr>4) then SetNewFilters(yr);
  if (yr>4) then RemoveFilters(yr);

  // BIN 1: No CCT, No POU and Full 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 * pp_lslr_lsl_adj);
```

```
    if (CCT = 1) and (GB1 > 0) then begin
      Up(1, 13, yr, GB1 - (pop_per_connection * num_lsl_replaced *
pp_lslr_lsl_adj));
      Up(1, 15, yr, (pop_per_connection * num_lsl_replaced * pp_lslr_lsl_adj));
    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 * pp_lslr_lsl_adj));
    end;

    // BIN 2: No CCT, No POU and Non-Full 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 * (1 -
pp_lslr_lsl_adj));

    if (CCT = 1) and (GB2 > 0) then begin
      Up(2, 14, yr, GB2 - (pop_per_connection * num_lsl_replaced * (1 -
pp_lslr_lsl_adj)));
      Up(2, 15, yr, (pop_per_connection * num_lsl_replaced * (1 -
pp_lslr_lsl_adj)));
    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 * (1 - pp_lslr_lsl_adj)));
    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);  // should be

    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 Full 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 * pp_lslr_lsl_adj);

    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 *
pp_lslr_lsl_adj));
      Up(4, 15, yr, (pop_per_connection * num_lsl_replaced * pp_lslr_lsl_adj));
    end
    else if (bCCT_Change = false) and (GB4 > 0) then begin
      Up(4, 10, yr, (pop_per_connection * num_lsl_replaced * pp_lslr_lsl_adj));
    end;

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

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

    // BIN 7: Partial CCT, No POU and Non-Full 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 * (1 -
pp_lslr_lsl_adj));

    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 * (1 -
pp_lslr_lsl_adj)));
      Up(7, 15, yr, (pop_per_connection * num_lsl_replaced * (1 -
pp_lslr_lsl_adj)));
    end
    else if (bCCT_Change = false) and (GB7 > 0) then begin
      Up(7, 10, yr, (pop_per_connection * num_lsl_replaced * (1 -
pp_lslr_lsl_adj)));
    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 Full 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 * pp_lslr_lsl_adj)
    else if cct_adjust_yr = yr then
      GBinA[13] := (GB4 + GB5 + GB6) - (pop_per_connection * num_lsl_replaced *
pp_lslr_lsl_adj)
```

```
    else if GB13 = 0 then
      GBinA[13] := 0
    else
      GBinA[13] := GB13 - (pop_per_connection * num_lsl_replaced * pp_lslr_lsl_adj);

    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 * pp_lslr_lsl_adj));
    end;

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

    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 * (1 -
pp_lslr_lsl_adj)));
    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 32 do begin
      GBin[i] := GBinA[i];
      if (GBin[i] > -0.01) and (GBin[i] < 0) then begin
        GBin[i] := 0;
        GBinA[i] := 0;
      end;
    end;
  end;

  GBSum := 0;
  for i := 1 to 32 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 + ',' + GBin[17].ToString + ',' + GBin[18].ToString + ','
+
       GBin[19].ToString + ',' + GBin[20].ToString + ','
       + GBin[21].ToString + ',' + GBin[22].ToString + ',' +
       GBin[23].ToString + ',' + GBin[24].ToString + ',' + GBin[25].ToString + ','
+
       GBin[26].ToString + ',' + GBin[27].ToString + ',' + GBin[28].ToString + ','
+
       GBin[29].ToString + ',' + GBin[30].ToString + ',' + GBin[31].ToString+ ','
       + GBin[32].ToString + ',' + GBSum.ToString
       + ',' + aLSL.ToString + ',' + CCT.ToString + ',' +
```

```
pop_per_connection.ToString + ',' +
        num_lsl_replaced.ToString + ',' + cct_adjust_yr.ToString + ',' +
cct_install_yr.ToString +
        ',' + perc_lsl_b.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.MakeLCRxBin(aOption: string; aALE, aIBin: integer;
  var vLCRxBin, vLCRxP90: integer);
var
  LCRI_BIN_15, LCRI_BIN_12, LCRI_BIN_10, LCRI_BIN_5: integer;
begin
  vLCRxBin := -1;
  vLCRxP90 := -1;

  if aOption = 'LCRI' then begin
    if aIBin <= 1 then
      LCRI_BIN_15 := 1;
    if aIBin <= 2 then
      LCRI_BIN_12 := 1;
    if aIBin <= 3 then
      LCRI_BIN_10 := 1;
    if aIBin <= 4 then
      LCRI_BIN_5 := 1;

    if aALE = 15 then begin
      if LCRI_BIN_15 = 1 then
        vLCRxBin := 1
      else
        vLCRxBin := 3;

      if vLCRxBin = 1 then
        vLCRxP90 := 20
      else
        vLCRxP90 := 10;
    end
    else if aALE = 12 then begin
      if LCRI_BIN_12 = 1 then
        vLCRxBin := 1
      else
        vLCRxBin := 3;

      if vLCRxBin = 1 then
        vLCRxP90 := 15
      else
        vLCRxP90 := 8;
```

```
    end
    else if aALE = 10 then begin
      if LCRI_BIN_10 = 1 then
        vLCRxBin := 1
      else
        vLCRxBin := 3;

      if vLCRxBin = 1 then
        vLCRxP90 := 12
      else
        vLCRxP90 := 5;
    end
    else if aALE = 5 then begin
      if LCRI_BIN_5 = 1 then
        vLCRxBin := 1
      else
        vLCRxBin := 3;

      if vLCRxBin = 1 then
        vLCRxP90 := 10
      else
        vLCRxP90 := 2;
    end;
  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: integer;
var
  r: double;
  i, iBin: integer;
  tmp: double;
  arrPBin: array [1 .. 5] of double;
begin
  r := iiRandom(rrBase);

  arrPBin[1] := Variables[fI.pbin1];
  arrPBin[2] := Variables[fI.pbin2];
  arrPBin[3] := Variables[fI.pbin3];
  arrPBin[4] := Variables[fI.pbin4];
  arrPBin[5] := Variables[fI.pbin5];

  iBin := 0;
  tmp := 0;
  i := 1;
  repeat
    tmp := tmp + arrPBin[i];
    iBin := i;
    Inc(i);
    if i > Length(arrPBin) then
      break;
  until tmp >= r;

  if iBin = 1 then
    iBin := 1
  else if (iBin >= 2) and (iBin <= 3) then
    iBin := 2
  else
    iBin := 3;
```

```
  Result := iBin;
end;

function TCostingSteps.AssignNewBinOption(option: string): integer;
var
  r: double;
  i, iBin: integer;
  tmp: double;
  arrPAdjBin: array [1 .. 5] of double;
begin
  r := iiRandom(rrScen);

  arrPAdjBin[1] := Variables[fI.padjbin1];
  arrPAdjBin[2] := Variables[fI.padjbin2];
  arrPAdjBin[3] := Variables[fI.padjbin3];
  arrPAdjBin[4] := Variables[fI.padjbin4];
  arrPAdjBin[5] := Variables[fI.padjbin5];

  if option = 'LCRI' then begin
    iBin := 0;
    tmp := 0;
    i := 1;
    repeat
      tmp := tmp + arrPAdjBin[i];
      iBin := i;
      Inc(i);
      if i > Length(arrPAdjBin) 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);
  if baseline then
    r := iiRandom(rrBase)
  else
    r := iiRandom(rrScen);
  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;
    if baseline then
      r := iiRandom(rrBase)
    else
      r := iiRandom(rrScen);
    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);
    if baseline then
      r := iiRandom(rrBase)
    else
      r := iiRandom(rrScen);
    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
    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(fRandomStream);
    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(fRandomStream);
    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
```

```
      for i := 0 to 6 do
        fPI[i] := fPI[i] / tp;
    end;
    r := iiRandom(fRandomStream);
    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;
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
  begin
    if pos('LCRR',Filename) > 0 then
      CC := TLSRCompiledCostBaselineLCRR.Create
    else
      CC := TLSRCompiledCostBaselineLCR.Create;
  end
  else
    CC := TLSRCompiledCostOption.Create;
  if IsBaseline then
    fRandomStream := rrBase
  else
    fRandomStream := rrScen;

  fIsBaseline := IsBaseline;
  fErrors := TStringList.Create;
  fillchar(fI, SizeOf(fI), -1);
  fCostVars := aCostVars;
  CostSteps := TObjectList<TCostingStep>.Create();
  DiscRate := 0.03;
  setlength(Variables, fCostVars.Count);
  setlength(RawVariables, fCostVars.Count);
  setlength(fPA, 9);
  setlength(fPI, 7);

  fillchar(GBin, SizeOf(GBin), 0);
  fillchar(GMoveBinTotalAllYears, SizeOf(GMoveBinTotalAllYears), 0);
  setlength(GMoveBin, aYears + 1);
  setlength(GMoveBinMicro, aYears + 1);
  setlength(GMoveBinMicroTotal, 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;
        //NOTE:  assumes that this object can accumulate (1 call per year per pws
essentially)
        GMoveBinMicroTotal[y, f, tt] := 0;
      end;
    end;
  end;

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

  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_dssa_cost' then
  fI.cct_dssa_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_dssa_cost_umra' then
  fI.cct_dssa_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_dssa_cost_umra_om' then
  fI.cct_dssa_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_dssa_cost_p' then
  fI.cct_dssa_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_lsl' then
  fI.pp_lslr_lsl := ix
else if T.fID = 'pp_lslr_partial' then
  fI.pp_lslr_partial := ix
else if T.fID = 'pp_lslr_gal_prev_lsl' then
  fI.pp_lslr_gal_prev_lsl := ix
else if T.fID = 'pp_lslr_leadcon' then
  fI.pp_lslr_leadcon := ix
else if T.fID = 'pp_lslr_gal_prev_leadcon' then
  fI.pp_lslr_galprev_leadcon := 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 = '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_dssa' then
  fI.b_dssa := 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_dssa' then
  fI.pp_overlap_dssa := 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 = 'pp_grandfather_opt_pub_elem' then
  fI.pp_grandfather_opt_pub_elem := ix
else if T.fID = 'pp_grandfather_opt_priv_elem' then
  fI.pp_grandfather_opt_priv_elem := ix
```

```
else if T.fID = 'pp_grandfather_mand_pub_elem' then
  fI.pp_grandfather_mand_pub_elem := ix
else if T.fID = 'pp_grandfather_mand_priv_elem' then
  fI.pp_grandfather_mand_priv_elem := ix
else if T.fID = 'pp_grandfather_opt_child' then
  fI.pp_grandfather_opt_child := ix
else if T.fID = 'pp_grandfather_mand_child' then
  fI.pp_grandfather_mand_child := ix
else if T.fID = 'pp_grandfather_opt1_pub_second' then
  fI.pp_grandfather_opt1_pub_second := ix
else if T.fID = 'pp_grandfather_opt2_pub_second' then
  fI.pp_grandfather_opt2_pub_second := ix
else if T.fID = 'pp_grandfather_opt1_priv_second' then
  fI.pp_grandfather_opt1_priv_second := ix
else if T.fID = 'pp_grandfather_opt2_priv_second' then
  fI.pp_grandfather_opt2_priv_second := 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_lsl_remain' then
  fI.num_lsl_remain := ix
else if T.fID = 'num_lslr_lsl_replace' then
  fI.num_lslr_lsl_replace := ix
else if T.fID = 'num_lslr_replace' then
  fI.num_lslr_replace := ix
else if T.fID = 'num_lslr_partial_replace' then
  fI.num_lslr_partial_replace := ix
else if T.fID = 'num_lslr_leadcon_replace' then
  fI.num_lslr_leadcon_replace := ix
else if T.fID = 'num_lslr_gal_prev_lsl_replace' then
  fI.num_lslr_gal_prev_lsl_replace := ix
else if T.fID = 'num_lslr_gal_prev_leadcon_replace' then
  fI.num_lslr_galprev_leadcon_replace := ix
else if T.fID = 'num_temp_pou' then
  fI.num_temp_pou := ix
else if T.fID = 'p_90_concurrent_15' then
  fI.p_90_concurrent_15 := ix
else if T.fID = 'p_90_concurrent_12' then
  fI.p_90_concurrent_12 := ix
else if T.fID = 'p_90_concurrent_10' then
  fI.p_90_concurrent_10 := ix
else if T.fID = 'p_90_concurrent_5' then
  fI.p_90_concurrent_5 := ix
else

  if T.fID = 'pp90aboveal15_1' then
  fI.pp90aboveal15_1 := ix
else if T.fID = 'pp90aboveal15_2' then
  fI.pp90aboveal15_2 := ix
```

```
else if T.fID = 'pp90aboveal15_3' then
  fI.pp90aboveal15_3 := ix
else if T.fID = 'pp90aboveal15_4' then
  fI.pp90aboveal15_4 := ix
else if T.fID = 'pp90aboveal15_5' then
  fI.pp90aboveal15_5 := ix
else

  if T.fID = 'pp90aboveal12_1' then
  fI.pp90aboveal12_1 := ix
else if T.fID = 'pp90aboveal12_2' then
  fI.pp90aboveal12_2 := ix
else if T.fID = 'pp90aboveal12_3' then
  fI.pp90aboveal12_3 := ix
else if T.fID = 'pp90aboveal12_4' then
  fI.pp90aboveal12_4 := ix
else if T.fID = 'pp90aboveal12_5' then
  fI.pp90aboveal12_5 := ix
else

  if T.fID = 'pp90aboveal10_1' then
  fI.pp90aboveal10_1 := ix
else if T.fID = 'pp90aboveal10_2' then
  fI.pp90aboveal10_2 := ix
else if T.fID = 'pp90aboveal10_3' then
  fI.pp90aboveal10_3 := ix
else if T.fID = 'pp90aboveal10_4' then
  fI.pp90aboveal10_4 := ix
else if T.fID = 'pp90aboveal10_5' then
  fI.pp90aboveal10_5 := ix
else

  if T.fID = 'pp90aboveal5_1' then
  fI.pp90aboveal5_1 := ix
else if T.fID = 'pp90aboveal5_2' then
  fI.pp90aboveal5_2 := ix
else if T.fID = 'pp90aboveal5_3' then
  fI.pp90aboveal5_3 := ix
else if T.fID = 'pp90aboveal5_4' then
  fI.pp90aboveal5_4 := ix
else if T.fID = 'pp90aboveal5_5' then
  fI.pp90aboveal5_5 := ix
else

  if T.fID = 'pbin1' then
  fI.pbin1 := ix
else if T.fID = 'pbin2' then
  fI.pbin2 := ix
else if T.fID = 'pbin3' then
  fI.pbin3 := ix
```

```
else if T.fID = 'pbin4' then
  fI.pbin4 := ix
else if T.fID = 'pbin5' then
  fI.pbin5 := ix
else

  if T.fID = 'padjbin1' then
  fI.padjbin1 := ix
else if T.fID = 'padjbin2' then
  fI.padjbin2 := ix
else if T.fID = 'padjbin3' then
  fI.padjbin3 := ix
else if T.fID = 'padjbin4' then
  fI.padjbin4 := ix
else if T.fID = 'padjbin5' then
  fI.padjbin5 := ix
else

  if T.fID = 'p_wqp_all' then
  fI.p_wqp_all := ix
else if T.fID = 'p_wqp_cond' then
  fI.p_wqp_cond := ix
else

  if T.fID = 'b_temp_pou_1' then
  fI.b_temp_pou_1 := ix
else if T.fID = 'b_temp_pou_2' then
  fI.b_temp_pou_2 := ix
else if T.fID = 'b_temp_pou_3' then
  fI.b_temp_pou_3 := ix
else if T.fID = 'b_temp_pou_4' then
  fI.b_temp_pou_4 := ix
else

  if T.fID = 'pp_lsl' then
  fI.pp_lsl := ix
else if T.fID = 'pp_lsl_unknown' then
  fI.pp_lsl_unknown := ix
else if T.fID = 'pp_lsl_nolead_unknown' then
  fI.pp_lsl_nolead_unknown := ix
else if T.fID = 'pws_lsl' then
  fI.pws_lsl := ix
else

  if T.fID = 'perc_lsl_known' then
  fI.perc_lsl_known := ix
else if T.fID = 'perc_lsl_known_lead' then
  fI.perc_lsl_known_lead := ix
else if T.fID = 'perc_lsl_unknown' then
  fI.perc_lsl_unknown := ix
```

```
      else if T.fID = 'perc_lsl_unknown_lead' then
        fI.perc_lsl_unknown_lead := ix
      else if T.fID = 'perc_lsl_nolead_unknown_unknown' then
        fI.perc_lsl_nolead_unknown_unknown := ix
      else

        if T.fID = 'num_lsl_testout' then
        fI.num_lsl_testout := ix
      else if T.fID = 'num_unknown_resolved' then
        fI.num_unknown_resolved := ix
      else if T.fID = 'num_lsl_validated' then
        fI.num_lsl_validated := ix
      else if T.fID = 'num_lsl_filter' then
        fI.num_lsl_filter := ix
      else if T.fID = 'num_unknown_remain' then
        fI.num_unknown_remain := ix
      else if T.fID = 'hh_unknown_remain' then
        fI.hh_unknown_remain := ix
      else if T.fID = 'pp_pou_adopt' then
        fI.pp_pou_adopt := ix
      else if T.fID = 'hrs_adopt_rule_js' then
        fI.hrs_adopt_rule_js := ix
      else if T.fID = 'hrs_modify_ds_js' then
        fI.hrs_modify_ds_js := ix
      else if T.fID = 'hrs_initial_ta_js' then
        fI.hrs_initial_ta_js := ix
      else if T.fID = 'hrs_train_imp_js' then
        fI.hrs_train_imp_js := ix
      else if T.fID = 'hrs_coord_epa_js' then
        fI.hrs_coord_epa_js := ix
      else if T.fID = 'hrs_ta_js' then
        fI.hrs_ta_js := ix
      else if T.fID = 'hrs_sdwis_js' then
        fI.hrs_sdwis_js := ix
      else if T.fID = 'hrs_train_ann_js' then
        fI.hrs_train_ann_js := ix
      else if T.fID = 'rate_js' then
        fI.rate_js := ix
      else if T.fID = 'numb_ntncws_filters' then
        fI.numb_ntncws_filters := ix
  ;

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

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;


procedure TCostingSteps.OutputBenefitMoveBins(aName : string);
begin
  //save all total movements by year...
  if fIsBaseline then begin

OutputYearMovementBin(UserPath+aName+'_GMoveBinMicroTotal_baseline.csv',GMoveBinMicr
oTotal);

OutputMovementBin(UserPath+aName+'_GMoveBinMicroTotalAllYears_baseline.csv',GMoveBin
TotalAllYears);
  end else begin

OutputYearMovementBin(UserPath+aName+'GMoveBinMicroTotal.csv',GMoveBinMicroTotal);

OutputMovementBin(UserPath+aName+'GMoveBinMicroTotalAllYears.csv',GMoveBinTotalAllYe
ars);
  end;
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; const IsBaseline:
boolean;
          const have_lsl: boolean; GetBaseCurValue: boolean = false);
begin
  if IsBaseline then begin
    if not have_lsl then begin
```

```
      PWS_p_values.p_lsl := Round(fCostVars.Calculate_p_lsl(aSz, aSrc, aCCT, aType,
SetProbsTo01,
        GetBaseCurValue));
      if PWS_p_values.p_lsl = 0 then
        PWS_p_values.pws_lsl := 0
      else
        PWS_p_values.pws_lsl := 1;

      fCostVars.Calculate_pws_p_values(aSz, aSrc, PWS_p_values.pws_lsl, aCCT, aType,
SetProbsTo01,
                                       PWS_p_values, GetBaseCurValue);
    end
    else
      fCostVars.Calculate_pws_p_values(aSz, aSrc, aLSL, aCCT, aType, SetProbsTo01,
                                       PWS_p_values, GetBaseCurValue);
  end
  else
    fCostVars.Calculate_pws_p_values(aSz, aSrc, aLSL, 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), ci);
        // Inc(NumSteps);
```

```
        // 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), ci);
        // Inc(NumSteps);
        // 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;
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 CostingData: TCostGenRec;
  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; const pws_state: string);
var
  c: TCostingStep;
  Cost, Labor, OM, Hours, DoIt, v: double;
  y: integer;
  NewDraw: boolean;
  CCT, LSL, POU, pws_lsl: 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, FindAndFixCostCap2, FindAndFixCostCapDisc, AdjustCCTCostCap,
AdjustCCTCostCapDisc: double;
  AdjustCCTCostCapDisc_p, FindAndFixCostCapDisc_p: double;

  Num_LSL_base, Num_potential_LSL_base, Num_potential_LSL_known: double;
  num_sl_known: double;
  num_lsl_known, num_sl_unknown, num_lsl_unknown_lead, num_lsl_nolead_unknown:
double;
  perc_lsl_b, perc_unknown_nonlead_b: double;
  num_unknown_remain: double;

  pp_lsl_replaced, lslr_missed_goal: array [1 .. 100] of double;
  Num_lsl_replace_ale: array [1 .. 100] of double;
  Num_lsl_replace_state: array [1 .. 100] of double;
  Num_lsl_replace_fed: array [1 .. 100] of double;
  Num_lsl_replace_rule: array [1 .. 100] of double;
  Num_lsl_replace_tot: array [1 .. 100] of double;
  num_unknown_resolved: array [1 .. 100] of double;
  Num_lsl_replace_rule_all: array [1 .. 100] of double;
  num_lsl_testout: array [1 .. 100] of double;
  num_lsl_validated: array [1 .. 100] of double;
  Num_lsl_validate_pool: array [1 .. 100] of double;

    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;
  fed_num_lslr_lsl_replace: array[0..100] of double;
  fed_num_lslr_partial_replace: array[0..100] of double;
  fed_num_lslr_gal_prev_lsl_replace: array[0..100] of double;
  fed_num_lslr_leadcon_replace: array[0..100] of double;
  fed_num_lslr_galprev_leadcon_replace: array[0..100] of double;
  tot_num_lslr_lsl_replace: array[0..100] of double;
  tot_num_lslr_partial_replace: array[0..100] of double;
  tot_num_lslr_gal_prev_lsl_replace: array[0..100] of double;
  tot_num_lslr_leadcon_replace: array[0..100] of double;
  tot_num_lslr_galprev_leadcon_replace: array[0..100] of double;
  num_temp_pou: double;
  lsl_replace_pct: double;
  num_lsl_filters: double;

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

IsBaseline: boolean;

cct_adjust_yr, cct_install_yr, pou_install_yr, cct_change_yr: integer;

num_lsl_base_adjust: double;
partial_cct_level: integer;

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, proxy2_pws90_y4: 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;
BinChg: 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;

SourceTreatChangeEver: boolean;
lsl_start_yr: integer;

LCRI_Bin_initial: integer;
LCRI_Bin: integer;
LCRI_P90: integer;
LCRI_BIN_15, LCRI_BIN_12, LCRI_BIN_10, LCRI_BIN_5: integer;
```

```
  pp_lslr_lsl_adj: double;
  P90_concurrent: double;
  pp_above_al_bin_one, pp_above_al_bin_two, pp_above_al_bin_three: double;
  pp_lsl_replaced_mand_pct: double;
  pp_lsl_replaced_mand_pct_small: double;

  b_ale_not_achieved: integer;
  state_lslr_rate: double;

  temp_int: integer;
  lslr_newpath: integer;
  hh_unknown_remain: double;
  lsl_replace_pct_a, lsl_replace_pct_b, lsl_replace_pct_c: double;

  num_lsl_replace_y: double;
  num_unknown_resolved_y: double;
  Num_lsl_replace_rule_allb: double;
  CapitalCostNotAnnualized: double;

  procedure SetFH();
  begin
    FH.pwsid := pwsid;
    FH.aNC := aNC;
    FH.perc_unknown_nonlead_b := perc_unknown_nonlead_b;
    FH.num_sl_unknown := num_sl_unknown;
    if FH.aNC > 0 then
      FH.pop_per_connection := aPop / aNC
    else
      FH.pop_per_connection := 0;

    FH.num_lsl_replaced := num_lsl_replace_y;
    FH.pp_lslr_lsl_adj := pp_lslr_lsl_adj;
    FH.num_unknown_resolved := num_unknown_resolved_y;
  end;
begin
  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 := CostingData.SystemType;
```

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

if LSL > 0 then
  pws_lsl := 1
else
  pws_lsl := 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;

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

ToBin1Count := 0;
SumFedLSLReplace := 0;
```

```
SumFedLSLReplaceMand := 0;
SumFedLSLPartialReplaceMand := 0;
SumFedLSLGalPrevMand := 0;
SumFedLSLLeadConMand := 0;
SumFedLSLGalPrevLeadConMand := 0;

SumFedLSLReplaceVol := 0;
SumFedLSLPartialReplaceVol := 0;
SumFedLSLGalPrevVol := 0;
SumFedLSLLeadConVol := 0;
SumFedLSLGalPrevLeadConVol := 0;

SumTotLSLReplace := 0;
SumTotLSLReplaceMand := 0;
SumTotLSLPartialReplaceMand := 0;
SumTotLSLGalPrevMand := 0;
SumTotLSLLeadConMand := 0;
SumTotLSLGalPrevLeadConMand := 0;

SumTotLSLReplaceVol := 0;
SumTotLSLPartialReplaceVol := 0;
SumTotLSLGalPrevVol := 0;
SumTotLSLLeadConVol := 0;
SumTotLSLGalPrevLeadConVol := 0;

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

TotalPWSCost := 0;
CapitalCostNotAnnualized := 0;

cct_adjust_yr := 0;
cct_install_yr := 0;
cct_change_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(fed_num_lslr_lsl_replace, SizeOf(fed_num_lslr_lsl_replace), 0);
  fillchar(fed_num_lslr_partial_replace, SizeOf(fed_num_lslr_partial_replace), 0);
  fillchar(fed_num_lslr_gal_prev_lsl_replace,
SizeOf(fed_num_lslr_gal_prev_lsl_replace), 0);
  fillchar(fed_num_lslr_leadcon_replace, SizeOf(fed_num_lslr_leadcon_replace), 0);
  fillchar(fed_num_lslr_galprev_leadcon_replace,
SizeOf(fed_num_lslr_galprev_leadcon_replace), 0);
  fillchar(tot_num_lslr_lsl_replace, SizeOf(tot_num_lslr_lsl_replace), 0);
  fillchar(tot_num_lslr_partial_replace, SizeOf(tot_num_lslr_partial_replace), 0);
  fillchar(tot_num_lslr_gal_prev_lsl_replace,
SizeOf(tot_num_lslr_gal_prev_lsl_replace), 0);
  fillchar(tot_num_lslr_leadcon_replace, SizeOf(tot_num_lslr_leadcon_replace), 0);
  fillchar(tot_num_lslr_galprev_leadcon_replace,
```

```
SizeOf(tot_num_lslr_galprev_leadcon_replace), 0);

  perc_unknown_nonlead_b := 0;

  SourceTreatChangeEver := false;
  lsl_start_yr := 0;

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

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

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

fillchar(pp_lsl_replaced,SizeOf(pp_lsl_replaced),0);
fillchar(Num_lsl_replace_ale,SizeOf(Num_lsl_replace_ale),0);
fillchar(Num_lsl_replace_state,SizeOf(Num_lsl_replace_state),0);
fillchar(Num_lsl_replace_fed,SizeOf(Num_lsl_replace_fed),0);
fillchar(Num_lsl_replace_rule,SizeOf(Num_lsl_replace_rule),0);
fillchar(Num_lsl_replace_tot,SizeOf(Num_lsl_replace_tot),0);
//fillchar(num_unknown_remain,SizeOf(num_unknown_remain),0);
fillchar(num_unknown_resolved,SizeOf(num_unknown_resolved),0);
fillchar(Num_lsl_replace_rule_all,SizeOf(Num_lsl_replace_rule_all),0);
fillchar(lslr_missed_goal,SizeOf(lslr_missed_goal),0);
fillchar(num_lsl_requested,SizeOf(num_lsl_requested),0);
fillchar(Num_lsl_testout,SizeOf(Num_lsl_testout),0);
fillchar(num_lsl_validated,SizeOf(num_lsl_validated),0);
fillchar(num_lsl_validate_pool,SizeOf(num_lsl_validate_pool),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_potential_LSL_base := 0;
num_lsl_known := 0;
num_sl_unknown := 0;
num_lsl_nolead_unknown := 0;
num_unknown_remain := 0;
lslr_newpath := 0;
LSLRequested := 0;

bp1 := 10;

if option = 'Baseline' then begin
  bp1 := 10;
  bp2 := 15;

  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;

  if pws_state = 'MI' then begin
    if owBin = 1 then
      state_lslr_rate := 0.07
    else
      state_lslr_rate := 0.05;
  end
  else if pws_state = 'IL' then begin
    if num_lsl <= 1200 then
      state_lslr_rate := 0.07
    else if (num_lsl > 1200) and (num_lsl < 5000) then
      state_lslr_rate := 0.06
    else if (num_lsl >= 5000) and (num_lsl < 10000) then
      state_lslr_rate := 0.05
    else if (num_lsl > 10000) and (num_lsl < 100000) then
      state_lslr_rate := 0.03
    else if num_lsl > 100000 then
      state_lslr_rate := 0.02;
  end
  else if pws_state = 'NJ' then begin
    state_lslr_rate := 0.10;
  end
  else
    state_lslr_rate := 0;
end
else if option = 'LCRI' then begin
  bp2 := Config.ALE;

  O.TryGetValue('bBin', tmp_double);
  LCRI_Bin_initial := trunc(tmp_double);

  MakeLCRxBin(option, Config.ALE, LCRI_Bin_initial, LCRI_Bin, LCRI_P90);

  owBin := LCRI_Bin;
  pws90pct := LCRI_P90;

  O.TryGetValue('bALENotAchieved', tmp_double);
  b_ale_not_achieved := trunc(tmp_double);

  if pws_state = 'MI' then begin
    if owBin = 1 then
      state_lslr_rate := 0.07
    else
      state_lslr_rate := 0.05;
  end
  else if pws_state = 'IL' then begin
```

```
      if num_lsl <= 1200 then
        state_lslr_rate := 0.07
      else if (num_lsl > 1200) and (num_lsl < 5000) then
        state_lslr_rate := 0.06
      else if (num_lsl >= 5000) and (num_lsl < 10000) then
        state_lslr_rate := 0.05
      else if (num_lsl > 10000) and (num_lsl < 100000) then
        state_lslr_rate := 0.03
      else if num_lsl > 100000 then
        state_lslr_rate := 0.02;
    end
    else if pws_state = 'NJ' then begin
      state_lslr_rate := 0.10;
    end
    else
      state_lslr_rate := 0;
  end;

    if Num_Proxies = 0 then begin
      Config.PWSBinCount[CostingData.SystemType, 0, CostingData.SystemSize,
CostingData.SourceWater, owBin] :=
         Config.PWSBinCount[CostingData.SystemType, 0, CostingData.SystemSize,
CostingData.SourceWater, owBin] + 1;
    end;

  if LSL > 0 then begin
    // p_lsl = 1
    // have known lead content
    if LSL = 1 then begin
      num_sl_known := aNC * Variables[fI.perc_lsl_known];
      num_lsl_known := num_sl_known * Variables[fI.perc_lsl_known_lead];
      num_sl_unknown := aNC * (1 - Variables[fI.perc_lsl_known]);
      num_lsl_unknown_lead := num_sl_unknown * Variables[fI.perc_lsl_unknown_lead];
      perc_lsl_b := (Variables[fI.perc_lsl_known] *
Variables[fI.perc_lsl_known_lead]) +
                    ((1 - Variables[fI.perc_lsl_known]) *
Variables[fI.perc_lsl_unknown_lead]);
      Num_LSL_base := num_lsl_known + num_lsl_unknown_lead;
      if CostingData.SystemType = 1 then
        perc_unknown_nonlead_b := 1 - (num_sl_known * ( 1 -
Variables[fI.perc_lsl_known])
                                      / (aNC - num_lsl_base))
      else
        perc_unknown_nonlead_b := 0;
    end
    // p_lsl_unknown = 1
    // all unknown service lines
    else if LSL = 2 then begin
      num_sl_known := 0;
      num_lsl_known := 0;
```

```
      num_sl_unknown := aNC;
      num_lsl_unknown_lead := num_sl_unknown * Variables[fI.perc_lsl_unknown_lead];
      perc_lsl_b := Variables[fI.perc_lsl_unknown_lead];
      Num_LSL_base := num_lsl_known + num_lsl_unknown_lead;
      if CostingData.SystemType = 1 then
        perc_unknown_nonlead_b := 1
      else
        perc_unknown_nonlead_b := 0;
    end
    // p_lsl_nolead_unknown = 1
    // mix of non-lead and unknown
    else if LSL = 3 then begin
      num_sl_unknown := aNC * Variables[fI.perc_lsl_nolead_unknown_unknown];
      num_sl_known := aNC - num_sl_unknown;
      num_lsl_known := 0;
      num_lsl_unknown_lead := num_sl_unknown * Variables[fI.perc_lsl_unknown_lead];
                              perc_lsl_b :=
Variables[fI.perc_lsl_nolead_unknown_unknown] *
        Variables[fI.perc_lsl_unknown_lead];
      Num_LSL_base := num_lsl_known + num_lsl_unknown_lead;
      if CostingData.SystemType = 1 then
        perc_unknown_nonlead_b := 1 - (num_sl_known / (aNC - num_lsl_base))
      else
        perc_unknown_nonlead_b := 0;
    end;

    if option = 'Baseline' then begin
      if Config.BaselineName = 'LCR' then
      begin
        Num_potential_LSL_base := num_lsl_known *
          (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial]);
        Variables[fI.pp_lslr_leadcon] := 0;
        Variables[fI.pp_lslr_galprev_leadcon] := 0;
        Variables[fI.pp_lslr_gal_prev_lsl] := 0;

        Num_potential_LSL_known := num_lsl_known *
          (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial]);

        if Num_potential_LSL_base > 0 then
          Variables[fI.pws_lsl] := 1;

        pp_lslr_lsl_adj := Variables[fI.pp_lslr_lsl] /
          (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial]);
      end
      else if Config.BaselineName = 'LCRR' then
      begin
        Num_potential_LSL_base := Num_LSL_base *
          (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial] +
          Variables[fI.pp_lslr_gal_prev_lsl]);
        Variables[fI.pp_lslr_leadcon] := 0;
```

```
        Variables[fI.pp_lslr_galprev_leadcon] := 0;

        Num_potential_LSL_known := num_lsl_known *
          (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial] +
          Variables[fI.pp_lslr_gal_prev_lsl]);

        if Num_potential_LSL_base > 0 then
          Variables[fI.pws_lsl] := 1;

        pp_lslr_lsl_adj := Variables[fI.pp_lslr_lsl] /
          (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial] +
          Variables[fI.pp_lslr_gal_prev_lsl]);
      end;
    end
    else if option = 'LCRI' then begin
      if Config.LSLOption = 1 then begin
        Num_potential_LSL_base := Num_LSL_base *
          (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial] +
          Variables[fI.pp_lslr_gal_prev_lsl]);
        Variables[fI.pp_lslr_leadcon] := 0;
        Variables[fI.pp_lslr_galprev_leadcon] := 0;

        Num_potential_LSL_known := num_lsl_known *
          (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial] +
          Variables[fI.pp_lslr_gal_prev_lsl]);

        pp_lslr_lsl_adj := Variables[fI.pp_lslr_lsl] /
          (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial] +
          Variables[fI.pp_lslr_gal_prev_lsl]);
      end else if Config.LSLOption = 2 then begin
        Variables[fI.pp_lslr_galprev_leadcon] := 0;
        Num_potential_LSL_base := Num_LSL_base *
          (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial] +
          Variables[fI.pp_lslr_gal_prev_lsl] + Variables[fI.pp_lslr_leadcon]);

        Num_potential_LSL_known := num_lsl_known *
          (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial] +
          Variables[fI.pp_lslr_gal_prev_lsl] + Variables[fI.pp_lslr_leadcon]);

        pp_lslr_lsl_adj := Variables[fI.pp_lslr_lsl] /
          (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial] +
Variables[fI.pp_lslr_leadcon]
          + Variables[fI.pp_lslr_gal_prev_lsl]);
      end else begin
        Num_potential_LSL_base := Num_LSL_base;

        pp_lslr_lsl_adj := Variables[fI.pp_lslr_lsl];

        Num_potential_LSL_known := num_lsl_known;
      end;
```

```
      lsl_replace_pct := Config.LSLRepPct;

      // deferrals
      if lsl_replace_pct * Num_potential_LSL_base > Config.LSLRCap then begin
        lsl_replace_pct_a := Config.LSLRCap / Num_potential_LSL_base;
        deferral_cap := 1;
      end else begin
        lsl_replace_pct_a := lsl_replace_pct;
        deferral_cap := 0;
      end;

      if lsl_replace_pct * Num_potential_LSL_base > (0.039 * (aPop /
Variables[fI.Numb_hh])) then begin
        lsl_replace_pct_b := (0.039 * (aPop / Variables[fI.Numb_hh])) /
Num_potential_LSL_base;
        deferral_pct := 1;
      end else begin
        lsl_replace_pct_b := lsl_replace_pct;
        deferral_pct := 0;
      end;

      if (deferral_cap * deferral_pct) = 1 then begin
        lsl_replace_pct := min(lsl_replace_pct_a, lsl_replace_pct_b);
        deferral_any := 1;
      end else begin
        if deferral_cap = 1 then lsl_replace_pct := lsl_replace_pct_a
        else if deferral_pct = 1 then lsl_replace_pct := lsl_replace_pct_b
        else lsl_replace_pct := lsl_replace_pct;

        deferral_any := 0;
      end;
    end;

    if Num_potential_LSL_base > 0 then Variables[fI.pws_lsl] := 1;
    num_lsl_remain := Num_potential_LSL_base;
    hh_remain_lsl := max(0, (num_hh_per_connect * num_lsl_remain));
  end;

  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('BinChg_' + i.ToString, tmp_double);
    BinChg[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[CostingData.SourceWater,
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 (option = 'Baseline') and (config.BaselineName = 'LCRR') 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;

    pp_above_al_bin_one := Variables[fI.pp90aboveal15_1];
    pp_above_al_bin_two := Variables[fI.pp90aboveal15_2] +
Variables[fI.pp90aboveal15_3];
    pp_above_al_bin_three := Variables[fI.pp90aboveal15_4] +
Variables[fI.pp90aboveal15_5];

    Variables[fI.pp_above_al_bin_one] := pp_above_al_bin_one;
```

```
    Variables[fI.pp_above_al_bin_two] := pp_above_al_bin_two;
    Variables[fI.pp_above_al_bin_three] := pp_above_al_bin_three;
  end;

  if option = 'LCRI' 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.ALE = 15 then begin
      pp_above_al_bin_one := Variables[fI.pp90aboveal15_1];
      pp_above_al_bin_three := Variables[fI.pp90aboveal15_2] +
Variables[fI.pp90aboveal15_3] +
        Variables[fI.pp90aboveal15_4] + Variables[fI.pp90aboveal15_5];
    end
    else if Config.ALE = 12 then begin
      pp_above_al_bin_one := Variables[fI.pp90aboveal12_1] +
Variables[fI.pp90aboveal12_2];
      pp_above_al_bin_three := Variables[fI.pp90aboveal12_3] +
Variables[fI.pp90aboveal12_4] +
        Variables[fI.pp90aboveal12_5];
    end
    else if Config.ALE = 10 then begin
      pp_above_al_bin_one := Variables[fI.pp90aboveal10_1] +
Variables[fI.pp90aboveal10_2] +
        Variables[fI.pp90aboveal10_3];
      pp_above_al_bin_three := Variables[fI.pp90aboveal10_4] +
Variables[fI.pp90aboveal10_5];
    end
    else if Config.ALE = 5 then begin
      pp_above_al_bin_one := Variables[fI.pp90aboveal5_1] +
Variables[fI.pp90aboveal5_2] +
        Variables[fI.pp90aboveal5_3] + Variables[fI.pp90aboveal5_4];
      pp_above_al_bin_three := Variables[fI.pp90aboveal5_5];
    end;

    Variables[fI.pp_above_al_bin_one] := pp_above_al_bin_one;
    Variables[fI.pp_above_al_bin_two] := pp_above_al_bin_two;
    Variables[fI.pp_above_al_bin_three] := pp_above_al_bin_three;
  end;

  CCT_Change := false;
  bCCT_Change := false;
```

```
  if not VLSystem then begin
    num_lsl_replace_y := 0;
    num_unknown_resolved_y := 0;
    if (Num_Proxies = 0) then begin
      SetFH();
      LeadConcentrationBins(pwsid, option, 0, CostingData.SystemSize,
CostingData.SourceWater, pws_lsl,
        CostingData.CCT, POU, aPop, aNC, fAdjust_CCT,
        fInstall_CCT, cct_adjust_yr, cct_install_yr, pou_install_yr, bCCT_Change,
pwswgt, 0, 0, 0,
        perc_lsl_b, Num_Proxies, partial_cct_level, Variables[fI.pp_lslr_lsl],
pp_lslr_lsl_adj,
        num_lsl_filters, num_hh_per_connect, num_temp_pou, True)
    end;
  end;

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

  num_replace := 0;
  num_remain := 0;

  // ************
  // 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 ='LCRI') or ((option = 'Baseline') and (Config.BaselineName =
'LCRR')) 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.pp_grandfather_mand_child] :=
SchoolSampData.pp_grandfather_mand_child;
      Variables[fI.pp_grandfather_opt_child] :=
SchoolSampData.pp_grandfather_opt_child;
      Variables[fI.pp_grandfather_mand_pub_elem] :=
SchoolSampData.pp_grandfather_mand_pub_elem;
      Variables[fI.pp_grandfather_opt_pub_elem] :=
SchoolSampData.pp_grandfather_opt_pub_elem;
      Variables[fI.pp_grandfather_mand_priv_elem] :=
SchoolSampData.pp_grandfather_mand_priv_elem;
      Variables[fI.pp_grandfather_opt_priv_elem] :=
SchoolSampData.pp_grandfather_opt_priv_elem;
      Variables[fI.pp_grandfather_opt1_pub_second] :=
SchoolSampData.pp_grandfather_opt1_pub_second;
      Variables[fI.pp_grandfather_opt2_pub_second] :=
SchoolSampData.pp_grandfather_opt2_pub_second;
      Variables[fI.pp_grandfather_opt1_priv_second] :=
SchoolSampData.pp_grandfather_opt1_priv_second;
      Variables[fI.pp_grandfather_opt2_priv_second] :=
SchoolSampData.pp_grandfather_opt2_priv_second;
    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;

    NewDraw := false;

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

    tpws90pct := 999999;
    ttpws90pct := 999999;

    // Baseline Source and Treatment Change
    if (option = 'Baseline') and not VLSystem then begin
      if Config.BaselineName = 'LCR' 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
              tBin := BinChg[y];

              if tBin = 0 then raise Exception.Create('LCR already treated tbin =
0');

              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
              tBin := BinChg[y];

              if tBin = 0 then raise Exception.Create('LCR not treated tbin = 0');

              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 begin
              cct_install_yr := y + 5;
              cct_change_yr := cct_install_yr;
            end;

            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 begin
              cct_install_yr := y + 4;
              cct_change_yr := cct_install_yr;
            end;
            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;

            proxy3_pws90[y + 6] := Variables[fI.post_cct_p90_bin1];
            CCT_Change := True;
            if cct_adjust_yr = 0 then begin
              cct_adjust_yr := y + 4;
              cct_change_yr := cct_adjust_yr;
            end;

            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;

            proxy3_pws90[y + 5] := Variables[fI.post_cct_p90_bin1];
            CCT_Change := True;
```

```
            if cct_adjust_yr = 0 then begin
              cct_adjust_yr := y + 3;
              cct_change_yr := cct_adjust_yr;
            end;

            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
      else if Config.BaselineName = 'LCRR' 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 >= 1 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_dssa] *
                Variables[fI.pp_above_al_bin_three] * Variables[fI.numb_reduced_tap]
            else if (Variables[fI.p_tap_triennial] = 1) and
              ((y = 1) or (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_dssa] *
                Variables[fI.pp_above_al_bin_three] * Variables[fI.numb_reduced_tap]
            else if (Variables[fI.p_tap_nine] = 1) and ((y = 1) or (y = 10) or (y =
19) or (y = 28))
            then
              numb_wqp_add_sites[y] := Variables[fI.pp_overlap_dssa] *
                Variables[fI.pp_above_al_bin_three] * Variables[fI.numb_reduced_tap]
            else
              numb_wqp_add_sites[y] := Variables[fI.pp_overlap_dssa] *
```

```
                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_dssa] *
              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_dssa] *
              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]);

          // 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
          // install or modify or pou
          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
                tBin := BinChg[y];

              if tBin = 0 then raise Exception.Create('LCRR already treated tBin =
0');

              if tBin = 1 then
                tpws90pct := 20
              else if (tBin = 2) or (tBin = 3) then
                tpws90pct := 12
              else if (tBin = 4) or (tBin = 5) then
                tpws90pct := 5;

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

```
              SourceTreatChangeEver := True;
            end;
          // no install, modify or pou
          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
              tBin := BinChg[y];

              if tBin = 0 then raise Exception.Create('LCRR not treated tBin = 0');

              if tBin = 1 then
                tpws90pct := 20
              else if (tBin = 2) or (tBin = 3) then
                tpws90pct := 12
              else if (tBin = 4) or (tBin = 5) then
                tpws90pct := 5;

              proxy2_pws90 := tpws90pct;

              SourceTreatChangeEver := True;
            end;
          end;

          // CCT, LSLR, and POU Plans/Studies
          if (SystemType = 1) and (aPop > 10000) then begin
            if LSL > 0 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 > 0) 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;

          // CCT and POU not VLS
          if (SystemType = 1) and (aPop > 10000) then begin
            // large CWS systems
            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 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
            // Small systems and NTNCWS
            if (proxy2_pws90 > bp1) and (proxy2_pws90 < bp2) and (CCT = 1) and (not
CCT_Change) and
              (POU = 0) then begin
              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 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;

              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; // y >= 1
      end; // if Config.BaselineName = 'LCRR'
    end
    else if (option = 'LCRI') 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_dssa] *
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_dssa] *
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_dssa] *
pp_above_al_bin_three *
              Variables[fI.numb_reduced_tap]
          else
            numb_wqp_add_sites[y] := Variables[fI.pp_overlap_dssa] *
pp_above_al_bin_three *
              (2 * Variables[fI.numb_samp_customer]);
        end
        else if owBin = 1 then
          numb_wqp_add_sites[y] := Variables[fI.pp_overlap_dssa] *
            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]);

        // 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
            MakeLCRxBin(option, Config.ALE, BinChg[y], tBin, temp_int);

            if tBin = 1 then begin
              if Config.ALE = 15 then
                tpws90pct := 20
              else if Config.ALE = 12 then
                tpws90pct := 15
```

```
            else if Config.ALE = 10 then
              tpws90pct := 12
            else if Config.ALE = 5 then
              tpws90pct := 8;
          end else begin
            if Config.ALE = 15 then
              tpws90pct := 10
            else if Config.ALE = 12 then
              tpws90pct := 8
            else if Config.ALE = 10 then
              tpws90pct := 5
            else if Config.ALE = 5 then
              tpws90pct := 2;
          end;

          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
          MakeLCRxBin(option, Config.ALE, BinChg[y], tBin, temp_int);

          if tBin = 1 then begin
            if Config.ALE = 15 then
              tpws90pct := 20
            else if Config.ALE = 12 then
              tpws90pct := 15
            else if Config.ALE = 10 then
              tpws90pct := 12
            else if Config.ALE = 5 then
              tpws90pct := 8;
            ToBin1Count := 1;
          end else begin
            if Config.ALE = 15 then
              tpws90pct := 10
            else if Config.ALE = 12 then
              tpws90pct := 8
            else if Config.ALE = 10 then
              tpws90pct := 5
            else if Config.ALE = 5 then
              tpws90pct := 2;
          end;
```

```
            proxy2_pws90 := tpws90pct;

            SourceTreatChangeEver := True;
          end;
        end;

        if y = 4 then
        begin
          proxy2_pws90_y4 := proxy2_pws90;

          if (LSL > 0) and (Num_potential_LSL_base < 80) and (proxy2_pws90 > bp2)
then
          begin
            lslr_newpath := 1;
            lsl_replace_pct := 0.2;
            pubNewPath := lslr_newpath;
          end;
        end;

        // *********************
        // CCT, LSLR, and POU Plans/Studies
        // *********************

        if LSL > 0 then
          Variables[fI.b_lslr_study] := 1;

        // CCT and POU

        // Modify CCT
        if ((SystemType = 1) and (aPop > Config.SmallProxyPop) and (proxy2_pws90 >
bp2) and
          (CCT = 1) and (not CCT_Change) and (POU = 0) and
          ((lslr_newpath <> 1) or ((lslr_newpath = 1) and (LSL = 0)))) or

          ((SystemType = 2) and (proxy2_pws90 > bp2) and (CCT = 1) and (not
CCT_Change) and
          (POU = 0) and ((lslr_newpath <> 1) or ((lslr_newpath = 1) and (LSL = 0))))
or

          ((SystemType = 1) and (aPop <= Config.SmallProxyPop) and (Small_Correct =
2) and
          (proxy2_pws90 > bp2) and (CCT = 1) and (not CCT_Change) and (POU = 0) and
          ((lslr_newpath <> 1) or ((lslr_newpath = 1) and (LSL = 0)))) 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;
```

```
          if (Config.ALE = 5) and (b_ale_not_achieved = 1) then
            proxy3_pws90[y + 6] := 6
          else
            proxy3_pws90[y + 6] := proxy2_pws90 * 0.2;

          CCT_Change := True;
          if cct_adjust_yr = 0 then
          begin
            cct_adjust_yr := y + 4;
            cct_change_yr := cct_adjust_yr;
          end;

          b_cct_study_mod_al[y + 3] := 1;
        end
        // Install CCT
        else if ((SystemType = 1) and (aPop > Config.SmallProxyPop) and
(proxy2_pws90 > bp2) and
          (CCT = 0) and (not CCT_Change) and (POU = 0) and
          ((lslr_newpath <> 1) or ((lslr_newpath = 1) and (LSL = 0)))) or

          ((SystemType = 2) and (proxy2_pws90 > bp2) and (CCT = 0) and (not
CCT_Change) and
          (POU = 0) and ((lslr_newpath <> 1) or ((lslr_newpath = 1) and (LSL = 0))))
or

          ((SystemType = 1) and (aPop <= Config.SmallProxyPop) and (Small_Correct =
2) and
          (proxy2_pws90 > bp2) and (CCT = 0) and (not CCT_Change) and (POU = 0) and
          ((lslr_newpath <> 1) or ((lslr_newpath = 1) and (LSL = 0)))) then begin

          for i := y + 4 to Config.YearsOfAnalysis do begin
            InstallCCT[i] := True;
            b_install_cct[i] := 1;
          end;

          if cct_install_yr = 0 then
          begin
            cct_install_yr := y + 4;
            cct_change_yr := cct_install_yr;
          end;

          b_cct_study_install[y + 3] := 1;

          if (Config.ALE = 5) and (b_ale_not_achieved = 1) then
            proxy3_pws90[y + 6] := 6
          else
            proxy3_pws90[y + 6] := proxy2_pws90 * 0.2;

          CCT_Change := True;
```

```
          // 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; // y >= 4
    end; // end option = 'LCRI'

    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;

    proxy4_pws90 := -1;
```

```
    if (LSL > 0) and not VLSystem then begin
      if option = 'Baseline' then begin
        if Config.BaselineName = 'LCR' then
        begin
          if y >= 4 then begin
            Num_lsl_replace_state[y] := min(state_lslr_rate *
Num_potential_LSL_known, num_lsl_remain);

            if proxy3_pws90[y] > bp2 then
              Num_lsl_replace_rule_all[y] := min(0.07 * Num_potential_LSL_known,
num_lsl_remain);

            Num_lsl_replace_rule[y] := Num_lsl_replace_rule_all[y] * (1 -
Variables[fI.pp_lcr_test]);
            num_lsl_testout[y] := Num_lsl_replace_rule_all[y] *
Variables[fI.pp_lcr_test];

            Num_lsl_replace_tot[y] := max(Num_lsl_replace_rule[y],
Num_lsl_replace_state[y]);
            Num_lsl_replace_fed[y] := Num_lsl_replace_tot[y] -
Num_lsl_replace_state[y];

            // NOTE in Baseline unknowns are not investigated NEEDS TO GO TO
WORKBOOK
            num_unknown_resolved[y] := min(0.1 * num_sl_unknown,
num_unknown_remain);

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

            if Num_lsl_replace_rule_allb >= Num_potential_LSL_known * 0.21 then
begin
              proxy4_pws90 := bp1 + ((bp2 - bp1) / 2);
              //proxy4_pws90 := 10;
            end;

            // NOTE After Num_replace >= Num_potential_LSL_known * 0.21 do not do
any rule replacement.

            tot_num_lslr_lsl_replace[y] := Num_lsl_replace_tot[y] *
              (Variables[fI.pp_lslr_lsl] / (Variables[fI.pp_lslr_lsl] +
Variables[fI.pp_lslr_partial]));

            tot_num_lslr_partial_replace[y] := Num_lsl_replace_tot[y] *
              (Variables[fI.pp_lslr_partial] / (Variables[fI.pp_lslr_lsl] +
Variables[fI.pp_lslr_partial]));

            fed_num_lslr_lsl_replace[y] := Num_lsl_replace_fed[y] *
              (Variables[fI.pp_lslr_lsl] / (Variables[fI.pp_lslr_lsl] +
Variables[fI.pp_lslr_partial]));
```

```
            fed_num_lslr_partial_replace[y] := Num_lsl_replace_fed[y] *
              (Variables[fI.pp_lslr_partial] / (Variables[fI.pp_lslr_lsl] +
Variables[fI.pp_lslr_partial]));

            tot_num_lslr_gal_prev_lsl_replace[y] := 0;
            tot_num_lslr_leadcon_replace[y] := 0;
            tot_num_lslr_galprev_leadcon_replace[y] := 0;

            fed_num_lslr_gal_prev_lsl_replace[y] := 0;
            fed_num_lslr_leadcon_replace[y] := 0;
            fed_num_lslr_galprev_leadcon_replace[y] := 0;

            if Num_lsl_replace_fed[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.pws_lsl] := LSL;
              NewDraw := True;
              num_lsl_remain := 0;

              proxy4_pws90 := 5;
            end;

            num_replace := 0;
            Num_lsl_replace_rule_allb := 0;
            for i := 1 to y do begin
              num_replace := num_replace + Num_lsl_replace_tot[i];
              Num_lsl_replace_rule_allb := Num_lsl_replace_rule_allb +
Num_lsl_replace_rule_all[i];
            end;

            SumTotLSLReplaceMand := SumTotLSLReplaceMand +
tot_num_lslr_lsl_replace[y];
            SumTotLSLPartialReplaceMand := SumTotLSLPartialReplaceMand +
tot_num_lslr_partial_replace[y];
            SumTotLSLGalPrevMand := SumTotLSLGalPrevMand +
tot_num_lslr_gal_prev_lsl_replace[y];
            SumTotLSLLeadConMand := SumTotLSLLeadConMand +
tot_num_lslr_leadcon_replace[y];
            SumTotLSLGalPrevLeadConMand := SumTotLSLGalPrevLeadConMand +
tot_num_lslr_galprev_leadcon_replace[y];

            SumTotLSLReplaceVol := 0;
            SumTotLSLPartialReplaceVol := 0;
            SumTotLSLGalPrevVol := 0;
            SumTotLSLLeadConVol := 0;
```

```
            SumTotLSLGalPrevLeadConVol := 0;

            SumFedLSLReplace := SumFedLSLReplace + Num_lsl_replace_fed[y];
            SumFedLSLReplaceMand := SumFedLSLReplaceMand +
fed_num_lslr_lsl_replace[y];
            SumFedLSLPartialReplaceMand := SumFedLSLPartialReplaceMand +
fed_num_lslr_partial_replace[y];
            SumFedLSLGalPrevMand := SumFedLSLGalPrevMand +
fed_num_lslr_gal_prev_lsl_replace[y];
            SumFedLSLLeadConMand := SumFedLSLLeadConMand +
fed_num_lslr_leadcon_replace[y];
            SumFedLSLGalPrevLeadConMand := SumFedLSLGalPrevLeadConMand +
fed_num_lslr_galprev_leadcon_replace[y];

            SumFedLSLReplaceVol := 0;
            SumFedLSLPartialReplaceVol := 0;
            SumFedLSLGalPrevVol := 0;
            SumFedLSLLeadConVol := 0;
            SumFedLSLGalPrevLeadConVol := 0;

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

            num_unknown_remain := max(0, num_sl_unknown - num_unknown_resolved[y]);

            Variables[fI.num_lsl_replace] := Num_lsl_replace_tot[y];
            Variables[fI.num_lsl_testout] := num_lsl_testout[y];
            Variables[fI.hh_remain_lsl] := hh_remain_lsl;
            Variables[fI.num_lslr_lsl_replace] := fed_num_lslr_lsl_replace[y];
            Variables[fI.num_lslr_partial_replace] :=
fed_num_lslr_partial_replace[y];
          end;
        end // Config.BaselineName = 'LCR'
        else if Config.BaselineName = 'LCRR' then begin
          pp_lsl_replaced_vol_pct_yr[y] := max(pp_lsl_replaced_vol_pct_yr[y],
state_lslr_rate);
          pp_lsl_replaced_mand_pct := max(0.03, state_lslr_rate);
          pp_lsl_replaced_mand_pct_small := max(0.07, state_lslr_rate);

          num_unknown_resolved[y] := min(0.10 * num_sl_unknown, num_unknown_remain);
          num_unknown_remain := max(0, num_sl_unknown - num_unknown_resolved[y]);
          hh_unknown_remain := max(0, num_sl_unknown - num_unknown_resolved[y]) *
num_hh_per_connect;
          Variables[fI.num_unknown_resolved] := num_unknown_resolved[y];

          if num_lsl_remain > 0 then begin
            if (SystemType = 1) and (aPop > 10000) then begin
              if (proxy3_pws90[y] > bp1) and (proxy3_pws90[y] <= bp2) then begin
                Num_lsl_replace_tot[y] := min(pp_lsl_replaced_vol_pct_yr[y] *
Num_potential_LSL_base,
```

```
                num_lsl_remain);
              Num_lsl_replace_state[y] := min(state_lslr_rate *
Num_potential_LSL_base, num_lsl_remain);
              Num_lsl_replace_fed[y] := Num_lsl_replace_tot[y] -
Num_lsl_replace_state[y];

              if Num_lsl_replace_tot[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_tot[y] := min(pp_lsl_replaced_mand_pct *
(Num_potential_LSL_base),
                num_lsl_remain);
              Num_lsl_replace_state[y] := min(state_lslr_rate *
Num_potential_LSL_base,
                num_lsl_remain);
              Num_lsl_replace_fed[y] := Num_lsl_replace_tot[y] -
Num_lsl_replace_state[y];

              // NOTE in mandatory program unknowns are investigated at same rate
as the federal required rate lsls are replaced
              // num_unknown_resolved[y] must go to work book

              if Num_lsl_replace_tot[y] > 0 then
                Variables[fI.b_lslr_mand] := 1
             else
                Variables[fI.b_lslr_mand] := 0;
            end else begin
              Num_lsl_replace_tot[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_tot[y] := min(pp_lsl_replaced_mand_pct_small *
Num_potential_LSL_base,
                num_lsl_remain);
              Num_lsl_replace_state[y] := min(state_lslr_rate *
Num_potential_LSL_base, num_lsl_remain);
              Num_lsl_replace_fed[y] := Num_lsl_replace_tot[y] -
Num_lsl_replace_state[y];

              // NOTE in mandatory small business program unknowns are
investigated at same rate as lsls are replaced

              num_unknown_resolved[y] := min(pp_lsl_replaced_mand_pct_small *
```

```
num_sl_unknown, num_unknown_remain);

                if Num_lsl_replace_tot[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_tot[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;
            proxy4_pws90 := proxy3_pws90[y];
          end else begin
            LSL := 0;
            Variables[fI.pws_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_tot[i];
            num_requested := num_requested + num_lsl_requested[i];
          end;

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

          num_unknown_remain := max(0, num_sl_unknown - num_unknown_resolved[y]);

          Variables[fI.num_lsl_replace] := Num_lsl_replace_tot[y];
          Variables[fI.hh_remain_lsl] := hh_remain_lsl;
```

```
          fed_num_lslr_lsl_replace[y] := Num_lsl_replace_fed[y] *
            (Variables[fI.pp_lslr_lsl] / (Variables[fI.pp_lslr_lsl] +
Variables[fI.pp_lslr_partial] +
            Variables[fI.pp_lslr_gal_prev_lsl]));

          fed_num_lslr_partial_replace[y] := Num_lsl_replace_fed[y] *
            (Variables[fI.pp_lslr_partial] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_gal_prev_lsl]));

          fed_num_lslr_gal_prev_lsl_replace[y] := Num_lsl_replace_fed[y] *
            (Variables[fI.pp_lslr_gal_prev_lsl] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_gal_prev_lsl]));

          fed_num_lslr_leadcon_replace[y] := 0;
          fed_num_lslr_galprev_leadcon_replace[y] := 0;

          tot_num_lslr_lsl_replace[y] := Num_lsl_replace_tot[y] *
            (Variables[fI.pp_lslr_lsl] / (Variables[fI.pp_lslr_lsl] +
Variables[fI.pp_lslr_partial] +
            Variables[fI.pp_lslr_gal_prev_lsl]));

          tot_num_lslr_partial_replace[y] := Num_lsl_replace_tot[y] *
            (Variables[fI.pp_lslr_partial] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_gal_prev_lsl]));

          tot_num_lslr_gal_prev_lsl_replace[y] := Num_lsl_replace_tot[y] *
            (Variables[fI.pp_lslr_gal_prev_lsl] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_gal_prev_lsl]));

          tot_num_lslr_leadcon_replace[y] := 0;
          tot_num_lslr_galprev_leadcon_replace[y] := 0;

          if proxy3_pws90[y] > bp2 then begin
            SumFedLSLReplaceMand := SumFedLSLReplaceMand +
fed_num_lslr_lsl_replace[y];
            SumFedLSLPartialReplaceMand := SumFedLSLPartialReplaceMand +
fed_num_lslr_partial_replace[y];
            SumFedLSLGalPrevMand := SumFedLSLGalPrevMand +
fed_num_lslr_gal_prev_lsl_replace[y];
            SumFedLSLLeadConMand := SumFedLSLLeadConMand +
fed_num_lslr_leadcon_replace[y];
            SumFedLSLGalPrevLeadConMand := SumFedLSLGalPrevLeadConMand +
              fed_num_lslr_galprev_leadcon_replace[y];

            SumTotLSLReplaceMand := SumTotLSLReplaceMand +
tot_num_lslr_lsl_replace[y];
            SumTotLSLPartialReplaceMand := SumTotLSLPartialReplaceMand +
tot_num_lslr_partial_replace[y];
            SumTotLSLGalPrevMand := SumTotLSLGalPrevMand +
tot_num_lslr_gal_prev_lsl_replace[y];
```

```
            SumTotLSLLeadConMand := SumTotLSLLeadConMand +
tot_num_lslr_leadcon_replace[y];
            SumTotLSLGalPrevLeadConMand := SumTotLSLGalPrevLeadConMand +
tot_num_lslr_galprev_leadcon_replace[y];
          end
          else if (proxy3_pws90[y] > bp1) and (proxy3_pws90[y] <= bp2) then begin
            SumFedLSLReplaceVol := SumFedLSLReplaceVol +
fed_num_lslr_lsl_replace[y];
            SumFedLSLPartialReplaceVol := SumFedLSLPartialReplaceVol +
fed_num_lslr_partial_replace[y];
            SumFedLSLGalPrevVol := SumFedLSLGalPrevVol +
fed_num_lslr_gal_prev_lsl_replace[y];
            SumFedLSLLeadConVol := SumFedLSLLeadConVol +
fed_num_lslr_leadcon_replace[y];
            SumFedLSLGalPrevLeadConVol := SumFedLSLGalPrevLeadConVol +
fed_num_lslr_galprev_leadcon_replace[y];

            SumTotLSLReplaceVol := SumTotLSLReplaceVol +
tot_num_lslr_lsl_replace[y];
            SumTotLSLPartialReplaceVol := SumTotLSLPartialReplaceVol +
tot_num_lslr_partial_replace[y];
            SumTotLSLGalPrevVol := SumTotLSLGalPrevVol +
tot_num_lslr_gal_prev_lsl_replace[y];
            SumTotLSLLeadConVol := SumTotLSLLeadConVol +
tot_num_lslr_leadcon_replace[y];
            SumTotLSLGalPrevLeadConVol := SumTotLSLGalPrevLeadConVol +
tot_num_lslr_galprev_leadcon_replace[y];
          end;

          Variables[fI.num_lslr_lsl_replace] := fed_num_lslr_lsl_replace[y];
          Variables[fI.num_lslr_partial_replace] := fed_num_lslr_partial_replace[y];
          Variables[fI.num_lslr_gal_prev_lsl_replace] :=
fed_num_lslr_gal_prev_lsl_replace[y];
          Variables[fI.num_lslr_leadcon_replace] := fed_num_lslr_leadcon_replace[y];
          Variables[fI.num_lslr_galprev_leadcon_replace] :=
fed_num_lslr_galprev_leadcon_replace[y];

          // Failure to meet LSLR Voluntary Program in Bin 2
          if (SystemType = 1) and (aPop > 10000) 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_tot[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
            else
            begin
              if NM >= 1 then
                Variables[fI.fail_nm1] := 1;
              if NM >= 2 then
                Variables[fI.fail_nm2] := 1;
            end;
          end else begin
            NM := 0;
            Variables[fI.Meet_Lslr_Goal] := 1;
            Variables[fI.fail_nm1] := 0;
            Variables[fI.fail_nm2] := 0;
          end;
        end; // end Config.BaselineName = 'LCRR'
      end
      else if option = 'LCRI' then begin
        // NOTE in LCRI unknowns are investigated at same rate as lsls are replaced

        num_unknown_resolved[y] := min(0.10 * num_sl_unknown, num_unknown_remain);
        num_unknown_remain := max(0, num_sl_unknown - num_unknown_resolved[y]);
        hh_unknown_remain := max(0, num_sl_unknown - num_unknown_resolved[y]) *
num_hh_per_connect;

        // Determines the number of lines that must be validated in year 8.
        // Number in validation pool = unknowns resolved that are not lead.
        // Only occurs one year (year 8)
        if y = 8 then begin
          Num_lsl_validate_pool[y] := (num_sl_unknown - num_unknown_remain) *
            (1 - Variables[fI.perc_lsl_unknown_lead]);
          if Num_lsl_validate_pool[y] < 1500 then
            num_lsl_validated[y] := 0.20 * Num_lsl_validate_pool[y]
          else
            num_lsl_validated[y] := (Num_lsl_validate_pool[y] * 384.16) /
              (Num_lsl_validate_pool[y] + 383.16);
        end;

        Variables[fI.num_unknown_remain] := num_unknown_remain;
        Variables[fI.hh_unknown_remain] := hh_unknown_remain;
```

```
        Variables[fI.num_lsl_validated] := num_lsl_validated[y];
        Variables[fI.num_unknown_resolved] := num_unknown_resolved[y];

        if y >= 4 then begin
          if num_lsl_remain > 0 then begin
            Num_lsl_replace_tot[y] := min(lsl_replace_pct * Num_potential_LSL_base,
num_lsl_remain);
            Num_lsl_replace_state[y] := min(state_lslr_rate *
Num_potential_LSL_base,
              num_lsl_remain);
            Num_lsl_replace_fed[y] := Num_lsl_replace_tot[y] -
Num_lsl_replace_state[y];

            num_replace := num_replace + Num_lsl_replace_tot[y];
            num_lsl_remain := Num_potential_LSL_base - num_replace;
            hh_remain_lsl := max(0, (num_hh_per_connect * num_lsl_remain));

            if Num_lsl_replace_tot[y] > 0 then
              Variables[fI.b_lslr_mand] := 1
            else
              Variables[fI.b_lslr_mand] := 0;

            proxy4_pws90 := proxy3_pws90[y];
          end else begin
            LSL := 0;
            Variables[fI.pws_lsl] := LSL;
            NewDraw := True;
            num_lsl_remain := 0;

            // lower than any ALE
            ttpws90pct := bp2 / 2;

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

          Variables[fI.num_lsl_replace] := Num_lsl_replace_tot[y];
          Variables[fI.hh_remain_lsl] := hh_remain_lsl;

          if Config.LSLOption = 1 then begin
            Variables[fI.pp_lslr_leadcon] := 0;
            Variables[fI.pp_lslr_galprev_leadcon] := 0;
          end
          else if Config.LSLOption = 2 then
            Variables[fI.pp_lslr_galprev_leadcon] := 0;

          tot_num_lslr_lsl_replace[y] := Num_lsl_replace_tot[y] *
            (Variables[fI.pp_lslr_lsl] / (Variables[fI.pp_lslr_lsl] +
```

```
Variables[fI.pp_lslr_partial]
            + Variables[fI.pp_lslr_leadcon] + Variables[fI.pp_lslr_gal_prev_lsl] +
            Variables[fI.pp_lslr_galprev_leadcon]));

          tot_num_lslr_partial_replace[y] := Num_lsl_replace_tot[y] *
            (Variables[fI.pp_lslr_partial] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_leadcon] +
            Variables[fI.pp_lslr_gal_prev_lsl] +
Variables[fI.pp_lslr_galprev_leadcon]));

          tot_num_lslr_gal_prev_lsl_replace[y] := Num_lsl_replace_tot[y] *
            (Variables[fI.pp_lslr_gal_prev_lsl] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_leadcon] +
            Variables[fI.pp_lslr_gal_prev_lsl] +
Variables[fI.pp_lslr_galprev_leadcon]));

          tot_num_lslr_leadcon_replace[y] := Num_lsl_replace_tot[y] *
            (Variables[fI.pp_lslr_leadcon] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_leadcon] +
            Variables[fI.pp_lslr_gal_prev_lsl] +
Variables[fI.pp_lslr_galprev_leadcon]));

          tot_num_lslr_galprev_leadcon_replace[y] := Num_lsl_replace_tot[y] *
            (Variables[fI.pp_lslr_galprev_leadcon] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_leadcon] +
            Variables[fI.pp_lslr_gal_prev_lsl] +
Variables[fI.pp_lslr_galprev_leadcon]));

          fed_num_lslr_lsl_replace[y] := Num_lsl_replace_fed[y] *
            (Variables[fI.pp_lslr_lsl] / (Variables[fI.pp_lslr_lsl] +
Variables[fI.pp_lslr_partial]
            + Variables[fI.pp_lslr_leadcon] + Variables[fI.pp_lslr_gal_prev_lsl] +
            Variables[fI.pp_lslr_galprev_leadcon]));

          fed_num_lslr_partial_replace[y] := Num_lsl_replace_fed[y] *
            (Variables[fI.pp_lslr_partial] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_leadcon] +
            Variables[fI.pp_lslr_gal_prev_lsl] +
Variables[fI.pp_lslr_galprev_leadcon]));

          fed_num_lslr_gal_prev_lsl_replace[y] := Num_lsl_replace_fed[y] *
            (Variables[fI.pp_lslr_gal_prev_lsl] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_leadcon] +
            Variables[fI.pp_lslr_gal_prev_lsl] +
Variables[fI.pp_lslr_galprev_leadcon]));

          fed_num_lslr_leadcon_replace[y] := Num_lsl_replace_fed[y] *
            (Variables[fI.pp_lslr_leadcon] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_leadcon] +
            Variables[fI.pp_lslr_gal_prev_lsl] +
```

```
Variables[fI.pp_lslr_galprev_leadcon]));

          fed_num_lslr_galprev_leadcon_replace[y] := Num_lsl_replace_fed[y] *
            (Variables[fI.pp_lslr_galprev_leadcon] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_leadcon] +
            Variables[fI.pp_lslr_gal_prev_lsl] +
Variables[fI.pp_lslr_galprev_leadcon]));

          SumTotLSLReplaceMand := SumTotLSLReplaceMand +
tot_num_lslr_lsl_replace[y];
          SumTotLSLPartialReplaceMand := SumTotLSLPartialReplaceMand +
            tot_num_lslr_partial_replace[y];
          SumTotLSLGalPrevMand := SumTotLSLGalPrevMand +
tot_num_lslr_gal_prev_lsl_replace[y];
          SumTotLSLLeadConMand := SumTotLSLLeadConMand +
tot_num_lslr_leadcon_replace[y];
          SumTotLSLGalPrevLeadConMand := SumTotLSLGalPrevLeadConMand +
            tot_num_lslr_galprev_leadcon_replace[y];

          SumFedLSLReplaceMand := SumFedLSLReplaceMand +
fed_num_lslr_lsl_replace[y];
          SumFedLSLPartialReplaceMand := SumFedLSLPartialReplaceMand +
            fed_num_lslr_partial_replace[y];
          SumFedLSLGalPrevMand := SumFedLSLGalPrevMand +
fed_num_lslr_gal_prev_lsl_replace[y];
          SumFedLSLLeadConMand := SumFedLSLLeadConMand +
fed_num_lslr_leadcon_replace[y];
          SumFedLSLGalPrevLeadConMand := SumFedLSLGalPrevLeadConMand +
            fed_num_lslr_galprev_leadcon_replace[y];

          SumTotLSLReplaceVol := 0;
          SumTotLSLPartialReplaceVol := 0;
          SumTotLSLGalPrevVol := 0;
          SumTotLSLLeadConVol := 0;
          SumTotLSLGalPrevLeadConVol := 0;

          SumFedLSLReplaceVol := 0;
          SumFedLSLPartialReplaceVol := 0;
          SumFedLSLGalPrevVol := 0;
          SumFedLSLLeadConVol := 0;
          SumFedLSLGalPrevLeadConVol := 0;

          Variables[fI.num_lslr_lsl_replace] := fed_num_lslr_lsl_replace[y];
          Variables[fI.num_lslr_partial_replace] := fed_num_lslr_partial_replace[y];
          Variables[fI.num_lslr_gal_prev_lsl_replace] :=
fed_num_lslr_gal_prev_lsl_replace[y];
          Variables[fI.num_lslr_leadcon_replace] := fed_num_lslr_leadcon_replace[y];
          Variables[fI.num_lslr_galprev_leadcon_replace] :=
fed_num_lslr_galprev_leadcon_replace[y];
```

```
          // calculate number of households that get filters if Filters for
LSL/Unknowns is turned on
          // in interface (lsl_filter = 1)

          if Config.LSLFilters = 1 then
          begin
            if SystemType = 1 then num_lsl_filters := hh_remain_lsl +
hh_unknown_remain
            else num_lsl_filters := (num_lsl_remain + num_unknown_remain) *
Variables[fI.numb_ntncws_filters];
          end
          else
            num_lsl_filters := 0;

          // A. Calculate the number of households requiring temporary POU in year
and correct for
          // double counting between Filters for LSL/Unknowns and temp POU
          num_temp_pou := 0;

          if Config.ALE = 15 then
            P90_concurrent := Variables[fI.p_90_concurrent_15]
          else if Config.ALE = 12 then
            P90_concurrent := Variables[fI.p_90_concurrent_12]
          else if Config.ALE = 10 then
            P90_concurrent := Variables[fI.p_90_concurrent_10]
          else if Config.ALE = 5 then
            P90_concurrent := Variables[fI.p_90_concurrent_5];

          if Config.TempPOUOption = 1 then begin
            Variables[fI.b_temp_pou_2] := 0;
            Variables[fI.b_temp_pou_3] := 0;
            Variables[fI.b_temp_pou_4] := 0;

            if (proxy2_pws90_y4 > bp2) and (cct_change_yr > y) and (POU = 0) and
(P90_concurrent = 1)
            then begin
              Variables[fI.b_temp_pou_1] := 1;
              if SystemType = 1 then
                num_temp_pou := aNC - num_lsl_filters
              else
                num_temp_pou := (aNC * Variables[fI.numb_ntncws_filters]) -
num_lsl_filters;

            end
            else
            begin
              Variables[fI.b_temp_pou_1] := 0;
            end;
          end
          else if Config.TempPOUOption = 2 then begin
```

```
            Variables[fI.b_temp_pou_1] := 0;
            Variables[fI.b_temp_pou_3] := 0;
            Variables[fI.b_temp_pou_4] := 0;

            if (proxy2_pws90_y4 > bp2) and (cct_change_yr > y) and (POU = 0) and
(P90_concurrent = 1)
            then begin
              Variables[fI.b_temp_pou_2] := 1;

              if SystemType = 1 then num_temp_pou := hh_remain_lsl +
hh_unknown_remain - num_lsl_filters
              else num_temp_pou := ((num_lsl_remain + num_unknown_remain)
*Variables[fI.numb_ntncws_filters])- num_lsl_filters;
            end
            else
            begin
              Variables[fI.b_temp_pou_2] := 0;
            end;
          end
          else if Config.TempPOUOption = 3 then begin
            Variables[fI.b_temp_pou_1] := 0;
            Variables[fI.b_temp_pou_2] := 0;
            Variables[fI.b_temp_pou_4] := 0;

            if (proxy2_pws90_y4 > bp2) and (cct_change_yr > y) and (POU = 0) and
(P90_concurrent = 1)
            then begin
              Variables[fI.b_temp_pou_3] := 1;

              if SystemType = 1 then num_temp_pou := (hh_remain_lsl +
hh_unknown_remain - num_lsl_filters) * Variables[fI.pp_pou_adopt]
              else num_temp_pou := ((num_lsl_remain + num_unknown_remain)
*Variables[fI.numb_ntncws_filters])- num_lsl_filters;
            end
            else
            begin
              Variables[fI.b_temp_pou_3] := 0;
            end;
          end
          else if Config.TempPOUOption = 4 then begin
            Variables[fI.b_temp_pou_1] := 0;
            Variables[fI.b_temp_pou_2] := 0;
            Variables[fI.b_temp_pou_3] := 0;
            Variables[fI.b_temp_pou_4] := 1;
          end;

          Variables[fI.num_temp_pou] := num_temp_pou;
          Variables[fI.num_lsl_filter] := num_lsl_filters;
        end;  // y = 4
      end; // end option = 'LCRI'
```

```
    end; // end has LSL

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

    // NOTELCRI check on num_lsl_requested
    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;

    // find and fix
    if ((option = 'Baseline') and (Config.BaselineName = 'LCRR')) and
       (y >= 1) 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 = 1) or (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 = 1) or (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_dssa] := 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
    else if (option = 'LCRI') and (y >= 4) and not VLSystem then begin
      Num_tap_ge_al := 0;

      // bin = 3
      if pws90pct <= bp2 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],
            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],
            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],
            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,
            pp_above_al_bin_three);
      end
      else
```

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

      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_dssa] := 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];
      end;
    end;

    if (option = 'Baseline') and (Config.BaselineName = 'LCRR') then begin
      if y >= 1 then begin
        if ff_pws90pct > 0 then
          pws90pct := min(proxy4_pws90, ff_pws90pct)
        else
          pws90pct := proxy4_pws90;
      end;
    end
    else if option = 'LCRI' 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_dssa_cost] := 0;
    Variables[fI.cct_dssa_cost_umra] := 0;
    Variables[fI.cct_dssa_cost_umra_om] := 0;
    Variables[fI.cct_dssa_cost_p] := 0;

    FindAndFixCostCap2 := 0;

    if fnf and (hff_cct = 2) then begin
      Variables[fI.cct_dssa_cost] := (Variables[fI.hrs_act_wqp_op] *
Variables[fI.rate_op]) +
        Variables[fI.cost_act_wqp];
```

```
      Variables[fI.cct_dssa_cost_umra] := (Variables[fI.hrs_act_wqp_op] *
Variables[fI.rate_op])
        + Variables[fI.cost_act_wqp];
      Variables[fI.cct_dssa_cost_umra_om] := 0;
      Variables[fI.cct_dssa_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_dssa_cost] := (FindAndFixCostOM + FindAndFixCostCapDisc -
AdjustCCTCostOM)
        * (1 / aEP);
      if (y - hffY2) MOD UsefulLifeFF = 0 then begin
        Variables[fI.cct_dssa_cost_umra] := (FindAndFixCostCap) * (1 / aEP);
        FindAndFixCostCap2 := (FindAndFixCostCap) * (1 / aEP);
      end
      else
        Variables[fI.cct_dssa_cost_umra] := 0;
      Variables[fI.cct_dssa_cost_umra_om] := (FindAndFixCostOM - AdjustCCTCostOM) *
(1 / aEP);
      Variables[fI.cct_dssa_cost_p] :=
        (FindAndFixCostOM + FindAndFixCostCapDisc_p - AdjustCCTCostOM) * (1 / aEP);
    end
    else if (CCTB = 0) and (hff_cct = 3) then begin
      Variables[fI.cct_dssa_cost] :=
        (FindAndFixCostOM + FindAndFixCostCapDisc - InstallCCTCostOM) * (1 / aEP);
      if (y - hffY2) MOD UsefulLifeFF = 0 then begin
        Variables[fI.cct_dssa_cost_umra] := (FindAndFixCostCap) * (1 / aEP);
        FindAndFixCostCap2 := (FindAndFixCostCap) * (1 / aEP);
      end
      else
        Variables[fI.cct_dssa_cost_umra] := 0;
      Variables[fI.cct_dssa_cost_umra_om] := (FindAndFixCostOM - InstallCCTCostOM) *
(1 / aEP);
      Variables[fI.cct_dssa_cost_p] :=
        (FindAndFixCostOM + FindAndFixCostCapDisc_p - InstallCCTCostOM) * (1 / aEP);
    end
    else if (CCTB = 1) and (hff_cct >= 4) then begin
      Variables[fI.cct_dssa_cost] :=
        (FindAndFixCostOM + FindAndFixCostCapDisc - AdjustCCTCostOM);
      if (y - hFFY3) MOD UsefulLifeFF = 0 then begin
        Variables[fI.cct_dssa_cost_umra] := (FindAndFixCostCap);
        FindAndFixCostCap2 := FindAndFixCostCap;
      end
      else
        Variables[fI.cct_dssa_cost_umra] := 0;
      Variables[fI.cct_dssa_cost_umra_om] := (FindAndFixCostOM - AdjustCCTCostOM);
      Variables[fI.cct_dssa_cost_p] :=
        (FindAndFixCostOM + FindAndFixCostCapDisc_p - AdjustCCTCostOM);
    end
```

```
else if (CCTB = 0) and (hff_cct >= 4) then begin
  Variables[fI.cct_dssa_cost] :=
    (FindAndFixCostOM + FindAndFixCostCapDisc - InstallCCTCostOM);
  if (y - hFFY3) MOD UsefulLifeFF = 0 then begin
    Variables[fI.cct_dssa_cost_umra] := (FindAndFixCostCap);
    FindAndFixCostCap2 := FindAndFixCostCap;
  end
  else
    Variables[fI.cct_dssa_cost_umra] := 0;
  Variables[fI.cct_dssa_cost_umra_om] := (FindAndFixCostOM - InstallCCTCostOM);
  Variables[fI.cct_dssa_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 (option = 'Baseline') and (Config.BaselineName = 'LCRR') then begin
  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;
end else begin
  if pws90pct > bp2 then
    owBin_tmp := 1
  else
    owBin_tmp := 3;
end;

// 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;

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

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

```
    end;

    // Compute Costs
    TotalPWSCost := 0;
    for iC := 0 to CostSteps.Count - 1 do begin
      c := CostSteps.Items[iC];
      // 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;

{
  Dollar year adjustments from Jerry email 7/28/23
  From  To      Multiplier
  2021  2022     1.069
  2020  2022     1.095
  2021  2020     0.977

  For costs from database, including state costs, use 1.095
}

      if _UseCompiled then begin
        Cost := CC._Cost[c.fCostStepRec.ID];
        Cost := Cost * 1.095;
        Labor := CC._Labor[c.fCostStepRec.ID];
        Labor := Labor * 1.095;
        OM := CC._OM[c.fCostStepRec.ID];
        OM := OM * 1.095;
        Hours := CC._Hours[c.fCostStepRec.ID];
      end
      else
        c.Evaluate(Cost, Labor, OM, Hours, DoIt);

      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') and (Config.BaselineName = 'LCR') 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 = 'Baseline') and (Config.BaselineName = 'LCRR') 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
          else if (option = 'LCRI') 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;

          TotalPWSCost := TotalPWSCost + GetTotalPWSCost(acTotalPWSCost,
c.fCostStepRec, Cost);
        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;

      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;
    end; // end C in CostSteps loop
    // end Compute Costs

    if Num_Proxies = 0 then
    begin
      CapitalCostNotAnnualized := 0;
      if (b_install_cct[y] + b_modify_cct[y] > 0) then begin
        if (y - cct_adjust_yr) MOD UsefulLifeMod = 0 then
          CapitalCostNotAnnualized := CapitalCostNotAnnualized + AdjustCCTCostCap;

        if (y - cct_install_yr) MOD UsefulLifeMod = 0 then
          CapitalCostNotAnnualized := CapitalCostNotAnnualized + InstallCCTCostCap;
      end;

      if FindAndFixCostCap2 > 0 then
        CapitalCostNotAnnualized := CapitalCostNotAnnualized + FindAndFixCostCap2;

      if not LLL.Exists('PWSCost_' + Config.RunName) then
        LLL.L('PWSCost_' +
Config.RunName,'PWSId,SystemSize,Ownership,SourceWater,SystemType,y,TotalPWSCost,Cap
italCost,Weight');

      LLL.L('PWSCost_' + Config.RunName, pwsid + ',' +
CostingData.SystemSize.ToString + ',' +
            CostingData.Ownership.ToString + ',' + CostingData.SourceWater.ToString
+ ',' +
            CostingData.SystemType.ToString + ',' + y.ToString + ',' +
TotalPWSCost.ToString + ',' +
```

```
            CapitalCostNotAnnualized.ToString + ',' + pwswgt.ToString);
    end;

    // 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;
          // 1.184 GDP inflator from 2016 to 2020
          POTWCost := POTWCost + ((Discount((FlowLossP - ConnectionLossP), y - 1,
CostCapital)) * 1.184);
        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
    if NewDraw = true then
      fCostVars.FillValueArray(Variables, RawVariables, CostingData.SystemSize,
```

```
CostingData.SourceWater, pws_lsl,
                                CostingData.CCT, CostingData.SystemType, Y+1,
Config.PWS90PctBp1,
                                Config.PWS90PctBp2, SetProbsTo01, NewDraw, O,
isBaseline);

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

    Variables[fI.Pws_pop] := aPop;

    InitCCTBVarsToZero(option);

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

    if not IsBaseline 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);

      num_lsl_replace_y := num_lsl_replace_fed[y];
      num_unknown_resolved_y := num_unknown_resolved[y];
      if Num_Proxies = 0 then begin
        SetFH();
        LeadConcentrationBins(pwsid, option, y, CostingData.SystemSize,
CostingData.SourceWater, pws_lsl, CostingData.CCT,
          POU, aPop, aNC, fAdjust_CCT,
          fInstall_CCT, cct_adjust_yr, cct_install_yr, pou_install_yr, bCCT_Change,
pwswgt,
          Num_lsl_replace_fed[y], num_lsl_requested[y], num_lsl_remain, perc_lsl_b,
Num_Proxies,
          partial_cct_level, Variables[fI.pp_lslr_lsl], pp_lslr_lsl_adj,
num_lsl_filters,
          num_hh_per_connect, num_temp_pou, false);
      end;
    end;

    if 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] + pwswgt;
    end;
```

```
  end; // end year loop

  SumTotLSLReplace := SumTotLSLReplaceMand + SumTotLSLReplaceVol +
                      SumTotLSLPartialReplaceMand + SumTotLSLPartialReplaceVol +
                      SumTotLSLGalPrevMand + SumTotLSLGalPrevVol +
                      SumTotLSLLeadConMand + SumTotLSLLeadConVol +
                      SumTotLSLGalPrevLeadConMand + SumTotLSLGalPrevLeadConVol;

  SumFedLSLReplace := SumFedLSLReplaceMand + SumFedLSLReplaceVol +
                      SumFedLSLPartialReplaceMand + SumFedLSLPartialReplaceVol +
                      SumFedLSLGalPrevMand + SumFedLSLGalPrevVol +
                      SumFedLSLLeadConMand + SumFedLSLLeadConVol +
                      SumFedLSLGalPrevLeadConMand + SumFedLSLGalPrevLeadConVol;

  if not VLSystem then
    Annualize(CostCapital);
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 pws_state: string; 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, FindAndFixCostCap2, FindAndFixCostCapDisc, AdjustCCTCostCap,
AdjustCCTCostCapDisc: double;
    AdjustCCTCostCapDisc_p, FindAndFixCostCapDisc_p: double;

    Num_LSL_base, Num_potential_LSL_base, Num_potential_LSL_known: double;
    num_sl_known: double;
    num_lsl_known, num_sl_unknown, num_lsl_unknown_lead, num_lsl_nolead_unknown:
double;
    perc_lsl_b,perc_unknown_nonlead_b: double;
    num_unknown_remain: double;

    pp_lsl_replaced, lslr_missed_goal: array[1..100] of double;
```

```
Num_lsl_replace_ale: array[1..100] of double;
Num_lsl_replace_state: array[1..100] of double;
Num_lsl_replace_fed: array[1..100] of double;
Num_lsl_replace_rule: array[1..100] of double;
Num_lsl_replace_tot: array[1..100] of double;
num_unknown_resolved: array[1..100] of double;
Num_lsl_replace_rule_all: array[1..100] of double;
Num_lsl_testout: array[1..100] of double;
num_lsl_validated: array[1..100] of double;
Num_lsl_validate_pool: array[1..100] of double;

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;
fed_num_lslr_lsl_replace: array[0..100] of double;
fed_num_lslr_partial_replace: array[0..100] of double;
fed_num_lslr_gal_prev_lsl_replace: array[0..100] of double;
fed_num_lslr_leadcon_replace: array[0..100] of double;
fed_num_lslr_galprev_leadcon_replace: array[0..100] of double;
tot_num_lslr_lsl_replace: array[0..100] of double;
tot_num_lslr_partial_replace: array[0..100] of double;
tot_num_lslr_gal_prev_lsl_replace: array[0..100] of double;
tot_num_lslr_leadcon_replace: array[0..100] of double;
tot_num_lslr_galprev_leadcon_replace: array[0..100] of double;
num_temp_pou: double;
lsl_replace_pct: double;
num_lsl_filters: double;
Num_potential_LSL_base_pws_vls: double;

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

isBaseline: boolean;

cct_adjust_yr, cct_install_yr, pou_install_yr, cct_change_yr: integer;

num_lsl_base_adjust: double;
partial_cct_level: integer;

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, proxy2_pws90_y4: 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;
aSz, aSrc, aLSL, aCCT, aType, aEP, aNC, aPop, num_lsl: integer;
BinChg: 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;

  SourceTreatChangeEver: boolean;
  lsl_start_yr: integer;

  LCRI_Bin_initial: integer;
  LCRI_Bin: integer;
  LCRI_P90: integer;
  LCRI_Bin_15, LCRI_Bin_12, LCRI_Bin_10, LCRI_Bin_5: integer;

  pp_lslr_lsl_adj: double;
  P90_concurrent: double;
  pp_above_al_bin_one, pp_above_al_bin_two, pp_above_al_bin_three: double;
  pp_lsl_replaced_mand_pct: double;
  pp_lsl_replaced_mand_pct_small: double;

  b_ale_not_achieved: integer;
  state_lslr_rate: double;

  temp_int: integer;
  lslr_newpath: integer;
  hh_unknown_remain: double;
  lsl_replace_pct_a, lsl_replace_pct_b, lsl_replace_pct_c: double;
  perc_lsl_known_vls: double;

num_lsl_replace_y: double;
num_unknown_resolved_y: double;
Num_lsl_replace_rule_allb: double;
CapitalCostNotAnnualized: double;
num_lsl_lead_dwns: double;

procedure SetFH();
begin
  FH.pwsid := Costingdata.pwsid;
  FH.aNC := aNC;
  FH.perc_unknown_nonlead_b := perc_unknown_nonlead_b;
  FH.num_sl_unknown := num_sl_unknown;
  if FH.aNC > 0 then
    FH.pop_per_connection := aPop / aNC
  else
    FH.pop_per_connection := 0;

  FH.num_lsl_replaced := num_lsl_replace_y;
  FH.pp_lslr_lsl_adj := pp_lslr_lsl_adj;
  FH.num_unknown_resolved := num_unknown_resolved_y;
end;
```

```
begin
  fnum_proxies := 0;

  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);

  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;

  HasLSLRCost := false;
  HasCCTCost := false;
```

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


SumFedLSLReplace := 0;
SumFedLSLReplaceMand := 0;
SumFedLSLPartialReplaceMand := 0;
SumFedLSLGalPrevMand := 0;
SumFedLSLLeadConMand := 0;
SumFedLSLGalPrevLeadConMand := 0;

SumFedLSLReplaceVol := 0;
SumFedLSLPartialReplaceVol := 0;
SumFedLSLGalPrevVol := 0;
SumFedLSLLeadConVol := 0;
SumFedLSLGalPrevLeadConVol := 0;
SumTotLSLReplace := 0;
SumTotLSLReplaceMand := 0;
SumTotLSLPartialReplaceMand := 0;
SumTotLSLGalPrevMand := 0;
SumTotLSLLeadConMand := 0;
SumTotLSLGalPrevLeadConMand := 0;

SumTotLSLReplaceVol := 0;
SumTotLSLPartialReplaceVol := 0;
SumTotLSLGalPrevVol := 0;
SumTotLSLLeadConVol := 0;
SumTotLSLGalPrevLeadConVol := 0;

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

cct_adjust_yr := 0;
cct_install_yr := 0;
cct_change_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(fed_num_lslr_lsl_replace, SizeOf(fed_num_lslr_lsl_replace), 0);
  fillchar(fed_num_lslr_partial_replace, SizeOf(fed_num_lslr_partial_replace), 0);
  fillchar(fed_num_lslr_gal_prev_lsl_replace,
SizeOf(fed_num_lslr_gal_prev_lsl_replace), 0);
  fillchar(fed_num_lslr_leadcon_replace, SizeOf(fed_num_lslr_leadcon_replace), 0);
```

```
  fillchar(fed_num_lslr_galprev_leadcon_replace,
SizeOf(fed_num_lslr_galprev_leadcon_replace), 0);
  fillchar(tot_num_lslr_lsl_replace, SizeOf(tot_num_lslr_lsl_replace), 0);
  fillchar(tot_num_lslr_partial_replace, SizeOf(tot_num_lslr_partial_replace), 0);
  fillchar(tot_num_lslr_gal_prev_lsl_replace,
SizeOf(tot_num_lslr_gal_prev_lsl_replace), 0);
  fillchar(tot_num_lslr_leadcon_replace, SizeOf(tot_num_lslr_leadcon_replace), 0);
  fillchar(tot_num_lslr_galprev_leadcon_replace,
SizeOf(tot_num_lslr_galprev_leadcon_replace), 0);

  perc_unknown_nonlead_b := 0;
  Num_potential_LSL_base_pws_vls := 0;

  SourceTreatChangeEver := false;



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

  aSz := CostingData.SystemSize;
  aSrc :=  CostingData.SourceWater;
  aType := CostingData.SystemType;
  aEp := VLSEpWorkbook.NumberEPs;
  aPop := VLSEpWorkbook.Population;
  aNC := trunc(CostingData.VLSConnections / CostingData.VLSNumEPs);
  num_lsl := VLSEpWorkbook.NumberLSLs;

  // read data request data values from database
  fCostVars.FillValueArray(Variables, RawVariables, aSz, aSrc, LSL, CCT, aType, 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] := aEp;
  Variables[fI.Pws_Cct] := CCT;

  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_dssa_cost] := 0;

fillchar(pp_lsl_replaced,SizeOf(pp_lsl_replaced),0);
fillchar(Num_lsl_replace_ale,SizeOf(Num_lsl_replace_ale),0);
fillchar(Num_lsl_replace_state,SizeOf(Num_lsl_replace_state),0);
fillchar(Num_lsl_replace_fed,SizeOf(Num_lsl_replace_fed),0);
fillchar(Num_lsl_replace_rule,SizeOf(Num_lsl_replace_rule),0);
fillchar(Num_lsl_replace_tot,SizeOf(Num_lsl_replace_tot),0);
fillchar(num_unknown_resolved,SizeOf(num_unknown_resolved),0);
fillchar(Num_lsl_replace_rule_all,SizeOf(Num_lsl_replace_rule_all),0);
fillchar(lslr_missed_goal,SizeOf(lslr_missed_goal),0);
fillchar(num_lsl_requested,SizeOf(num_lsl_requested),0);
fillchar(Num_lsl_testout,SizeOf(Num_lsl_testout),0);
fillchar(num_lsl_validated,SizeOf(num_lsl_validated),0);
fillchar(num_lsl_validate_pool,SizeOf(num_lsl_validate_pool),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_requested := 0;
Num_potential_LSL_base := 0;
num_lsl_known := 0;
```

```
num_sl_unknown := 0;
num_lsl_nolead_unknown := 0;
num_unknown_remain := 0;
lslr_newpath := 0;
LSLRequested := 0;

bp1 := 10;

if option = 'Baseline' then  begin
  bp1 := 10;
  bp2 := 15;
  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;

  if pws_state = 'MI' then begin
    if owBin = 1 then
      state_lslr_rate := 0.07
    else
      state_lslr_rate := 0.05;
  end
  else if pws_state = 'IL' then begin
    if num_lsl <= 1200  then
      state_lslr_rate := 0.07
    else if (num_lsl > 1200) and (num_lsl < 5000) then
      state_lslr_rate := 0.06
    else if (num_lsl >= 5000) and (num_lsl < 10000) then
      state_lslr_rate := 0.05
    else if (num_lsl > 10000) and (num_lsl < 100000) then
      state_lslr_rate := 0.03
    else if num_lsl > 100000 then
      state_lslr_rate := 0.02;
  end
  else if pws_state = 'NJ' then begin
    state_lslr_rate := 0.10;
  end
  else
    state_lslr_rate := 0;
end
else if option = 'LCRI' then begin
  bp2 := Config.ALE;
  O.TryGetValue('bBin', tmp_double);
  LCRI_Bin_initial := trunc(tmp_double);

  MakeLCRxBin(option, Config.ALE, LCRI_Bin_initial, LCRI_BIN, LCRI_P90);

  owBin := LCRI_Bin;
  pws90pct := LCRI_P90;
```

```
    O.TryGetValue('bALENotAchieved', tmp_double);
    b_ale_not_achieved := trunc(tmp_double);

    if pws_state = 'MI' then begin
      if owBin = 1 then
        state_lslr_rate := 0.07
      else
        state_lslr_rate := 0.05;
    end
    else if pws_state = 'IL' then begin
      if num_lsl <= 1200  then
        state_lslr_rate := 0.07
      else if (num_lsl > 1200) and (num_lsl < 5000) then
        state_lslr_rate := 0.06
      else if (num_lsl >= 5000) and (num_lsl < 10000) then
        state_lslr_rate := 0.05
      else if (num_lsl > 10000) and (num_lsl < 100000) then
        state_lslr_rate := 0.03
      else if num_lsl > 100000 then
        state_lslr_rate := 0.02;
    end
    else if pws_state = 'NJ' then begin
      state_lslr_rate := 0.10;
    end
    else
      state_lslr_rate := 0;
  end; // option = 'LCRI'

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

  if LSL > 0 then  begin
    num_lsl_known := VLSEpWorkbook.num_lsl_known_vls;
    Perc_lsl_known_vls := num_lsl_known / VLSEpWorkbook.Connections;
    Num_lsl_unknown_lead := VLSEpWorkbook.num_lsl_unknown_lead_vls;
    Num_sl_unknown := VLSEpWorkbook.num_lsl_unknown_vls;

    num_lsl_base := num_lsl_known + Num_sl_unknown;

    perc_lsl_b := num_lsl_base / VLSEpWorkbook.Connections;

    if aType = 1 then
    begin
      if num_sl_unknown > 0 then
```

```
    perc_unknown_nonlead_b := 1 - (num_lsl_unknown_lead / num_sl_unknown)
  else
    perc_unknown_nonlead_b := 1;
end
else
  perc_unknown_nonlead_b := 0;

if option = 'Baseline' then begin
  if Config.BaselineName = 'LCR' then
  begin
    Num_potential_LSL_base := num_lsl_known *
      (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial]);
    Variables[fI.pp_lslr_leadcon] := 0;
    Variables[fI.pp_lslr_galprev_leadcon] := 0;
    Variables[fI.pp_lslr_gal_prev_lsl] := 0;

    Num_potential_LSL_known := num_lsl_known *
      (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial]);

    if Num_potential_LSL_base > 0 then
      Variables[fI.pws_lsl] := 1;

    pp_lslr_lsl_adj := Variables[fI.pp_lslr_lsl] /
      (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial]);
  end
  else if Config.BaselineName = 'LCRR' then
  begin
    Num_potential_LSL_base := Num_LSL_base *
      (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial] +
      Variables[fI.pp_lslr_gal_prev_lsl]);
    Variables[fI.pp_lslr_leadcon] := 0;
    Variables[fI.pp_lslr_galprev_leadcon] := 0;

    Num_potential_LSL_known := num_lsl_known *
      (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial] +
      Variables[fI.pp_lslr_gal_prev_lsl]);

    if Num_potential_LSL_base > 0 then
      Variables[fI.pws_lsl] := 1;

    pp_lslr_lsl_adj := Variables[fI.pp_lslr_lsl] /
      (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial] +
      Variables[fI.pp_lslr_gal_prev_lsl]);
  end;
end
else if option = 'LCRI' then begin
  if Config.LSLOption = 1 then begin
    Num_potential_LSL_base := Num_LSL_base *
      (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial] +
      Variables[fI.pp_lslr_gal_prev_lsl]);
```

```
        Num_potential_LSL_base_pws_vls := VLSEpWorkbook.num_lsl_pws_vls *
          (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial] +
          Variables[fI.pp_lslr_gal_prev_lsl]);


        Variables[fI.pp_lslr_leadcon] := 0;
        Variables[fI.pp_lslr_galprev_leadcon] := 0;

        Num_potential_LSL_known := num_lsl_known *
          (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial] +
          Variables[fI.pp_lslr_gal_prev_lsl]);

        pp_lslr_lsl_adj := Variables[fI.pp_lslr_lsl] /
          (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial] +
          Variables[fI.pp_lslr_gal_prev_lsl]);
      end else if Config.LSLOption = 2 then begin
        Variables[fI.pp_lslr_galprev_leadcon] := 0;

        Num_potential_LSL_base := Num_LSL_base *
          (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial] +
          Variables[fI.pp_lslr_gal_prev_lsl] + Variables[fI.pp_lslr_leadcon]);

        Num_potential_LSL_base_pws_vls := VLSEpWorkbook.num_lsl_pws_vls *
          (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial] +
          Variables[fI.pp_lslr_gal_prev_lsl] + Variables[fI.pp_lslr_leadcon]);

        Num_potential_LSL_known := num_lsl_known *
          (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial] +
          Variables[fI.pp_lslr_gal_prev_lsl] + Variables[fI.pp_lslr_leadcon]);

        pp_lslr_lsl_adj := Variables[fI.pp_lslr_lsl] /
          (Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial] +
Variables[fI.pp_lslr_leadcon]
          + Variables[fI.pp_lslr_gal_prev_lsl]);
      end else begin
        Num_potential_LSL_base := Num_LSL_base;

        Num_potential_LSL_base_pws_vls := VLSEpWorkbook.num_lsl_pws_vls;

        pp_lslr_lsl_adj := Variables[fI.pp_lslr_lsl];

        Num_potential_LSL_known := num_lsl_known;
      end;

      lsl_replace_pct := Config.LSLRepPct;

      // deferrals
      if lsl_replace_pct * VLSEpWorkbook.num_lsl_pws_vls > Config.LSLRCap then begin
        lsl_replace_pct_a := Config.LSLRCap / VLSEpWorkbook.num_lsl_pws_vls;
```

```
        deferral_cap := 1;
      end else begin
        lsl_replace_pct_a := lsl_replace_pct;
        deferral_cap := 0;
      end;

      if lsl_replace_pct * Num_potential_LSL_base > (0.039 * (aPop /
Variables[fI.Numb_hh])) then begin
        lsl_replace_pct_b := (0.039 * (aPop / Variables[fI.Numb_hh])) /
Num_potential_LSL_base;
        deferral_pct := 1;
      end else begin
        lsl_replace_pct_b := lsl_replace_pct;
        deferral_pct := 0;
      end;

      if (deferral_cap * deferral_pct) = 1 then begin
        lsl_replace_pct := min(lsl_replace_pct_a, lsl_replace_pct_b);
        deferral_any := 1;
      end else begin
        if deferral_cap = 1 then lsl_replace_pct := lsl_replace_pct_a
        else if deferral_pct = 1 then lsl_replace_pct := lsl_replace_pct_b
        else lsl_replace_pct := lsl_replace_pct;

        deferral_any := 0;
      end;
    end;

    if Num_potential_LSL_base > 0 then Variables[fI.pws_lsl] := 1;
    num_lsl_remain := Num_potential_LSL_base;
    hh_remain_lsl := max(0, (num_hh_per_connect * num_lsl_remain));
  end;

  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('BinChg_' + i.ToString, tmp_double);
    BinChg[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[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 *
      (CostingData.CostCapital / (1 - Power((1 + CostingData.CostCapital),
-CCTCostEquations.UsefulLifeCap)));

  fAdjust_CCT := 0;
  fInstall_CCT := 0;

  lslr_conducted := false;
  if (option = 'Baseline') and (config.BaselineName = 'LCRR') 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;

    pp_above_al_bin_one := Variables[fI.pp90aboveal15_1];
    pp_above_al_bin_two := Variables[fI.pp90aboveal15_2] +
Variables[fI.pp90aboveal15_3];
    pp_above_al_bin_three := Variables[fI.pp90aboveal15_4] +
Variables[fI.pp90aboveal15_5];
    Variables[fI.pp_above_al_bin_one] := pp_above_al_bin_one;
    Variables[fI.pp_above_al_bin_two] := pp_above_al_bin_two;
    Variables[fI.pp_above_al_bin_three] := pp_above_al_bin_three;
  end;
  if option = 'LCRI' 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.ALE = 15 then begin
      pp_above_al_bin_one := Variables[fI.pp90aboveal15_1];
      pp_above_al_bin_three := Variables[fI.pp90aboveal15_2] +
Variables[fI.pp90aboveal15_3] +
        Variables[fI.pp90aboveal15_4] + Variables[fI.pp90aboveal15_5];
    end
    else if Config.ALE = 12 then begin
      pp_above_al_bin_one := Variables[fI.pp90aboveal12_1] +
Variables[fI.pp90aboveal12_2];
      pp_above_al_bin_three := Variables[fI.pp90aboveal12_3] +
Variables[fI.pp90aboveal12_4] +
        Variables[fI.pp90aboveal12_5];
    end
    else if Config.ALE = 10 then begin
      pp_above_al_bin_one := Variables[fI.pp90aboveal10_1] +
Variables[fI.pp90aboveal10_2] +
        Variables[fI.pp90aboveal10_3];
      pp_above_al_bin_three := Variables[fI.pp90aboveal10_4] +
Variables[fI.pp90aboveal10_5];
    end
    else if Config.ALE = 5 then begin
      pp_above_al_bin_one := Variables[fI.pp90aboveal5_1] +
Variables[fI.pp90aboveal5_2] +
        Variables[fI.pp90aboveal5_3] + Variables[fI.pp90aboveal5_4];
      pp_above_al_bin_three := Variables[fI.pp90aboveal5_5];
    end;

    Variables[fI.pp_above_al_bin_one] := pp_above_al_bin_one;
    Variables[fI.pp_above_al_bin_two] := pp_above_al_bin_two;
    Variables[fI.pp_above_al_bin_three] := pp_above_al_bin_three;
  end;

  CCT_Change := false;
  bCCT_Change := false;

  num_lsl_replace_y := 0;
  num_unknown_resolved_y := 0;
  if AddCostingData.Num_Proxies = 0 then begin
    SetFH();
    LeadConcentrationBins(CostingData.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, CostingData.SamplingWeight,
                          0, 0, 0, perc_lsl_b, AddCostingData.Num_Proxies,
partial_cct_level,
                          Variables[fI.pp_lslr_lsl], pp_lslr_lsl_adj, num_lsl_filters,
```

```
num_hh_per_connect, num_temp_pou, True);
  end;

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

  num_replace := 0;
  num_remain := 0;

  // ************
  // Year loop
  // ************
  for Y:=1 to fYears do begin
    if (AddCostingData.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 ='LCRI') or ((option = 'Baseline') and (Config.BaselineName =
'LCRR')) 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.pp_grandfather_mand_child] :=
SchoolSampData.pp_grandfather_mand_child;
      Variables[fI.pp_grandfather_opt_child] :=
SchoolSampData.pp_grandfather_opt_child;
      Variables[fI.pp_grandfather_mand_pub_elem] :=
SchoolSampData.pp_grandfather_mand_pub_elem;
      Variables[fI.pp_grandfather_opt_pub_elem] :=
SchoolSampData.pp_grandfather_opt_pub_elem;
      Variables[fI.pp_grandfather_mand_priv_elem] :=
SchoolSampData.pp_grandfather_mand_priv_elem;
      Variables[fI.pp_grandfather_opt_priv_elem] :=
SchoolSampData.pp_grandfather_opt_priv_elem;
      Variables[fI.pp_grandfather_opt1_pub_second] :=
SchoolSampData.pp_grandfather_opt1_pub_second;
      Variables[fI.pp_grandfather_opt2_pub_second] :=
SchoolSampData.pp_grandfather_opt2_pub_second;
      Variables[fI.pp_grandfather_opt1_priv_second] :=
SchoolSampData.pp_grandfather_opt1_priv_second;
      Variables[fI.pp_grandfather_opt2_priv_second] :=
SchoolSampData.pp_grandfather_opt2_priv_second;
    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;

    NewDraw := false;

    // Baseline
    // Baseline Source and Treatment Change
    if option = 'Baseline' then begin
      if Config.BaselineName = 'LCR' 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
              tBin := BinChg[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
              tBin := BinChg[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 begin
            cct_install_yr := y + 5;
            cct_change_yr := cct_install_yr;
          end;

          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 begin
            cct_install_yr := y + 4;
            cct_change_yr := cct_install_yr;
          end;
          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;

            proxy3_pws90[y + 6] := Variables[fI.post_cct_p90_bin1];
            CCT_Change := True;
            if cct_adjust_yr = 0 then begin
              cct_adjust_yr := y + 4;
              cct_change_yr := cct_adjust_yr;
            end;

            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;

            proxy3_pws90[y + 5] := Variables[fI.post_cct_p90_bin1];
            CCT_Change := True;
            if cct_adjust_yr = 0 then begin
              cct_adjust_yr := y + 3;
              cct_change_yr := cct_adjust_yr;
            end;

            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
      else if Config.BaselineName = 'LCRR' 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 >= 1 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_dssa] *
                Variables[fI.pp_above_al_bin_three] * Variables[fI.numb_reduced_tap]
            else if (Variables[fI.p_tap_triennial] = 1) and
              ((y = 1) or (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_dssa] *
                Variables[fI.pp_above_al_bin_three] * Variables[fI.numb_reduced_tap]
            else if (Variables[fI.p_tap_nine] = 1) and ((y = 1) or (y = 10) or (y =
19) or (y = 28))
            then
              numb_wqp_add_sites[y] := Variables[fI.pp_overlap_dssa] *
                Variables[fI.pp_above_al_bin_three] * Variables[fI.numb_reduced_tap]
            else
              numb_wqp_add_sites[y] := Variables[fI.pp_overlap_dssa] *
                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_dssa] *
              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_dssa] *
              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]);

          // 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
              tBin := BinChg[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
              tBin := BinChg[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 LSL > 0 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;

          // CCT and POU LCRR Baseline
          if (proxy2_pws90 > bp1) and (proxy2_pws90 < bp2) and (CCT = 1) and (not
CCT_Change) then
          begin
            // bin 2
            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 if (proxy2_pws90 > bp2) and (CCT = 1) and (not CCT_Change) then begin
            // bin 1
            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;
            // System Capital Cost undiscounted
            ValuesCapital[2] := ValuesCapital[2] + InstallCCTCostCap;
          end;

          if proxy3_pws90[y] = 0 then
            proxy3_pws90[y] := proxy2_pws90;
        end; // y >= 1
      end; // if Config.BaselineName = 'LCRR'
    end // optiopn = 'Baseline'
    else if option = 'LCRI' 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 > 0 then Variables[fI.b_lslr_study] := 1;

        //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_dssa] *
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_dssa] *
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_dssa] *
pp_above_al_bin_three *
```

```
              Variables[fI.numb_reduced_tap]
          else
            numb_wqp_add_sites[y] := Variables[fI.pp_overlap_dssa] *
pp_above_al_bin_three *
              (2 * Variables[fI.numb_samp_customer]);
        end
        else if owBin = 1 then
          numb_wqp_add_sites[y] := Variables[fI.pp_overlap_dssa] *
            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]);

        //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
            MakeLCRxBin(option, Config.ALE, BinChg[y], tBin, temp_int);

            if tBin = 1 then begin
              if Config.ALE = 15 then
                tpws90pct := 20
              else if Config.ALE = 12 then
                tpws90pct := 15
              else if Config.ALE = 10 then
                tpws90pct := 12
              else if Config.ALE = 5 then
                tpws90pct := 8;
            end else begin
              if Config.ALE = 15 then
                tpws90pct := 10
              else if Config.ALE = 12 then
                tpws90pct := 8
              else if Config.ALE = 10 then
```

```
              tpws90pct := 5
            else if Config.ALE = 5 then
              tpws90pct := 2;
          end;

          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
          MakeLCRxBin(option, Config.ALE, BinChg[y], tBin, temp_int);

          if tBin = 1 then begin
            if Config.ALE = 15 then
              tpws90pct := 20
            else if Config.ALE = 12 then
              tpws90pct := 15
            else if Config.ALE = 10 then
              tpws90pct := 12
            else if Config.ALE = 5 then
              tpws90pct := 8;
            ToBin1Count := 1;
          end else begin
            if Config.ALE = 15 then
              tpws90pct := 10
            else if Config.ALE = 12 then
              tpws90pct := 8
            else if Config.ALE = 10 then
              tpws90pct := 5
            else if Config.ALE = 5 then
              tpws90pct := 2;
          end;

          proxy2_pws90 := tpws90pct;

          SourceTreatChangeEver := True;
        end;
      end;

      if y = 4 then
      begin
```

```
          proxy2_pws90_y4 := proxy2_pws90;

          if (LSL > 0) and (Num_potential_LSL_base < 80) and (proxy2_pws90 > bp2)
then
          begin
            lslr_newpath := 1;
            lsl_replace_pct := 0.2;
          end;
        end;

        // *********************
        // CCT, LSLR, and POU Plans/Studies
        // *********************

        if LSL > 0 then
          Variables[fI.b_lslr_study] := 1;

        // CCT and POU

        // Modify CCT
        if ((aType = 1) and (aPop > Config.SmallProxyPop) and (proxy2_pws90 > bp2)
and
            (CCT = 1) and (not CCT_Change) and (POU = 0) and
            ((lslr_newpath <> 1) or ((lslr_newpath = 1) and (LSL = 0)))) or

           ((aType = 2) and (proxy2_pws90 > bp2) and (CCT = 1) and (not CCT_Change)
and (POU = 0) and
            ((lslr_newpath <> 1) or ((lslr_newpath = 1) and (LSL = 0)))) 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;

          //NOTELCRI check on post_cct_p90_bin1

          if (Config.ALE = 5) and (b_ale_not_achieved = 1) then
            proxy3_pws90[y+6] := 6
          else
            proxy3_pws90[y+6] := proxy2_pws90 * 0.2;

          CCT_Change := true;
          if cct_adjust_yr = 0 then
          begin
            cct_adjust_yr := y + 4;
            cct_change_yr := cct_adjust_yr;
          end;

          b_cct_study_mod_al[y+3] := 1;
```

```
        end
        // Install CCT
        else
        if ((aType = 1) and (aPop > Config.SmallProxyPop) and (proxy2_pws90 > bp2)
and
            (CCT = 0) and (not CCT_Change) and (POU = 0) and
            ((lslr_newpath <> 1) or ((lslr_newpath = 1) and (LSL = 0)))) or

           ((aType = 2) and (proxy2_pws90 > bp2) and (CCT = 0) and (not CCT_Change)
and (POU = 0) and
            ((lslr_newpath <> 1) or ((lslr_newpath = 1) and (LSL = 0)))) then
        begin
          for i := y + 4 to Config.YearsOfAnalysis do
          begin
            InstallCCT[i] := true;
            b_install_cct[i] := 1;
          end;

          if cct_install_yr = 0 then
          begin
            cct_install_yr := y + 4;
            cct_change_yr := cct_install_yr;
          end;

          b_cct_study_install[y+3] := 1;

          if (Config.ALE = 5) and (b_ale_not_achieved = 1) then
            proxy3_pws90[y+6] := 6
          else
            proxy3_pws90[y+6] := proxy2_pws90 * 0.2;

          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; // y >= 4
    end; // end option = 'LCRI'

    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;

    proxy4_pws90 := -1;
    if LSL > 0 then begin
      if option = 'Baseline' then begin
        if Config.BaselineName = 'LCR' then
        begin
          if y >= 4 then begin
            Num_lsl_replace_state[y] := min(state_lslr_rate *
Num_potential_LSL_known, Num_lsl_remain);

            if proxy3_pws90[y] > bp2 then
              Num_lsl_replace_rule_all[y] := min(0.07 * Num_potential_LSL_known,
Num_lsl_remain);

            Num_lsl_replace_rule[y] := Num_lsl_replace_rule_all[y] * (1 -
Variables[fI.pp_lcr_test]);
            Num_lsl_testout[y] := Num_lsl_replace_rule_all[y] *
Variables[fI.pp_lcr_test];

            Num_lsl_replace_tot[y] := max(Num_lsl_replace_rule[y],
Num_lsl_replace_state[y]);
            Num_lsl_replace_fed[y] := Num_lsl_replace_tot[y] -
Num_lsl_replace_state[y];

            // NOTE in Baseline unknowns are not investigated NEEDS TO GO TO
WORKBOOK
            num_unknown_resolved[y] := min(0.1 * num_sl_unknown,
num_unknown_remain);

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

            if Num_lsl_replace_rule_allb >= Num_potential_LSL_known * 0.21 then
```

```
begin
              proxy4_pws90 := bp1 + ((bp2 - bp1) / 2);
            end;

            // NOTE After Num_replace >= Num_potential_LSL_known * 0.21 do not do
any rule replacement.

            tot_num_lslr_lsl_replace[y] := num_lsl_replace_tot[y] *
                                           (Variables[fI.pp_lslr_lsl] /
(Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial]));

            tot_num_lslr_partial_replace[y] := num_lsl_replace_tot[y] *
                                           (Variables[fI.pp_lslr_partial] /
(Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial]));

            fed_num_lslr_lsl_replace[y] := num_lsl_replace_fed[y] *
                                           (Variables[fI.pp_lslr_lsl] /
(Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial]));

            fed_num_lslr_partial_replace[y] := num_lsl_replace_fed[y] *
                                           (Variables[fI.pp_lslr_partial] /
(Variables[fI.pp_lslr_lsl] + Variables[fI.pp_lslr_partial]));

            tot_num_lslr_gal_prev_lsl_replace[y] := 0;
            tot_num_lslr_leadcon_replace[y] := 0;
            tot_num_lslr_galprev_leadcon_replace[y] := 0;

            fed_num_lslr_gal_prev_lsl_replace[y] := 0;
            fed_num_lslr_leadcon_replace[y] := 0;
            fed_num_lslr_galprev_leadcon_replace[y] := 0;

            if num_lsl_replace_fed[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.pws_lsl] := LSL;
              NewDraw := True;
              num_lsl_remain := 0;

              proxy4_pws90 := 5;
            end;

            num_replace := 0;
            Num_lsl_replace_rule_allb := 0;
            for i := 1 to y do begin
              num_replace := num_replace + Num_lsl_replace_tot[i];
              Num_lsl_replace_rule_allb := Num_lsl_replace_rule_allb +
```

```
Num_lsl_replace_rule_all[i];
            end;

            SumTotLSLReplaceMand := SumTotLSLReplaceMand +
tot_num_lslr_lsl_replace[y];
            SumTotLSLPartialReplaceMand := SumTotLSLPartialReplaceMand +
tot_num_lslr_partial_replace[y];
            SumTotLSLGalPrevMand := SumTotLSLGalPrevMand +
tot_num_lslr_gal_prev_lsl_replace[y];
            SumTotLSLLeadConMand := SumTotLSLLeadConMand +
tot_num_lslr_leadcon_replace[y];
            SumTotLSLGalPrevLeadConMand := SumTotLSLGalPrevLeadConMand +
tot_num_lslr_galprev_leadcon_replace[y];

            SumTotLSLReplaceVol := 0;
            SumTotLSLPartialReplaceVol := 0;
            SumTotLSLGalPrevVol := 0;
            SumTotLSLLeadConVol := 0;
            SumTotLSLGalPrevLeadConVol := 0;

            SumFedLSLReplace := SumFedLSLReplace + Num_lsl_replace_fed[y];
            SumFedLSLReplaceMand := SumFedLSLReplaceMand +
fed_num_lslr_lsl_replace[y];
            SumFedLSLPartialReplaceMand := SumFedLSLPartialReplaceMand +
fed_num_lslr_partial_replace[y];
            SumFedLSLGalPrevMand := SumFedLSLGalPrevMand +
fed_num_lslr_gal_prev_lsl_replace[y];
            SumFedLSLLeadConMand := SumFedLSLLeadConMand +
fed_num_lslr_leadcon_replace[y];
            SumFedLSLGalPrevLeadConMand := SumFedLSLGalPrevLeadConMand +
fed_num_lslr_galprev_leadcon_replace[y];

            SumFedLSLReplaceVol := 0;
            SumFedLSLPartialReplaceVol := 0;
            SumFedLSLGalPrevVol := 0;
            SumFedLSLLeadConVol := 0;
            SumFedLSLGalPrevLeadConVol := 0;

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

            num_unknown_remain := max(0,Num_sl_unknown - num_unknown_resolved[y]);

            Variables[fI.num_lsl_replace] := num_lsl_replace_tot[y];
                  Variables[fI.num_lsl_testout] := num_lsl_testout[y];
            Variables[fI.hh_remain_lsl] := hh_remain_lsl;
            Variables[fI.num_lslr_lsl_replace] := fed_num_lslr_lsl_replace[y];
            Variables[fI.num_lslr_partial_replace] :=
fed_num_lslr_partial_replace[y];
          end; // year >= 4
```

```
        end // Config.BaselineName = 'LCR'
        else if Config.BaselineName = 'LCRR' then begin
          pp_lsl_replaced_vol_pct_yr[y] := max(pp_lsl_replaced_vol_pct_yr[y],
state_lslr_rate);
          pp_lsl_replaced_mand_pct := max(0.03, state_lslr_rate);
          pp_lsl_replaced_mand_pct_small := max(0.07, state_lslr_rate);

          Num_unknown_resolved[y] := min (0.10 * Num_sl_unknown,
Num_unknown_remain);
          Num_unknown_remain := max(0, Num_sl_unknown - num_unknown_resolved[y]);
            hh_unknown_remain := max(0, Num_sl_unknown - num_unknown_resolved[y]) *
num_hh_per_connect;
          Variables[fI.num_unknown_resolved] := num_unknown_resolved[y];

          if Num_lsl_remain > 0 then begin
            if (proxy3_pws90[y] > bp1) and (proxy3_pws90[y] <= bp2) then begin
                Num_lsl_replace_tot[y] := min(pp_lsl_replaced_vol_pct_yr[y] *
Num_potential_LSL_base,
                  num_lsl_remain);
                    Num_lsl_replace_state[y] := min(state_lslr_rate *
Num_Potential_LSL_base, Num_lsl_remain);
                Num_lsl_replace_fed[y] :=  Num_lsl_replace_tot[y] -
Num_lsl_replace_state[y];

              if num_lsl_replace_tot[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_tot[y] := min(pp_lsl_replaced_mand_pct *
(Num_Potential_LSL_base), Num_lsl_remain);
                      Num_lsl_replace_state[y] := min(state_lslr_rate *
Num_Potential_LSL_base, Num_lsl_remain);
                    Num_lsl_replace_fed[y] :=  Num_lsl_replace_tot[y] -
Num_lsl_replace_state[y];

                // NOTE in mandatory program unknowns are investigated at same rate
as the federal required rate lsls are replaced

                if num_lsl_replace_tot[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
```

```
              begin
                Num_lsl_replace_tot[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.pws_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_tot[i];
            num_requested := num_requested + Num_lsl_requested[i];
          end;

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

              Num_unknown_remain := max(0, Num_sl_unknown -
num_unknown_resolved[y]);

          Variables[fI.num_lsl_replace] := num_lsl_replace_tot[y];
          Variables[fI.hh_remain_lsl] := hh_remain_lsl;

          fed_num_lslr_lsl_replace[y] := num_lsl_replace_fed[y] *
            (Variables[fI.pp_lslr_lsl] / (Variables[fI.pp_lslr_lsl] +
Variables[fI.pp_lslr_partial] +
            Variables[fI.pp_lslr_gal_prev_lsl]));

          fed_num_lslr_partial_replace[y] := num_lsl_replace_fed[y] *
            (Variables[fI.pp_lslr_partial] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_gal_prev_lsl]));

          fed_num_lslr_gal_prev_lsl_replace[y] := num_lsl_replace_fed[y] *
            (Variables[fI.pp_lslr_gal_prev_lsl] / (Variables[fI.pp_lslr_lsl] +
```

```
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_gal_prev_lsl]));

          fed_num_lslr_leadcon_replace[y] := 0;
          fed_num_lslr_galprev_leadcon_replace[y] := 0;

          tot_num_lslr_lsl_replace[y] := num_lsl_replace_tot[y] *
            (Variables[fI.pp_lslr_lsl] / (Variables[fI.pp_lslr_lsl] +
Variables[fI.pp_lslr_partial] +
            Variables[fI.pp_lslr_gal_prev_lsl]));

          tot_num_lslr_partial_replace[y] := num_lsl_replace_tot[y] *
            (Variables[fI.pp_lslr_partial] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_gal_prev_lsl]));

          tot_num_lslr_gal_prev_lsl_replace[y] := num_lsl_replace_tot[y] *
            (Variables[fI.pp_lslr_gal_prev_lsl] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_gal_prev_lsl]));

          tot_num_lslr_leadcon_replace[y] := 0;
          tot_num_lslr_galprev_leadcon_replace[y] := 0;

          if proxy3_pws90[y] > bp2 then begin
            SumFedLSLReplaceMand := SumFedLSLReplaceMand +
fed_num_lslr_lsl_replace[y];
            SumFedLSLPartialReplaceMand := SumFedLSLPartialReplaceMand +
fed_num_lslr_partial_replace[y];
            SumFedLSLGalPrevMand := SumFedLSLGalPrevMand +
fed_num_lslr_gal_prev_lsl_replace[y];
            SumFedLSLLeadConMand := SumFedLSLLeadConMand +
fed_num_lslr_leadcon_replace[y];
            SumFedLSLGalPrevLeadConMand := SumFedLSLGalPrevLeadConMand +
fed_num_lslr_galprev_leadcon_replace[y];

            SumTotLSLReplaceMand := SumTotLSLReplaceMand +
tot_num_lslr_lsl_replace[y];
            SumTotLSLPartialReplaceMand := SumTotLSLPartialReplaceMand +
tot_num_lslr_partial_replace[y];
            SumTotLSLGalPrevMand := SumTotLSLGalPrevMand +
tot_num_lslr_gal_prev_lsl_replace[y];
            SumTotLSLLeadConMand := SumTotLSLLeadConMand +
tot_num_lslr_leadcon_replace[y];
            SumTotLSLGalPrevLeadConMand := SumTotLSLGalPrevLeadConMand +
tot_num_lslr_galprev_leadcon_replace[y];
          end
          else if (proxy3_pws90[y] > bp1) and (proxy3_pws90[y] <= bp2) then begin
            SumFedLSLReplaceVol := SumFedLSLReplaceVol +
fed_num_lslr_lsl_replace[y];
            SumFedLSLPartialReplaceVol := SumFedLSLPartialReplaceVol +
fed_num_lslr_partial_replace[y];
            SumFedLSLGalPrevVol := SumFedLSLGalPrevVol +
```

```
fed_num_lslr_gal_prev_lsl_replace[y];
            SumFedLSLLeadConVol := SumFedLSLLeadConVol +
fed_num_lslr_leadcon_replace[y];
            SumFedLSLGalPrevLeadConVol := SumFedLSLGalPrevLeadConVol +
fed_num_lslr_galprev_leadcon_replace[y];

              SumTotLSLReplaceVol := SumTotLSLReplaceVol +
tot_num_lslr_lsl_replace[y];
            SumTotLSLPartialReplaceVol := SumTotLSLPartialReplaceVol +
tot_num_lslr_partial_replace[y];
            SumTotLSLGalPrevVol := SumTotLSLGalPrevVol +
tot_num_lslr_gal_prev_lsl_replace[y];
            SumTotLSLLeadConVol := SumTotLSLLeadConVol +
tot_num_lslr_leadcon_replace[y];
            SumTotLSLGalPrevLeadConVol := SumTotLSLGalPrevLeadConVol +
tot_num_lslr_galprev_leadcon_replace[y];
          end;

          Variables[fI.num_lslr_lsl_replace] := fed_num_lslr_lsl_replace[y];
          Variables[fI.num_lslr_partial_replace] := fed_num_lslr_partial_replace[y];
          Variables[fI.num_lslr_gal_prev_lsl_replace] :=
fed_num_lslr_gal_prev_lsl_replace[y];
          Variables[fI.num_lslr_leadcon_replace] := fed_num_lslr_leadcon_replace[y];
          Variables[fI.num_lslr_galprev_leadcon_replace] :=
fed_num_lslr_galprev_leadcon_replace[y];

          // Failure to meet LSLR Voluntary Program in Bin 2
            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_tot[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
```

```
            else
            begin
              if NM >= 1 then
                Variables[fI.fail_nm1] := 1;
              if NM >= 2 then
                Variables[fI.fail_nm2] := 1;
            end;
        end; // Config.BaselineName = 'LCRR'
      end // if option = 'Baseline'
      else
      if option = 'LCRI' then begin
        // NOTE in LCRI unknowns are investigated at same rate as lsls are replaced

        num_unknown_resolved[y] := min(0.10 * Num_sl_unknown, Num_unknown_remain);
        Num_unknown_remain := max(0, Num_sl_unknown - num_unknown_resolved[y]);
          hh_unknown_remain := max(0, Num_sl_unknown - num_unknown_resolved[y]) *
num_hh_per_connect ;

        // Determines the number of lines that must be validated in year 8.
        // Number in validation pool = unknowns resolved that are not lead.
        // Only occurs one year (year 8)
        if y = 8 then begin
          Num_lsl_validate_pool[y] := (Num_sl_unknown - Num_unknown_remain) *
                                            (1 -
Variables[fI.perc_lsl_unknown_lead]);
          if Num_lsl_validate_pool[y] < 1500 then
            Num_lsl_validated[y] := 0.20 * Num_lsl_validate_pool[y]
          else
            num_lsl_validated[y] := (Num_lsl_validate_pool[y] * 384.16) /
              (Num_lsl_validate_pool[y] + 383.16);
        end;

        Variables[fI.num_unknown_remain] := Num_unknown_remain;
        Variables[fI.hh_unknown_remain] := hh_unknown_remain;
        Variables[fI.num_lsl_validated] := Num_lsl_validated[y];

        num_unknown_resolved[y] := min(0.10 * num_sl_unknown, num_unknown_remain);

        if y >= 4 then begin
          if Num_lsl_remain > 0 then begin
            Num_lsl_replace_tot[y] := min(lsl_replace_pct * num_potential_LSL_base,
Num_lsl_remain);
            Num_lsl_replace_state[y] := min(state_lslr_rate *
num_potential_LSL_base, Num_lsl_remain);
            Num_lsl_replace_fed[y] := Num_lsl_replace_tot[y] -
Num_lsl_replace_state[y];

            num_replace := num_replace + Num_lsl_replace_tot[y];
            num_lsl_remain := num_potential_LSL_Base - num_replace;
            hh_remain_lsl := max(0, (num_hh_per_connect * num_lsl_remain));
```

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

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

            // lower than any ALE
            ttpws90pct := bp2/2;

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

          Variables[fI.num_lsl_replace] := Num_lsl_replace_tot[y];
          Variables[fI.hh_remain_lsl] := hh_remain_lsl;

          if Config.LSLOption = 1 then begin
            Variables[fI.pp_lslr_leadcon] := 0;
            Variables[fI.pp_lslr_galprev_leadcon] := 0;
          end
          else if Config.LSLOption = 2 then
            Variables[fI.pp_lslr_galprev_leadcon] := 0;


          tot_num_lslr_lsl_replace[y] := num_lsl_replace_tot[y] *
            (Variables[fI.pp_lslr_lsl] / (Variables[fI.pp_lslr_lsl] +
Variables[fI.pp_lslr_partial]
            + Variables[fI.pp_lslr_leadcon] + Variables[fI.pp_lslr_gal_prev_lsl] +

Variables[fI.pp_lslr_galprev_leadcon]));

          tot_num_lslr_partial_replace[y] := num_lsl_replace_tot[y] *
            (Variables[fI.pp_lslr_partial] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_leadcon] +
            Variables[fI.pp_lslr_gal_prev_lsl] +
Variables[fI.pp_lslr_galprev_leadcon]));

          tot_num_lslr_gal_prev_lsl_replace[y] := num_lsl_replace_tot[y] *
            (Variables[fI.pp_lslr_gal_prev_lsl] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_leadcon] +
            Variables[fI.pp_lslr_gal_prev_lsl] +
```

```
Variables[fI.pp_lslr_galprev_leadcon]));

          tot_num_lslr_leadcon_replace[y] := num_lsl_replace_tot[y] *
            (Variables[fI.pp_lslr_leadcon] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_leadcon] +
            Variables[fI.pp_lslr_gal_prev_lsl] +
Variables[fI.pp_lslr_galprev_leadcon]));

          tot_num_lslr_galprev_leadcon_replace[y] := num_lsl_replace_tot[y] *
            (Variables[fI.pp_lslr_galprev_leadcon] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_leadcon] +
            Variables[fI.pp_lslr_gal_prev_lsl] +
Variables[fI.pp_lslr_galprev_leadcon]));

          fed_num_lslr_lsl_replace[y] := num_lsl_replace_fed[y] *
            (Variables[fI.pp_lslr_lsl] / (Variables[fI.pp_lslr_lsl] +
Variables[fI.pp_lslr_partial]
            + Variables[fI.pp_lslr_leadcon] + Variables[fI.pp_lslr_gal_prev_lsl] +

Variables[fI.pp_lslr_galprev_leadcon]));

          fed_num_lslr_partial_replace[y] := num_lsl_replace_fed[y] *
            (Variables[fI.pp_lslr_partial] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_leadcon] +
            Variables[fI.pp_lslr_gal_prev_lsl] +
Variables[fI.pp_lslr_galprev_leadcon]));

          fed_num_lslr_gal_prev_lsl_replace[y] := num_lsl_replace_fed[y] *
            (Variables[fI.pp_lslr_gal_prev_lsl] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_leadcon] +
            Variables[fI.pp_lslr_gal_prev_lsl] +
Variables[fI.pp_lslr_galprev_leadcon]));

          fed_num_lslr_leadcon_replace[y] := num_lsl_replace_fed[y] *
            (Variables[fI.pp_lslr_leadcon] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_leadcon] +
            Variables[fI.pp_lslr_gal_prev_lsl] +
Variables[fI.pp_lslr_galprev_leadcon]));

          fed_num_lslr_galprev_leadcon_replace[y] := num_lsl_replace_fed[y] *
            (Variables[fI.pp_lslr_galprev_leadcon] / (Variables[fI.pp_lslr_lsl] +
            Variables[fI.pp_lslr_partial] + Variables[fI.pp_lslr_leadcon] +
            Variables[fI.pp_lslr_gal_prev_lsl] +
Variables[fI.pp_lslr_galprev_leadcon]));

          SumTotLSLReplaceMand := SumTotLSLReplaceMand +
tot_num_lslr_lsl_replace[y];
          SumTotLSLPartialReplaceMand := SumTotLSLPartialReplaceMand +
tot_num_lslr_partial_replace[y];
          SumTotLSLGalPrevMand := SumTotLSLGalPrevMand +
```

```
tot_num_lslr_gal_prev_lsl_replace[y];
          SumTotLSLLeadConMand := SumTotLSLLeadConMand +
tot_num_lslr_leadcon_replace[y];
          SumTotLSLGalPrevLeadConMand := SumTotLSLGalPrevLeadConMand +
tot_num_lslr_galprev_leadcon_replace[y];

          SumTotLSLReplaceVol := 0;
          SumTotLSLPartialReplaceVol := 0;
          SumTotLSLGalPrevVol := 0;
          SumTotLSLLeadConVol := 0;
          SumTotLSLGalPrevLeadConVol := 0;

          SumFedLSLReplaceMand := SumFedLSLReplaceMand +
fed_num_lslr_lsl_replace[y];
          SumFedLSLPartialReplaceMand := SumFedLSLPartialReplaceMand +
fed_num_lslr_partial_replace[y];
          SumFedLSLGalPrevMand := SumFedLSLGalPrevMand +
fed_num_lslr_gal_prev_lsl_replace[y];
          SumFedLSLLeadConMand := SumFedLSLLeadConMand +
fed_num_lslr_leadcon_replace[y];
          SumFedLSLGalPrevLeadConMand := SumFedLSLGalPrevLeadConMand +
fed_num_lslr_galprev_leadcon_replace[y];

          SumFedLSLReplaceVol := 0;
          SumFedLSLPartialReplaceVol := 0;
          SumFedLSLGalPrevVol := 0;
          SumFedLSLLeadConVol := 0;
          SumFedLSLGalPrevLeadConVol := 0;

          Variables[fI.num_lslr_lsl_replace] := fed_num_lslr_lsl_replace[y];
          Variables[fI.num_lslr_partial_replace] := fed_num_lslr_partial_replace[y];
          Variables[fI.num_lslr_gal_prev_lsl_replace] :=
fed_num_lslr_gal_prev_lsl_replace[y];
          Variables[fI.num_lslr_leadcon_replace] := fed_num_lslr_leadcon_replace[y];
          Variables[fI.num_lslr_galprev_leadcon_replace] :=
fed_num_lslr_galprev_leadcon_replace[y];

          // calculate number of households that get filters if Filters for
LSL/Unknowns is turned on
          // in interface (lsl_filter = 1)

          if Config.LSLFilters = 1 then
          begin
            if aType = 1 then num_lsl_filters := hh_remain_lsl + hh_unknown_remain
            else num_lsl_filters := (num_lsl_remain + num_unknown_remain) *
Variables[fI.numb_ntncws_filters];
          end
          else
            num_lsl_filters := 0;
```

```
          // A. Calculate the number of households requiring temporary POU in year
and correct for
          // double counting between Filters for LSL/Unknowns and temp POU
          num_temp_pou := 0;

          if Config.ALE = 15 then
            P90_concurrent := Variables[fI.p_90_concurrent_15]
          else if Config.ALE = 12 then
            P90_concurrent := Variables[fI.p_90_concurrent_12]
          else if Config.ALE = 10 then
            P90_concurrent := Variables[fI.p_90_concurrent_10]
          else if Config.ALE = 5 then
            P90_concurrent := Variables[fI.p_90_concurrent_5];

          if Config.TempPOUOption = 1 then begin
            Variables[fI.b_temp_pou_2] := 0;
            Variables[fI.b_temp_pou_3] := 0;
            Variables[fI.b_temp_pou_4] := 0;

            if (proxy2_pws90_y4 > bp2) and (cct_change_yr > y) and (POU = 0) and
(P90_concurrent = 1) then
            begin
              Variables[fI.b_temp_pou_1] := 1;
              num_temp_pou := aNC - num_lsl_filters;
            end
            else
            begin
              Variables[fI.b_temp_pou_1] := 0;
            end;
          end
          else if Config.TempPOUOption = 2 then begin
            Variables[fI.b_temp_pou_1] := 0;
            Variables[fI.b_temp_pou_3] := 0;
            Variables[fI.b_temp_pou_4] := 0;

            if (proxy2_pws90_y4 > bp2) and (cct_change_yr > y) and (POU = 0) and
(P90_concurrent = 1) then
            begin
              Variables[fI.b_temp_pou_2] := 1;
              num_temp_pou := hh_remain_lsl + hh_unknown_remain - num_lsl_filters;
            end
            else
            begin
              Variables[fI.b_temp_pou_2] := 0;
            end;
          end
          else if Config.TempPOUOption = 3 then begin
            Variables[fI.b_temp_pou_1] := 0;
            Variables[fI.b_temp_pou_2] := 0;
            Variables[fI.b_temp_pou_4] := 0;
```

```
            if (proxy2_pws90_y4 > bp2) and (cct_change_yr > y) and (POU = 0) and
(P90_concurrent = 1) then
            begin
              Variables[fI.b_temp_pou_3] := 1;
              num_temp_pou := (hh_remain_lsl + hh_unknown_remain - num_lsl_filters)
* Variables[fI.pp_pou_adopt];
            end
            else
            begin
              Variables[fI.b_temp_pou_3] := 0;
            end;
          end
          else if Config.TempPOUOption = 4 then begin
            Variables[fI.b_temp_pou_1] := 0;
            Variables[fI.b_temp_pou_2] := 0;
            Variables[fI.b_temp_pou_3] := 0;
            Variables[fI.b_temp_pou_4] := 1;
          end;

          Variables[fI.num_temp_pou] := num_temp_pou;
          Variables[fI.num_lsl_filter] := num_lsl_filters;
        end;  // y = 4
      end; // end option = LCRI
    end; // end has LSL

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

    // NOTELCRI check on num_lsl_requested
    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;

    // find and fix
    if ((option = 'Baseline') and (Config.BaselineName = 'LCRR')) and (y >= 1) 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=1) or (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 = 1) or (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_dssa] := 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
    else if (option = 'LCRI') and (y >= 4) then
    begin
      Num_tap_ge_al := 0;

      // bin = 3
      if pws90pct <= bp2 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],
            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],
            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],
            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,
            pp_above_al_bin_three);
      end
      else
        Num_tap_ge_al :=
fCostVars.Calculate_Num_tap_ge_al(Variables[fI.numb_samp_customer] * 2,
          pp_above_al_bin_one);

      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_dssa] := 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];
      end;
    end;
```

```
    if (option = 'Baseline') and (Config.BaselineName = 'LCRR') then begin
      if y >= 1 then begin
        if ff_pws90pct > 0 then
          pws90pct := min(proxy4_pws90, ff_pws90pct)
        else
          pws90pct := proxy4_pws90;
      end;
    end
    else if option = 'LCRI' 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_dssa_cost]:=0;
      Variables[fI.cct_dssa_cost_umra]:=0;
      Variables[fI.cct_dssa_cost_umra_om]:=0;
      Variables[fI.cct_dssa_cost_p]:=0;

    FindAndFixCostCap2 := 0;
      if fnf and (hFF_CCT = 2) then begin
      Variables[fI.cct_dssa_cost] := (Variables[fI.hrs_act_wqp_op] *
Variables[fI.rate_op]) +
        Variables[fI.cost_act_wqp];
      Variables[fI.cct_dssa_cost_umra] := (Variables[fI.hrs_act_wqp_op] *
Variables[fI.rate_op])
        + Variables[fI.cost_act_wqp];
        Variables[fI.cct_dssa_cost_umra_om] := 0;
      Variables[fI.cct_dssa_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_dssa_cost] := (FindAndFixCostOM + FindAndFixCostCapDisc -
AdjustCCTCostOM)
        * (1 / aEP);
      if (y - hffY2) MOD UsefulLifeFF = 0 then begin
        Variables[fI.cct_dssa_cost_umra] := (FindAndFixCostCap) * (1 / aEP);
        FindAndFixCostCap2 := (FindAndFixCostCap) * (1 / aEP);
      end
        else
          Variables[fI.cct_dssa_cost_umra] := 0;
        Variables[fI.cct_dssa_cost_umra_om] := (FindAndFixCostOM - AdjustCCTCostOM)
* (1/aEP);
      Variables[fI.cct_dssa_cost_p] :=
        (FindAndFixCostOM + FindAndFixCostCapDisc_p - AdjustCCTCostOM) * (1 / aEP);
    end
    else if (CCTB = 0) and (hff_cct = 3) then begin
      Variables[fI.cct_dssa_cost] :=
        (FindAndFixCostOM + FindAndFixCostCapDisc - InstallCCTCostOM) * (1 / aEP);
      if (y - hffY2) MOD UsefulLifeFF = 0 then begin
        Variables[fI.cct_dssa_cost_umra] := (FindAndFixCostCap) * (1 / aEP);
        FindAndFixCostCap2 := (FindAndFixCostCap) * (1 / aEP);
      end
        else
          Variables[fI.cct_dssa_cost_umra] := 0;
        Variables[fI.cct_dssa_cost_umra_om] := (FindAndFixCostOM - InstallCCTCostOM)
* (1/aEP);
      Variables[fI.cct_dssa_cost_p] :=
        (FindAndFixCostOM + FindAndFixCostCapDisc_p - InstallCCTCostOM) * (1 / aEP);
    end
    else if (CCTB = 1) and (hff_cct >= 4) then begin
```

```
      Variables[fI.cct_dssa_cost] :=
        (FindAndFixCostOM + FindAndFixCostCapDisc - AdjustCCTCostOM);
      if (y - hFFY3) MOD UsefulLifeFF = 0 then begin
        Variables[fI.cct_dssa_cost_umra] := (FindAndFixCostCap);
        FindAndFixCostCap2 := FindAndFixCostCap;
      end
        else
          Variables[fI.cct_dssa_cost_umra] := 0;

        Variables[fI.cct_dssa_cost_umra_om] := (FindAndFixCostOM - AdjustCCTCostOM);
        Variables[fI.cct_dssa_cost_p] := (FindAndFixCostOM + FindAndFixCostCapDisc_p
- AdjustCCTCostOM);
      end else if (CCTB = 0) and (hFF_CCT >= 4) then begin
        Variables[fI.cct_dssa_cost] := (FindAndFixCostOM + FindAndFixCostCapDisc -
InstallCCTCostOM);
        if (Y-hFFY3) MOD UsefulLIfeFF = 0 then
          Variables[fI.cct_dssa_cost_umra] := (FindAndFixCostCap)
        else
          Variables[fI.cct_dssa_cost_umra] := 0;
        Variables[fI.cct_dssa_cost_umra_om] := (FindAndFixCostOM -
InstallCCTCostOM);
        Variables[fI.cct_dssa_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 (option = 'Baseline') and (Config.BaselineName = 'LCRR') then begin
      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;
    end else begin
      if pws90pct > bp2 then
        owBin_tmp := 1
      else
        owBin_tmp := 3;
    end;

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

```
    else
      owBin := owBin;

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

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

{
  Dollar year adjustments from Jerry email 7/28/23
  From  To      Multiplier
  2021  2022     1.069
  2020  2022     1.095
  2021  2020     0.977

  For costs from database, including state costs, use 1.095
}

      if _UseCompiled then begin
        Cost := CC._Cost[C.fCostStepRec.ID];
        Cost := Cost * 1.095;
        Labor := CC._Labor[C.fCostStepRec.ID];
        Labor := Labor * 1.095;
        OM := CC._OM[C.fCostStepRec.ID];
        OM := OM * 1.095;
        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 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') and (Config.BaselineName = 'LCR') 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 = 'Baseline') and (Config.BaselineName = 'LCRR') 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
          else if (option = 'LCRI') 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;

          TotalPWSCost := TotalPWSCost + GetTotalPWSCost(acTotalPWSCost,
```

```
c.fCostStepRec, Cost);

            if AddCostingData.Num_Proxies = 0 then
            begin
              if not LLL.Exists('PWSCost_' + Config.RunName) then
                LLL.L('PWSCost_' +
Config.RunName,'PWSId,SystemSize,Ownership,SourceWater,SystemType,y,TotalPWSCost,Cap
italCost,Weight');

              lll.L('PWSCost_' + Config.RunName, CostingData.PWSid + ',' +
CostingData.SystemSize.ToString + ',' +
                    CostingData.Ownership.ToString + ',' +
CostingData.SourceWater.ToString + ',' +
                    CostingData.SystemType.ToString + ',' + y.ToString + ',' +
TotalPWSCost.ToString + ',' +
                    (AdjustCCTCostCap + InstallCCTCostCap +
FindAndFixCostCap).ToString + ',' + CostingData.SamplingWeight.ToString);
            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;

      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) 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;
    end;  // end C in CostSteps loop
    // end Compute Costs

    // 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*aEP) * PDose * 10893.71;
          ConnectionLossP := aNC * PDose * 0.86;
        // 1.184 GDP inflator from 2016 to 2020
```

```
        POTWCost := POTWCost + ((Discount((FlowLossP - ConnectionLossP), y - 1,
CostingData.CostCapital)) * 1.184);

      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
    if NewDraw = true then
      fCostVars.FillValueArray(Variables, RawVariables, aSz, aSrc, LSL, CCT, aType,
Y+1, Config.PWS90PctBp1,
                               Config.PWS90PctBp2, SetProbsTo01, NewDraw, O,
isBaseline);

    Variables[fI.pws_lsl] := LSL;

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

    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_dssa] := 1;

    if ((option = 'Baseline') and (Config.BaselineName = 'LCRR')) or (option =
'LCRI') 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);

    num_lsl_replace_y := num_lsl_replace_fed[y];
    num_unknown_resolved_y := num_unknown_resolved[y];
    SetFH;
    LeadConcentrationBins(CostingData.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, CostingData.SamplingWeight,
num_lsl_replace_fed[y], num_lsl_requested[y], num_lsl_remain,
                                                perc_lsl_b,
AddCostingData.Num_Proxies, partial_cct_level,
                          Variables[fI.pp_lslr_lsl], pp_lslr_lsl_adj,
num_lsl_filters, num_hh_per_connect, num_temp_pou,
                          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
```

```
  SumTotLSLReplace := SumTotLSLReplaceMand + SumTotLSLReplaceVol +
                      SumTotLSLPartialReplaceMand + SumTotLSLPartialReplaceVol +
                      SumTotLSLGalPrevMand + SumTotLSLGalPrevVol +
                      SumTotLSLLeadConMand + SumTotLSLLeadConVol +
                      SumTotLSLGalPrevLeadConMand + SumTotLSLGalPrevLeadConVol;

  SumFedLSLReplace := SumFedLSLReplaceMand + SumFedLSLReplaceVol +
                         SumFedLSLPartialReplaceMand + SumFedLSLPartialReplaceVol +
                         SumFedLSLGalPrevMand + SumFedLSLGalPrevVol +
                         SumFedLSLLeadConMand + SumFedLSLLeadConVol +
                         SumFedLSLGalPrevLeadConMand + SumFedLSLGalPrevLeadConVol;
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;

  hrs_adopt_rule_js: double;
  hrs_modify_ds_js: double;
  hrs_initial_ta_js: double;
  hrs_train_imp_js: double;
  hrs_coord_epa_js: double;
  hrs_ta_js: double;
  hrs_sdwis_js: double;
  hrs_train_ann_js: double;
  rate_js: 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);

  hrs_adopt_rule_js := 0;
  hrs_modify_ds_js := 0;
  hrs_initial_ta_js := 0;
  hrs_train_imp_js := 0;

  hrs_coord_epa_js := Variables[fI.hrs_coord_epa_js];
  hrs_ta_js := Variables[fI.hrs_ta_js];
  hrs_sdwis_js := Variables[fI.hrs_sdwis_js];
  hrs_train_ann_js := Variables[fI.hrs_train_ann_js];
  rate_js := Variables[fI.rate_js];

  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;

      StateCostUndiscounted := 0;

      if c.fCostStepRec.CostName = 'state_adopt' then begin
        StateCost := StateCost + Discount(hrs_adopt_rule_js*rate_js, y - 1, -2);
        StateCostUndiscounted := StateCostUndiscounted + hrs_adopt_rule_js*rate_js;
      end
      else if c.fCostStepRec.CostName = 'state_modify_ds' then begin
        StateCost := StateCost + Discount(hrs_modify_ds_js*rate_js, y - 1, -2);
        StateCostUndiscounted := StateCostUndiscounted + hrs_modify_ds_js*rate_js;
      end
      else if c.fCostStepRec.CostName = 'state_provide_train' then begin
        StateCost := StateCost + Discount(hrs_initial_ta_js*rate_js, y - 1, -2);
        StateCostUndiscounted := StateCostUndiscounted + hrs_initial_ta_js*rate_js;
      end
      else if c.fCostStepRec.CostName = 'state_attend_train' then begin
        StateCost := StateCost + Discount(hrs_train_imp_js*rate_js, y - 1, -2);
        StateCostUndiscounted := StateCostUndiscounted + hrs_train_imp_js*rate_js;
      end
      else if c.fCostStepRec.CostName = 'state_epa' then begin
        StateCost := StateCost + Discount(hrs_coord_epa_js*rate_js, y - 1, -2);
        StateCostUndiscounted := StateCostUndiscounted + hrs_coord_epa_js*rate_js;
      end
      else if c.fCostStepRec.CostName = 'state_ta' then begin
        StateCost := StateCost + Discount(hrs_ta_js*rate_js, y - 1, -2);
        StateCostUndiscounted := StateCostUndiscounted + hrs_ta_js*rate_js;
```

```
      end
      else if c.fCostStepRec.CostName = 'state_sdwis' then begin
        StateCost := StateCost + Discount(hrs_sdwis_js*rate_js, y - 1, -2);
        StateCostUndiscounted := StateCostUndiscounted + hrs_sdwis_js*rate_js;
      end
      else if c.fCostStepRec.CostName = 'state_train_admin' then begin
        StateCost := StateCost + Discount(hrs_train_ann_js*rate_js, y - 1, -2);
        StateCostUndiscounted := StateCostUndiscounted + hrs_train_ann_js*rate_js;
      end;

      //StateCost := StateCost + Discount(Cost, y - 1, -2);
    end;

    if not LLL.Exists('StateCost_' + Config.RunName) then
      LLL.L('StateCost_' + Config.RunName,'y,StateCostUndiscounted');
    LLL.L('StateCost_' + Config.RunName, y.ToString + ',' +
StateCostUndiscounted.ToString);
  end;

{
  Dollar year adjustments from Jerry email 7/28/23
  From  To      Multiplier
  2021  2022     1.069
  2020  2022     1.095
  2021  2020     0.977

  For costs from database, including state costs, use 1.095
}
  StateCost := StateCost * 1.095;

  // 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;

  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_install_cct_lead_ale') 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 := '';
  fOKAll := True;
  for i := 1 to high(farrCalculateYr) do
    ArrCalculateYr[i] := false;

  if IsBaseline  then
    fRandomStream := rrBase
  else
    fRandomStream := rrScen;
  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));
        rYr := iiRandomRange(iYr, jYr, fRandomStream);
        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);
            k := iiRandomRange(1, j, fRandomStream);
            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, fRandomStream);
    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, fRandomStream);
    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.
```