# APPENDIX A –

# Model Run Code

[from basecase run called “SimBase_update_aug2009”]

Files included are:

main.ocl
udef_list.ocl
irrig_calculation.ocl
irrig_allocation.ocl
set_inflows.ocl
return_flows.ocl
routing.ocl
crabtree.ocl
upper_eno_ops.ocl
durham_ops.ocl
falls_bdam_ops.ocl
falls_flood_ops.ocl
raleigh_ops.ocl
buckhorn_ops.ocl
Falls_Bdam_WS_WQ_Accounts.ocl

## main.ocl

```
:INCLUDE: OCL\constants.ocl
:INCLUDE: OCL\Forecast-Trigger_Parms.ocl

:Static: statdata.mdb
:Time:   [HomeDir]\basedata\basedata.dss

:if: {[UseForecast]=1}
  :if: {[ForecastData]=cond}
    :Time:   [HomeDir]\basedata\forecasts_cond.dss
  :else:
    :Time:   [HomeDir]\basedata\forecasts_non_cond.dss
  :endif:
:endif:


:MODULE: DLL  IrrigDem = modules\IrrigDem.DLL

:Include: ocl\udef_list.ocl

:Commands:

:Include: ocl\irrig_calculation.ocl
:Include: ocl\irrig_allocation.ocl
:Include: ocl\set_inflows.ocl
:Include: ocl\routing.ocl
:Include: ocl\return_flows.ocl
:Include: ocl\crabtree.ocl
:Include: ocl\upper_eno_ops.ocl
:Include: ocl\durham_ops.ocl
:Include: ocl\falls_bdam_ops.ocl
:Include: ocl\falls_flood_ops.ocl
:Include: ocl\raleigh_ops.ocl
:Include: ocl\buckhorn_ops.ocl

/*  Set demands not used in basecase run to zero */
Set :  demand152  {  value :  0 } /* Durham-OWASA Interconnect */
Set :  demand154  {  value :  0 } /* Durham-Chatham Interconnect */
Set :  demand156  {  value :  0 } /* Durham-Cary Interconnect */
Set :  demand158  {  value :  0 } /* Durham-Raleigh Interconnect */
Set :  demand318  {  value :  0 } /* Old Burlington Industries */
Set :  demand702  {  value :  0 } /* Falls-Cary */
Set :  demand704  {  value :  0 } /* Falls-Holly Springs */
Set :  demand706  {  value :  0 } /* Falls-Fuquay-Varina*/

/* Set Creedmor WD from L. Rogers to zero - currently buying water from SGWASA */
Constraint : {dFlow270.258 = 0}

/*  Set inflows not used in basecase run to zero*/
```

```
Set : inflow252  {  value :  0 } /* Kerr Lake indirect transfer*/
Set : inflow403  {  value :  0 } /* Quarry dewatering */
Set : inflow528  {  value :  0 } /* Tar River transfer */


/*  Solve command for the current time step. */
Solve : {Priority : 1}


/*  Compute the Falls/Beaverdam water quality and supply accounting after the SOLVE
statement */

:Include:   ocl\falls_bdam_wq_ws_accounts.ocl


:End:
```

## Udef_list.ocl

```
:Udef:

/* Udefs for inflow filtering */
:For:
{   [node] = {010, 050, 060, 080, 100, 110, 115, 250, 270, 290, 420, 440, 445, 450, 480, 500, 560,
630, 740, 750, 780, 800, 900}
}
   Udef : _TempInf[node]
   Udef : _InfDeficit[node]  init{0}
:Next:

/* Udefs for Crabtree filtering */
Udef : _TempInfCrabtree
Udef : _TotInfCrabtree
Udef : _InfDeficitCrabtree  init{0}

/*  Udefs For the agricultural demand computations */
Udef : dem01
Udef : dem02
Udef : dem03
Udef : dem04
Udef : dem05
Udef : dem06
Udef : dem07
Udef : dem08
Udef : dem09
Udef : dem10
Udef : dem11
Udef : dem12
Udef : dem13
Udef : dem14

/*  Udefs for cumulative tracking variables */
Udef :  _CumWSupplyDemand   init {0}
Udef :  _CumWSupplyDelivery init {0}
Udef :  _CumWWReturn        init {0}
Udef :  _CumWSupplyWWReturn init {0}

/*  Udefs for Eno operations */
Udef :  _Orange_streamflow  init {3.376} /* Initially set to 1.1 mgd*/
Udef :  _Eno_Channel_Loss   init {0}
Udef :  _Max_Hills_WD       init {4.634} /* Initially set to 1.51 mgd */
Udef :  _Max_OrAl_WD        init {2.157} /* Initially set to 0.82 mgd */
Udef :  _Max_Pied_WD        init {1.320} /* Initially set to 0.43 mgd */
Udef :  _WFER_Tier          init {1}

/*  For proportional drawdown */
Udef : _StorRatio   Decision { 0 , unbounded }
```

Udef : _StorRatio2  Decision { 0 , unbounded }

/*  Udef for Local Clayton Inflow for Falls Flood control */
Udef : _ClaytLocal init {0}

/*  Udefs for Falls/Clayton minimum flow */
Udef : ClaytonTarget_Override
Udef : ClaytonTarget_SafeFact
Udef : FLake_WQ_Threshold
Udef : FLakeMin_SafeFact

/*  Udefs for Raleigh system storage pct. - only used when Swift
   Creek WD being used. */
UDef : _Raleigh_Sys_Stor_Pct

/*  Udefs for the Falls and Beaverdam pool allocation */
Udef :  _Pct_Max_WQ_Storage
Udef :  _Pct_Max_WS_Storage
Udef :  _FLake_Max_WQ_Storage
Udef :  _FLake_Max_WS_Storage
Udef :  _Bdam_Max_WQ_Storage
Udef :  _Bdam_Max_WS_Storage
Udef :  Total_Max_WQ_Storage
Udef :  Total_Max_WQ_WS_Storage
Udef :  FLake_Max_WQ_WS_Storage
Udef :  _WQ_Stor_fraction
Udef :  _WS_Stor_fraction
Udef :  _Bdam_Inf
Udef :  _Bdam_Inf_WS
Udef :  _Bdam_WS_Rel
Udef :  _Bdam_WS_Stor_Init
Udef :  _Bdam_WStoWQ_Spill
Udef :  _Bdam_Inf_WQ
Udef :  _Bdam_WQ_Rel
Udef :  _Bdam_WQ_Stor      init {[Bdam_WQ_Stor]  }
Udef :  _Bdam_WQtoWS_Spill
Udef :  _Bdam_WS_Stor      init {[Bdam_WS_Stor]  }
Udef :  _FLake_Inf
Udef :  _FLake_Inf_WS
Udef :  _Flake_WS_Rel
Udef :  _FLake_WS_Stor_Init
Udef :  _FLake_WStoWQ_Spill
Udef :  _FLake_Inf_WQ
Udef :  _Flake_WQ_Rel
Udef :  _FLake_WQ_Stor      init {[FLake_WQ_Stor] }
Udef :  _Total_WQ_Stor
Udef :  _FLake_WQtoWS_Spill
Udef :  _FLake_WS_Stor      init {[FLake_WS_Stor] }
Udef :  _Total_WS_Stor
Udef :  _Bdam_WS_Pct
Udef :  _Bdam_WQ_Pct

Udef :  _FLake_WS_Pct
Udef :  _FLake_WQ_Pct
Udef :  _Total_WS_Pct
Udef :  _Total_WQ_Pct

:substitute: [InflowNd] = "010, 050, 060, 080, 100, 110, 115, 140, 200, 230, 250, 270, 290, 300, 420,
440, 445, 450, 480, 500, 560, 630, 740, 750, 780, 800, 900"

:substitute: [ResNd] = "010, 050, 060, 080, 100, 120, 140, 200, 230, 250, 290, 300, 420,
440, 445, 450, 500, 740"

:substitute: [CrabtreeNd] = "400, 402, 404, 406, 408, 410, 412, 414, 416, 418, 422"

/* For municipal and industrial demand */
:substitute: [DemandNd] = "046, 106, 116, 162, 256, 258, 296, 306, 506, 646, 666, 766, 786, 906"

/*  For WW returns (arcs) associated with water supply withdrawals */
:substitute: [WWRetArc] = "046.100, 106.115, 162.300, 256.300, 258.300, 306.600, 306.500, 296.310, 306.500, 646.630, 675.700, 766.770, 786.780, 506.560, 906.999"

/*  For WW returns (nodes) not associated with water supply withdrawals */
:substitute: [WWRetNd] = "307, 401, 405, 407, 409, 563, 565, 567, 643, 645, 705, 747, 753, 757, 759, 787, 845, 847"

/*  For Ag demand (nodes)  */
:substitute: [IrrigNd] = "062, 112, 202, 142, 302, 632, 482, 742, 782, 802, 502, 562, 902"

Udef :  _WtAvgRunoff

## Irrig_Calculation.ocl

```
/*  Note the precip data is contained in the basedata file and is just one record
using the precip record for Falls Lake.  This should be fairly representative of the basin
as it is in the middle.  For more precision, use as county precip the precip for reservoirs
throughout the basin. */

:For:
{   [cty] = {01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14}
}

   RUN_MODULE: IrrigDem
   {
     Input:  { [cty],                    /* County number */
             timesers(neuse/precip),   /* Precip in inches */
             pattern(IrrCoef_Tobacco),    /* Water Use Coefficient for Tobacco */
             pattern(IrrCoef_Turf),       /* Water Use Coefficient for Turf */
             pattern(IrrCoef_Golf),       /* Water Use Coefficient for Golf */
             pattern(IrrCoef_ContNurs),   /* Water Use Coefficient for Cont (continuously irrigated)
Nurseries */
             pattern(IrrCoef_FieldNurs),  /* Water Use Coefficient for Field Nurseries */
             pattern(IrrCoef_Cotton),     /* Water Use Coefficient for Cotton */
             pattern(IrrCoef_EarlySoy),   /* Water Use Coefficient for Early Soy */
             pattern(IrrCoef_LateSoy),    /* Water Use Coefficient for Late Soy */
             pattern(IrrCoef_Corn),       /* Water Use Coefficient for Corn */
             pattern(IrrCoef_Veg),        /* Water Use Coefficient for Vegetables */
             pattern(IrrCoef_PastHay),    /* Water Use Coefficient for Pasture Hay */
             pattern(IrrCoef_Peanut),     /* Water Use Coefficient for Peanuts */
             pattern(IrrCoef_Blueberry),  /* Water Use Coefficient for Blueberries */
             pattern(IrrCoef_Strawberry), /* Water Use Coefficient for Strawberries */
             pattern(IrrCoef_Fruit),      /* Water Use Coefficient for Fruit */
             pattern(IrrCoef_Beef),       /* Water Use Coefficient for Beef Cattle */
             pattern(IrrCoef_Dairy),      /* Water Use Coefficient for Dairy Cattle */
             pattern(IrrCoef_Horse),      /* Water Use Coefficient for Horses */
             pattern(IrrCoef_Pig),       /* Water Use Coefficient for Pigs */
             pattern(IrrCoef_Chicken),    /* Water Use Coefficient for Chickens */
             pattern(IrrCoef_Turkey),     /* Water Use Coefficient for Turkeys */
             pattern(IrrCoef_OtherAnimal) /* Water Use Coefficient for Other Animals */
           }

     Output: { dem[cty] }
   }
:Next:

/*  The results are in mgd.  Now convert these to acre feet for use in the
irrigation_allocation ocl file  */

Set  :  dem01  {  value  :  convert_units {dem01, mgd, af }  }
Set  :  dem02  {  value  :  convert_units {dem02, mgd, af }  }
```

```
Set : dem03 { value : convert_units {dem03, mgd, af } }
Set : dem04 { value : convert_units {dem04, mgd, af } }
Set : dem05 { value : convert_units {dem05, mgd, af } }
Set : dem06 { value : convert_units {dem06, mgd, af } }
Set : dem07 { value : convert_units {dem07, mgd, af } }
Set : dem08 { value : convert_units {dem08, mgd, af } }
Set : dem09 { value : convert_units {dem09, mgd, af } }
Set : dem10 { value : convert_units {dem10, mgd, af } }
Set : dem11 { value : convert_units {dem11, mgd, af } }
Set : dem12 { value : convert_units {dem12, mgd, af } }
Set : dem13 { value : convert_units {dem13, mgd, af } }
Set : dem14 { value : convert_units {dem14, mgd, af } }
```

# Irrig_allocation.ocl

/* This file allocates the irrigation demands by the assumed distribution of demand within each reach of interest */

/* One portion of the county demand is set to the "dem__" that varies by number. These numbers are established in the irrigation input dialog box, with 01 set for Craven, 02 for Durham, and so on. (Also see irrig_calculation.ocl file and irrigation dialog box) */

```
Set Demand_Or_Pond_Ag : demand062 { value : dem09 * 0.095 }
Set Demand_WFER_Ag : demand052 { value : dem09 * 0.095 }
Set Demand_Michie_Ag : demand142 { value : dem02 * 0.078 + dem10 * 0.31 }
Set Demand_LitRes_Ag : demand202 { value : dem09 * 0.03 + dem02 * 0.097 + dem09
* 0.14 }

Set Demand_Falls_Ag : demand302 { value : dem02 * 0.076 + dem04 * 0.02 + dem02 *
0.488 + dem10 * 0.01 + dem04 * 0.07
                                + dem02 * 0.094 + dem12 * 0.05 + dem03 * 0.02 + dem04 *
0.15 }
Set Demand_Clayt_Ag : demand632 { value : dem06 * 0.04 + dem02 * 0.04 + dem12 *
0.20 + dem03 * 0.01 + dem12 * 0.035 }
Set Demand_Middl_Ag : demand482 { value : dem06 * 0.01 + dem12 * 0.158 }
Set Demand_Litpr_Ag : demand752 { value : dem14 * 0.02 + dem06 * 0.18 + dem12 *
0.158 + dem03 * 0.03 }

Set Demand_Golds_Ag : demand782 { value : dem06 * 0.10 + dem12 * 0.053 + dem06
* 0.08 + dem13 * 0.42 + dem06 * 0.39 + dem06 * 0.13 }
Set Demand_Kinst_Ag : demand802 { value : dem07 * 0.35 + dem05 * 0.05 }
Set Demand_Buckhorn_Ag : demand502 { value : dem14 * 0.05 + dem06 * 0.04 + dem08
* 0.13 + dem12 * 0.035 + dem03 * 0.05 }
Set Demand_Hooke_Ag : demand562 { value : dem13 * 0.23 + dem05 * 0.57 + dem14 *
0.70 + dem08 * 0.07 + dem06 * 0.01 }

Set Demand_Weyer_Ag : demand902 { value : dem01 * 0.04 + dem07 * 0.39 + dem07 *
0.09 + dem05 * 0.39 + dem11 * 0.17 + dem14 * 0.04 + dem11 * 0.23 }
Paint
```

## Set_inflows.ocl

```
/* Sets the inflows for nodes that need to be filtered. The finalized inflows
   (through April 2008) were already filtered, however the provisional inflows
   from the update record routine can be negative due to time of travel or
   imperfect impairment estimations, and therefore are filtered here to prevent
   model infeasibility or unrealistic reservoir releases and/or demand shortages.
   All inflows are filtered except Falls/Bdam, which can have negatives due to
   the use of back-calculated inflows; and Michie/LRR in Durham, which are based
   on already unimpaired gages that will not be negative.  The inflows to the
   Crabtree Creek system are set and filtered separately in crabtree.ocl */

:For:
{   [node] = {010, 050, 060, 080, 100, 110, 115, 250, 270, 290, 420, 440, 445, 450, 480, 500, 560,
630, 740, 750, 780, 800, 900}
}
    Set : _TempInf[node] { Value : timesers([node]/inflow) }
    Set : inflow[node] { Value : max{0, _TempInf[node] - _InfDeficit[node](-1) } }
    Set : _InfDeficit[node] { Value : max{0, _InfDeficit[node](-1) - _TempInf[node] } }

:Next:
```

## return_flows.ocl

```
/*  File is RETURN_FLOWS.OCL, which has the coding to handle the return flows
    from demand node. */

/*  Hillsborough return flows */
Constraint : {dFlow106.115 = lookup {HillsReturn, month} * dDelivery106}

/*  Durham return flows */
Constraint : {dFlow162.300 = lookup {DurhamReturn, month} * dDelivery162}

/*  SGWASA return flows - Note: Creedmor WW is sent to SGWASA */
Constraint : {dFlow259.300 = lookup {SGWASAReturn, month} * (dDelivery256 +
ddelivery258)}

/*  Raleigh Neuse WWTP return flows  */
Constraint : {dFlow306.620 = lookup {RaleighReturn, month} * dDelivery306}

/* Zebulon - Little Creek return flows - is a percentage of Raleigh demand */
Constraint : {dFlow306.500 = lookup {LitCrReturn, month} * dDelivery306}

/*  Wake-Forest Smith Creek return flows - based on WF estimated as 5% of Raleigh's total
demand  */
Constraint : {dFlow306.320 =  lookup {WakeFReturn, month} * 0.05 * dDelivery306}

/*  Cary return flows - divide between NRF and SRF */
Constraint : {dFlow401.416 = 0.55 * inflow401}

Constraint : {dFlow401.480 = 0.45 * inflow401}

/*  Clayton return flows - Clayton demand is inlcuded in Johnst Co. */
Constraint : {dFlow646.630 = lookup {ClaytReturn, month} * dDelivery646}

/*  Johnston return flows - also treats Smithfield WW */
Constraint : {dFlow675.700 = lookup {JohnstReturn, month} * (dDelivery646 + dDelivery666)}

/*  Progress Energy return flows - generally all WD is used for operations */
Constraint : {dFlow766.770 = 0.0 * dDelivery766}

/*  Goldsboro return flows */
Constraint : {dFlow786.780 = lookup {GoldsReturn, month} * dDelivery786}

/*  NRWASA return flows */
Constraint : {dFlow806.850 = lookup {NRWASAReturn, month} * dDelivery806}

/*  Wilson return flows */
Constraint : {dFlow506.560 = lookup {WilsonReturn, month} * dDelivery506}

/*  Weyerhauser return flows */
Constraint : {dFlow906.999 = lookup {WeyerReturn, month} * dDelivery906}
```

## routing.ocl

```
/*  File is ROUTING.OCL, which has the coding to handle the Falls - Clayton,
    Clayton - Goldsboro and Goldsboro - Kinston routing, daily time step only. */

/*  Falls-to-Clayton.
       Falls Release seen at Clayton
       = .5 * yesterday's release + .5 * today's release.

    Clayton-to-Goldsboro.
       Clayton flow seen at Goldsboro
       = Average flow of previous four days

    Goldsboro-to-Kinston.
       Goldsboro flow seen at Kinston
       = Average flow of previous two days

    Nodes 310, 640 and 790 are reservoirs used for channel storage.
    The flow into these three nodes is unrouted; the release from them is routed.
    The storage in the nodes makes up the difference.

    Note that for the first day of the simulation the routed flows need to be
    estimated; the current values set the routed flows equal to the upstream flow
    for that day when the simulation is started on 01/01/1929; they may need to be
    adjusted if the run is started on a different date. */

Target FallsRout : dflow310.320 - .5 * dflow300.310
{   Condition : abs_period <= 1
    priority  : 1
    penalty+  : 10000
    penalty-  : 10000
    value     : convert_units { 49.25, cfs, af }

    Condition : default
    priority  : 1
    penalty+  : 10000
    penalty-  : 10000
    value     : .5 * flow300.310(-1)
}
Target ClaytRout : dflow640.650
{   Condition : abs_period <= 1
    priority  : 1
    penalty+  : 10000
    penalty-  : 10000
    value     : convert_units { 353.6, cfs, af }

    Condition : abs_period = 2
    priority  : 1
    penalty+  : 10000
```

```
   penalty-  : 10000
   value     : flow630.640(-1)

   Condition : abs_period = 3
   priority  : 1
   penalty+  : 10000
   penalty-  : 10000
   value     : ( flow630.640(-2) + flow630.640(-1) ) / 2

   Condition : abs_period = 4
   priority  : 1
   penalty+  : 10000
   penalty-  : 10000
   value     : ( flow630.640(-3) + flow630.640(-2) + flow630.640(-1) ) / 3

   Condition : default
   priority  : 1
   penalty+  : 10000
   penalty-  : 10000
   value     : ( flow630.640(-4) + flow630.640(-3) + flow630.640(-2) + flow630.640(-1) ) / 4
}
Target GoldsRout : dflow790.795
{   Condition : abs_period <= 1
   priority  : 1
   penalty+  : 10000
   penalty-  : 10000
   value     : convert_units { 1047.6, cfs, af }

   Condition : abs_period = 2
   priority  : 1
   penalty+  : 10000
   penalty-  : 10000
   value     : flow780.790(-1)

   Condition : default
   priority  : 1
   penalty+  : 10000
   penalty-  : 10000
   value     : ( flow780.790(-2) + flow780.790(-1) ) / 2
}
```

## crabtree.ocl

```
/* Set the Crabtree Creek inflows and net evap for the individual flood control impoundments
  Instead of having 11 inflow and evap records in the basedata file, base all of them
  the inflow400/evap400 from the main basedata.dss */


/* Variables for the incremental DA into each reservoir */
:Substitute: [400] = 2.1
:Substitute: [402] = 1.4
:Substitute: [404] = 2.3
:Substitute: [406] = 8.9
:Substitute: [408] = 5.2
:Substitute: [410] = 8.2
:Substitute: [412] = 2.5
:Substitute: [414] = 11.5
:Substitute: [416] = 3.7
:Substitute: [418] = 23.1
:Substitute: [422] = 24.8
:Substitute: [CrabtreeTotal] = 93.7

/* Filter the total inflow (in case there are any negative provisional inflows) */
Set : _TempInfCrabtree { Value : timesers(400/inflow) }
Set : _TotInfCrabtree { Value : max{0, _TempInfCrabtree - _InfDeficitCrabtree(-1) } }
Set : _InfDeficitCrabtree { Value : max{0, _InfDeficitCrabtree(-1) - _TempInfCrabtree } }

/* Set the inflows - proportion by drainage area the total inflow to Crabtree Creek */
Set : inflow400  { value :  _TotInfCrabtree / [CrabtreeTotal] * [400]  }
Set : inflow402  { value :  _TotInfCrabtree / [CrabtreeTotal] * [402]  }
Set : inflow404  { value :  _TotInfCrabtree / [CrabtreeTotal] * [404]  }
Set : inflow406  { value :  _TotInfCrabtree / [CrabtreeTotal] * [406]  }
Set : inflow408  { value :  _TotInfCrabtree / [CrabtreeTotal] * [408]  }
Set : inflow410  { value :  _TotInfCrabtree / [CrabtreeTotal] * [410]  }
Set : inflow412  { value :  _TotInfCrabtree / [CrabtreeTotal] * [412]  }
Set : inflow414  { value :  _TotInfCrabtree / [CrabtreeTotal] * [414]  }
Set : inflow416  { value :  _TotInfCrabtree / [CrabtreeTotal] * [416]  }
Set : inflow418  { value :  _TotInfCrabtree / [CrabtreeTotal] * [418]  }
Set : inflow422  { value :  _TotInfCrabtree / [CrabtreeTotal] * [422]  }

/* Set the evap for all nodes */
:For:
{   [node] = { 400, 402, 404, 406, 408, 410, 412, 414, 416, 418, 422 }
}
   Set : evap[node]  { value : stor_to_area { [node], storage[node] } * timesers(400/evap) / 12 }

:Next:

/* Set the spillway rating curves for the Crabtree Creek impoundments.  Prevent releases from
drawing pond below spill elevation  */
Target 400_release: dflow400.412
```

{ Condition : default
priority : 1
penalty+ : 1000
penalty- : 1000
Value : min {convert_units { Lookup{ 400Outflow, elevation400 }, cfs, af}, storage400 - elev_to_stor {400, 305} }
}

Target 402_release: dflow402.412
{ Condition : default
priority : 1
penalty+ : 1000
penalty- : 1000
Value : min {convert_units { Lookup{ 402Outflow, elevation402 }, cfs, af}, storage402 - elev_to_stor {402, 326} }
}

Target 404_release: dflow404.412
{ Condition : default
priority : 1
penalty+ : 1000
penalty- : 1000
Value : min {convert_units { Lookup{ 404Outflow, elevation404 }, cfs, af}, storage404 - elev_to_stor {404, 342.5} }
}

Target 406_release: dflow406.412
{ Condition : default
priority : 1
penalty+ : 1000
penalty- : 1000
Value : min {convert_units { Lookup{ 406Outflow, elevation406 }, cfs, af}, storage406 - elev_to_stor {406, 306.5} }
}

Target 408_release: dflow408.412
{ Condition : default
priority : 1
penalty+ : 1000
penalty- : 1000
Value : min {convert_units { Lookup{ 408Outflow, elevation408 }, cfs, af}, storage408 - elev_to_stor {408, 313} }
}

Target 410_release: dflow410.412
{ Condition : default
priority : 1
penalty+ : 1000
penalty- : 1000
Value : min {convert_units { Lookup{ 410Outflow, elevation410 }, cfs, af}, storage410 - elev_to_stor {410, 306.5} }

}

Target 412_release: dflow412.416
{ Condition : default
priority : 1
penalty+ : 1000
penalty- : 1000
Value : min {convert_units { Lookup{ 412Outflow, elevation412 }, cfs, af}, storage412 - elev_to_stor {412, 276} }
}

Target 414_release: dflow414.416
{ Condition : default
priority : 1
penalty+ : 1000
penalty- : 1000
Value : min {convert_units { Lookup{ 414Outflow, elevation414 }, cfs, af}, storage414 - elev_to_stor {414, 301 } }
}

Target 416_release: dflow416.424
{ Condition : default
priority : 1
penalty+ : 1000
penalty- : 1000
Value : min {convert_units { Lookup{ 416Outflow, elevation416 }, cfs, af}, storage416 - elev_to_stor {416, 230} }
}

Target 418_release: dflow418.424
{ Condition : default
priority : 1
penalty+ : 1000
penalty- : 1000
Value : min {convert_units { Lookup{ 418Outflow, elevation418 }, cfs, af}, storage418 - elev_to_stor {418, 335} }
}

Target 422_release: dflow422.424
{ Condition : default
priority : 1
penalty+ : 1000
penalty- : 1000
Value : min {convert_units { Lookup{ 422Outflow, elevation422 }, cfs, af}, storage422 - elev_to_stor {422, 252} }
}

## upper_eno_ops.ocl

```
/* Operation policy the Upper Eno, based on the Capacity Use Plan */

/*  Compute channel loss factor, which is used in the Lake Orange and WFER min release
computations,
   and to calculate channel loss at the Hillsborough Gage.  Based on the values in the WFER dam
safety permit.  */
Set  :  _Eno_Channel_Loss
{   condition : flow110.115 < convert_units{4,cfs,af}
   Value     : 0.2

   condition : flow110.115 <= convert_units{12,cfs,af}
   Value     : 0.1

   condition : default
   Value     : 0
}

/*  Determine the maximum release from Lake Orange for Hillsborough demand; based on stage.
*/
Set :  _Max_Hills_WD
{      /* Stages 6 */
      condition : storage060 / max_stor060 <= 0.3
      Value     : convert_units { 0.68, mgd, af }

      /* Stages 5 */
      condition : storage060 / max_stor060 <= 0.4
      Value     : convert_units { 1.13, mgd, af }

      /* Stages 3 & 4 */
      condition : storage060 / max_stor060  <= 0.6
      Value     : convert_units { 1.28, mgd, af }

      /* Stage 2 */
      condition : storage060 / max_stor060 <= 0.8
      Value     : convert_units { 1.36, mgd, af }
      /* Stage 1 */
      condition : default
      Value     : convert_units { 1.51, mgd, af }
}

/*  Determine the maximum release from Lake Orange for Orange-Alamance demand; based on
stage. */
Set :  _Max_OrAl_WD
{      /* Stages 6 */
      condition : storage060 / max_stor060 <= 0.3
      Value     : convert_units { 0.37, mgd, af }

      /* Stages 5 */
```

```
      condition : storage060 / max_stor060 <= 0.4
      Value     : convert_units { 0.62, mgd, af }

      /* Stages 3 & 4 */
      condition : storage060 / max_stor060  <= 0.6
      Value     : convert_units { 0.70, mgd, af }

      /* Stage 2 */
      condition : storage060 / max_stor060 <= 0.8
      Value     : convert_units { 0.74, mgd, af }
      /* Stage 1 */
      condition : default
      Value     : convert_units { 0.82, mgd, af }
}

/*  Determine the maximum release from Lake Orange for Piedmont Minerals demand; based on
stage. */
Set :  _Max_Pied_WD
{      /* Stages 6 */
      condition : storage060 / max_stor060 <= 0.3
      Value     : convert_units { 0.00, mgd, af }

      /* Stages 5 */
      condition : storage060 / max_stor060 <= 0.4
      Value     : convert_units { 0.19, mgd, af }

      /* Stages 4 */
      condition : storage060 / max_stor060  <= 0.5
      Value     : convert_units { 0.32, mgd, af }

      /* Stages 3 */
      condition : storage060 / max_stor060  <= 0.6
      Value     : convert_units { 0.36, mgd, af }

      /* Stage 2 */
      condition : storage060 / max_stor060 <= 0.8
      Value     : convert_units { 0.38, mgd, af }
      /* Stage 1 */
      condition : default
      Value     : convert_units { 0.43, mgd, af }
}

/*  Determine the Lake Orange required streamflow augmentation; based on stage. */
Set :  _Orange_streamflow
{      /* Stages 6 */
      condition : storage060 / max_stor060 <= 0.3
      Value     : 0

      /* Stages 5 */
      condition : storage060 / max_stor060 <= 0.4
      Value     : 0
```

```
      /* Stages 3 & 4 */
      condition : storage060 / max_stor060  <= 0.6
      Value     : convert_units { 0.45, mgd, af }

      /* Stage 2 */
      condition : storage060 / max_stor060 <= 0.8
      Value     : convert_units { 0.65, mgd, af }
      /* Stage 1 */
      condition : default
      Value     : convert_units { 1.10, mgd, af }
}

/*  Lake Orange minimum release, based on Eno Capacity Use plan.  Includes a streamflow
augmentation release plus
   a release for downstream demands.  The release for Hillsborough, Or-Al and Pied. Minerals'
demand is only up to
   the maximum specified in the plan. A multiplier for channel loss, the same as for WFER, is
also included.  */
Set :  min_flow060.080
{      /* All stage-dependant streamflow augmentation and max. WDs have been determined
above  */
      condition : default
      Value     : (1 + _Eno_Channel_Loss) * (_Orange_streamflow + min { demand046,
_Max_OrAl_WD } + min { demand106, _Max_Hills_WD } + min { demand116,
_Max_Pied_WD })
}

/*  Determine which Phase WFER is in.  Set to Tier 3 if elevation < 624.  Set to Tier 2 if
elevation is between 624 and 628 ft.
          Otherwise set to Tier 1 */

Set : _WFER_Tier
{    Condition : elevation050 < 624
     Value     : 3

           Condition : elevation050 < 628
     Value     : 2

     Condition : default
     Value     : 1
}

/*  WFER Minimm release - depends on storage tier, and remaining Hillsborough demand that
cannot be met by Lake Orange.
   From Eno Capacity Use pland and WFER dam safety permit; either enough to meet
downstream requirements, or the specified
   seasonal pattern, whichever is greater  */
Set :  min_flow050.080
{    /* If in Tier1 */
     condition : _WFER_Tier = 1
```

```
    Value     : max { (1 + _Eno_Channel_Loss) * (convert_units{1.0,cfs,af} + max { 0,
demand106 - _Max_Hills_WD }) , convert_units{pattern(Tier1),cfs,af} }

    /* If in Tier2 */
    condition : _WFER_Tier = 2
    Value     : max { (1 + _Eno_Channel_Loss) * (convert_units{1.0,cfs,af} + max { 0,
demand106 - _Max_Hills_WD }) , convert_units{pattern(Tier2),cfs,af} }

    /* If in Tier3 */
    condition : _WFER_Tier = 3
    Value     : max { (1 + _Eno_Channel_Loss) * (convert_units{1.0,cfs,af} + max { 0,
demand106 - _Max_Hills_WD }) , convert_units{pattern(Tier3),cfs,af} }
}

/*  Compute channel loss at Hills. Gage (computed) */
Target Channel_Loss  :  dflow110.107
{   condition : default
   priority  : 1
   penalty+  : 1000
   penalty-  : 1000
   Value     : max { 0, _Eno_Channel_Loss * (flow105.110 + inflow110) }
}
```

## durham_ops.ocl

```
/*  Operational policy for Durham's system */

/*  Set Teer Quarry flows to zero - operating policy can be set in alternative scenarios */
Constraint : {dFlow115.120 = 0 }

Constraint : {dFlow140.120 = 0 }

Constraint : {dFlow200.120 = 0 }

Constraint : {dFlow120.151 = 0 }

/*  Turn off interconnects from Durham for base case - transfers can be simulated
    with an operating policy in alternative scenarios */
Constraint : {dFlow151.106 = 0 }

Constraint : {dFlow151.152 = 0 }

Constraint : {dFlow151.154 = 0 }

Constraint : {dFlow151.156 = 0 }

Constraint : {dFlow151.158 = 0 }

/*  Sets the Little River Reservoir (Durham) minimum release - based on stage and time of year */
Set LittleRiverRelease : min_flow200.205
{   condition : (storage200 / max_stor200) < 0.7
    value     : convert_units {0.64,cfs,af}

    condition : month > 5 and month < 12
    value     : convert_units {2.0,cfs,af}

    condition : default
    value     : convert_units {6.0,cfs,af}
}

/*  RESERVOIR BALANCING -- Durham
    With a very low penalty in Priority 1, balance the water (usable stor pct) between the
    reservoirs. This will draw Michie and Little River Res. down proportionally.
    This can be replaced with a more comprehensive operating policy from Durham
    in alternative scenarios */

Constraint Balance200 : { d_StorRatio > ( Dstorage200 - dead_stor200 ) / (max_stor200 -
dead_stor200 ) }

Constraint Balance140 : { d_StorRatio > ( Dstorage140 - dead_stor140 ) / (max_stor140 -
dead_stor140 ) }

Minimax : d_StorRatio
```

```
{   priority  : 1
    penalty   : 2
    tolerance : .02
}
```

## falls_bdam_ops.ocl

```
/* Operational policies for Falls and Beaverdam */

/*  Sets the Falls minimum release - seasonal pattern (100/60 cfs) with user-defined
safety factor (only when using service gates)
   The safety factor can be set in the ocl constants table. */
Set FallsRelease : min_flow300.310
{   condition : month > 3 and month < 11
   value     : convert_units { 100, cfs, af } * ( 1 + [FLakeMin_SafeFact] / 100 )

   condition : default
   /*  Winter min. release based on amount from Piggyback gates, which is approx. 60 cfs
depending on stage. */
   value     : convert_units { lookup { PiggybackGate, elevation300 }, cfs, af }
}

/*  Sets the Clayton minimum flow target - seasonal pattern (254/184 cfs) with user-
defined safety factor
   Also can be overridden if user chooses, when WQ storage drops below 20%; this can
be truned on in the ocl constants table. */
Set Clayt_Target: min_flow630.640
{   condition : [ClaytonTarget_Override] = 1 and ( _Total_WQ_Pct(-1) <
[FLake_WQ_Threshold] or elevation300 <= 245 )
   value     : 0

   condition : month > 3 and month < 11
   value     : convert_units { 254, cfs, af } * ( 1 + [ClaytonTarget_SafeFact] / 100 )

   condition : default
   value     : convert_units { 184, cfs, af } * ( 1 + [ClaytonTarget_SafeFact] / 100 )
}

/*  Beaverdam elevation = projected Falls elevation if at or above 249 ft.  */
Target Bdam_elevation : dstorage230
{   condition : elevation300 >= 249.0
         priority  : 1
   penalty+  : 10000
   penalty-  : 10000
   Value     : elev_to_stor {230, elevation300}
}

/*Beaverdam drought release policy*/

Target Beaverdam_release : dflow230.300
```

{  /* If Falls is between 237 and 240, and Beaverdam is above 236.5, draw down Beaverdam maximum of 1-ft (down to a minimum of 236.5); next drawdown can occur after 5 days */
  condition : elevation230(-5) - elevation230 < 5 and elevation300 < 240 and elevation230 > elevation300 and elevation230> 236.5
  priority  : 1
  penalty+  : 1000
  penalty-  : 1000
  Value     : min { elev_to_stor { 230, elevation230 } - elev_to_stor { 230, elevation230 - 1 }, elev_to_stor { 230, elevation230 } - elev_to_stor { 230, 236.5 } }

  /* If Falls is below 249 ft, make sure Beaverdam is not above 249 ft */
  condition : elevation230 > 249 and elevation300 < 249
  priority  : 1
  penalty+  : 1000
  penalty-  : 1000
  Value     : elev_to_stor { 230, elevation230 } - elev_to_stor { 230, 249 }

 /*  To ensure accurate accounting of water supply and quality pools, when stressed, release some contents from Beaverdam if either yesterday's water supply or
  water quality account for Falls was less than 5% full; do not draw down more than 1 ft per day  THIS IS NOT PART OF TERRY'S SPREADSHEET (sn) */
  condition :  _FLake_WS_Pct(-1) < 5 or _FLake_WQ_Pct(-1) < 5 and elevation230 > elevation300 + 0.9 and elevation230(-5) - elevation230 < 5
  priority  : 1
  penalty+  : 1000
  penalty-  : 1000
  Value     : max {0, min {elev_to_stor { 230, elevation230 } - elev_to_stor { 230, elevation230 - 1 } , ( _Bdam_WS_Stor(-1) + _Bdam_WQ_Stor(-1) + inflow230 - evap230 )} }

  condition : default
  priority  : 1
  penalty+  : 1000
  penalty-  : 1000
  Value     : 0
}

/*  Constrain the delivery for Raleigh from Falls based on amount in the total (Falls+Bdam) water supply storage account.
  Demand will not be fulfilled if the account has gone empty and there is not enough inflow to the water supply account (for the previous day).
  Similarly, constrain the water quality release from Falls (arc 300.310) based on amount in the water quality storage account.

Since we refer to yesterday's storage estimates, for the first day of the run, constrain deliveries and releases to the initialized volumes for
each account (for simplicity, ignore inflows) so that deliveries and releases are made.
*/

```
Constraint   Deliv_Lim_Raleigh :
        {  condition  :  abs_period = 1
           expression :  dflow300.306 <= [FLake_WS_Stor] + [Bdam_WS_Stor]
        }

Constraint   Deliv_Lim_Raleigh :
        {  condition  :  abs_period > 1
           expression :  dflow300.306 <= _Total_WS_Stor(-1)
        }

Constraint   Release_Limit_Falls :
        {  condition  :  abs_period = 1
           expression :  dflow300.310 <= [FLake_WQ_Stor] + [Bdam_WQ_Stor]
        }

Constraint   Release_Limit_Falls :
        {  condition  :  abs_period > 1
           expression :  dflow300.310 <= _Total_WQ_Stor(-1)
        }
```

## falls_flood_ops.ocl

```
/* Flood release policy for Falls Lake */

/*  Calculate estimated local inflow to Clayton, which is used in determing Falls flood releases */
Set : _ClaytLocal
{        Value : inflow290 + inflow445 + inflow450 + inflow630 +
        /*  To increase accuracy, estimate Crabtree outflows by using starting elevation and spillway curve */
        min {convert_units { Lookup{ 416Outflow, elevation416 }, cfs, af}, storage416 - elev_to_stor {416, 230} } +

        min {convert_units { Lookup{ 418Outflow, elevation418 }, cfs, af}, storage418 - elev_to_stor {418, 335} } +

        min {convert_units { Lookup{ 422Outflow, elevation422 }, cfs, af}, storage422 - elev_to_stor {422, 252} }
}

/*  Falls flood operations per recommendations in Terry Brown's email 02/26/2009 */
Target Falls_release  :  dflow300.310
{   /*  Release only the minimum release if elevation is less than 255 and est. Clayton inflows >= 7000 cfs */
   condition : elevation300 >= 251.5 and elevation300 <= 255 and _ClaytLocal >= convert_units { 7000, cfs, af}
   priority  : 1
   penalty+  : 50
   penalty-  : 50
   Value     : min_flow300.310

   /*  If elev. < 255 and est. Clayton inflow < 7000 cfs, release the lesser of 4000 cfs or (7000 - the Clayt inflow).
       Also added provision to only drop Falls to the guide curve, but not less than the minimum release */
   condition : elevation300 >= 251.5 and elevation300 <= 255 and _ClaytLocal < convert_units { 7000, cfs, af}
   priority  : 1
   penalty+  : 50
   penalty-  : 50
   Value     : max { min_flow300.310, min { convert_units { 4000, cfs, af }, convert_units { 7000, cfs, af } - _ClaytLocal, storage300 - elev_to_stor { 300, 251.5 } } }

   /*  Release only the minimum release if elevation is less than 255 - 258 and est. Clayton inflows >= 7000 cfs */
```

```
   condition : elevation300 > 255 and elevation300 <= 258 and _ClaytLocal >=
convert_units { 7000, cfs, af}
   priority  : 1
   penalty+  : 50
   penalty-  : 50
   Value     : min_flow300.310

   /*  If elev. is 255 - 258 and est. Clayton inflow < 7000 cfs, release 7000 - the Clayt
inflow. */
   condition : elevation300 > 255 and elevation300 <= 258 and _ClaytLocal <
convert_units { 7000, cfs, af}
   priority  : 1
   penalty+  : 50
   penalty-  : 50
   Value     : convert_units { 7000, cfs, af } - _ClaytLocal

   /*  Release only the minimum release if elevation is 258 - 264.8 and est. Clayton
inflows >= 8000 cfs */
   condition : elevation300 > 258 and elevation300 <= 264.8 and _ClaytLocal >=
convert_units { 8000, cfs, af}
   priority  : 1
   penalty+  : 50
   penalty-  : 50
   Value     : min_flow300.310

   /*   If elev. is 258 - 264.8 and est. Clayton inflow < 8000 cfs, release 8000 - the Clayt
inflow. */
   condition : elevation300 > 258 and elevation300 <= 264.8 and _ClaytLocal <
convert_units { 8000, cfs, af}
   priority  : 1
   penalty+  : 50
   penalty-  : 50
   Value     : convert_units { 8000, cfs, af } - _ClaytLocal

   /*  If elev. is 264.8 - 268 and est. Clayton inflow >= 8000 cfs, release only the
computed spillway amount */
   condition : elevation300 > 264.8 and elevation300 <= 268 and _ClaytLocal >=
convert_units { 8000, cfs, af}
   priority  : 1
   penalty+  : 50
   penalty-  : 50
   Value     : convert_units { lookup { FallsSpillway, elevation300 }, cfs, af }

   /*  If elev. is 264.8 - 268 and est. Clayton inflow < 8000 cfs, release [(8000 cfs + the
computed spillway amount) - the Clayt inflow]. */
```

```
   condition : elevation300 > 264.8 and elevation300 <= 268 and _ClaytLocal <
convert_units { 8000, cfs, af}
   priority  : 1
   penalty+  : 50
   penalty-  : 50
   Value     : convert_units {  8000 + lookup { FallsSpillway, elevation300 }, cfs, af } -
_ClaytLocal

   /*  If elev. is > 268, release the full amounts possible from the service gates and the
spillway. */
   condition : elevation300 > 268
   priority  : 1
   penalty+  : 50
   penalty-  : 50
   Value     : convert_units { lookup { FallsSvcGate, elevation300 } + lookup {
FallsSpillway, elevation300 }, cfs, af }
}

/*  Set targets based on flood stages at downstream gages; note that the above release
policy takes priorty over these targets */

/*  Clayton flood control target */
Target Clayton_max  :  dflow630.640
{   condition : default
   priority  : 1
   penalty+  : 10
   penalty-  : 0
   Value     : convert_units{ 5332, cfs, af }
}

/*  Goldsboro flood control target  */
Target Goldsboro_max  :  dflow780.790
{   condition : default
   priority  : 1
   penalty+  : 10
   penalty-  : 0
   Value     : convert_units{ 6003, cfs, af }
}

/*  Kinston flood control target */
Target Kinston_max  :  dflow800.850
{   condition : default
   priority  : 1
   penalty+  : 10
   penalty-  : 0
   Value     : convert_units{ 7168, cfs, af }}
```

## raleigh_ops.ocl

```
/*  Operational policy for other Raleigh systems - Swift Creek and the proposed Little
River Reservoir. */

/*  Set the minimum release for Benson/Wheeler - per 3/3/08 e-mail from Mary Sadler
via Reed Palmer - based on useable storage */
Set :  min_flow440.700
{   condition : (storage420 - dead_stor420 + storage440 -  dead_stor440) / (max_stor420 -
dead_stor420 + max_stor440 - dead_stor440) < 0.3
    Value     : convert_units { 1 , cfs , af }

    condition : (storage420 - dead_stor420 + storage440 - dead_stor440) / (max_stor420 -
dead_stor420 + max_stor440 - dead_stor440) < 0.6
    Value     : convert_units { 2 , cfs , af }

    condition : default
    Value     : convert_units { 3 , cfs , af }
}

/*  Set flow from Lake Benson to Raleigh demand to zero; basecase simulates Raleigh's
withdrawal from Falls;
    this section can be commented out and the next section dictating the Swift Creek
operations policiy can
    be activated for alternative runs. */
Constraint : {dFlow440.306 = 0 }

/*  Operational policy for withdrawals from Benson as proposed by Hazen & Sawyer.
    This code is commented out for basecase run; it can be activated for an
    alternative run where the new WTP is being used. */

/* System Storage for Raleigh - needs to be adjusted if one of the supply systems is
turned on or off */
Set : _Raleigh_Sys_Stor_Pct
{  value      :   (_Total_WS_Stor(-1) + storage420 - dead_stor420 + storage440 -
dead_stor440 ) /
                ( [Total_Max_WS_Storage] + max_stor420 - dead_stor420 + max_stor440 -
dead_stor440)  }

/* Set a delivery constraint on Benson to make it maximize usage when storage gets low
so
   that the full useable storage is utilized. */
/*Constraint   Maximize_Lake_Benson_withdrawal :
{   condition  :  storage440 >= dead_stor440 + convert_units { 20 , mgd , af } and
                  _Raleigh_Sys_Stor_Pct < 0.10
          expression :  dflow440.306 = convert_units { 20 , mgd , af }
```

```
}*/

/* Set minimum delivery targets for Benson & Falls to improve withdrawal stability */
/*Target Benton_WTP_Min_production  :  dflow440.306
{   condition : default
    priority  : 1
    penalty+  : 0
    penalty-  : 50
    Value     : convert_units{ 5, mgd, af }
}

Target Falls_WTP_Min_production  :  dflow300.306
{   condition : default
    priority  : 1
    penalty+  : 0
    penalty-  : 50
    Value     : convert_units{ 45, mgd, af }
}*/

/* Set the maximum delivery from Benton to Raleigh based on the Benton WTP capacity
*/
/*Set :  max_flow440.306
{   condition  :  default
          Value      :  convert_units { 20, mgd , af }
} */

/*  RESERVOIR BALANCING -- Raleigh - Attempt to draw down the Falls/Bdam WS
account and the usable storage
    in Benson/Wheeler proportionally. */

/*Constraint Balance300 : { d_StorRatio2 > ( Dstorage300 + Dstorage230 - elev_to_stor
{ 300 , 236.5 } - elev_to_stor { 230, 236.5 } - _Total_WQ_Stor(-1))
                              / ( (elev_to_stor { 300 , 251.5 } - elev_to_stor { 300 , 236.5 })
+
                                (elev_to_stor { 230 , 251.5 } - elev_to_stor { 230 , 236.5 }) -
[Total_Max_WQ_Storage] ) + 0 }

Constraint Balance440 : { d_StorRatio2 > ( Dstorage440 + Dstorage420 - dead_stor440 -
dead_stor420 ) / (max_stor440 + max_stor420 - dead_stor440 - dead_stor420) + 0.1 }


Minimax : d_StorRatio2

{  priority  : 1
   penalty   : 25
   tolerance : .02
```

} */

/*  Raleigh's proposed Little River Reservoir --
    Set flow from Little River Reservoir (Raleigh) to Raleigh demand to zero;
    basecase simulates Raleigh's withdrawal from Falls; this
    can be changed to reflect a future operating policy using that lake. */
Constraint : {dFlow740.306 = 0 }

/*  Also note that the net evaporation for the propsed reservoir is turned off in the
    basecase run, to prevent drawdown in the reservoir from cutting off flows in the
    river when modeling current scenarios with the reservoir.  The net evaporation
    can be turned back on for an alternative scenario by switching the Evaporation
    Type to "Time Series" under the Net Evaporation table. */

## buckhorn_ops.ocl

```
/*  Minimum release policy for Buckhorn Lake - based on storage remaining. */
Set :  min_flow500.520
{   condition : storage500 / max_stor500 < 0.5
    Value     : convert_units { 1.4 , cfs , af }

    condition : storage500 / max_stor500 < 0.7
    Value     : convert_units { 5.3 , cfs , af }

    condition : default
    Value     : convert_units { 7.6 , cfs , af }
}
```

## Falls_Bdam_WQ_WS_Accounts.ocl

```
/* The Falls conservation storage zone is divided into water quality and water storage
accounts
Fraction of conservation storage and inflow :
WQ = 57.7% allocated (61,322 acft.)
WS = 42.3% allocated (45,000 acft.)

Beaverdam is also included in the WQ/WS accounting, the proportion to each is the same
as Falls. */


/*  WQ and WS accounts by lake */
Set :  _Pct_Max_WQ_Storage  {  value  :  ( [Total_Max_WQ_Storage] /
[Total_Max_WQ_WS_Storage] ) * 100 }
Set :  _Pct_Max_WS_Storage  {  value  :  100 - _Pct_Max_WQ_Storage }

Set :  _FLake_Max_WQ_Storage {  value :  _Pct_Max_WQ_Storage / 100 *
[FLake_Max_WQ_WS_Storage] }
Set :  _FLake_Max_WS_Storage {  value :  _Pct_Max_WS_Storage / 100 *
[FLake_Max_WQ_WS_Storage] }
Set :  _Bdam_Max_WQ_Storage  {  value  : [Total_Max_WQ_Storage] -
_FLake_Max_WQ_Storage }
Set :  _Bdam_Max_WS_Storage  {  value  : [Total_Max_WS_Storage] -
_FLake_Max_WS_Storage }


/*  Beaverdam Accounting */

Set  :  _Bdam_Inf  {  value :  inflow230 - evap230(0) }  /* Base it on net inflow for
Beaverdam.

                    Evap (0) since this file is included after the solve statement and we want
today's evap, not tomorrow's */

Set  :  _Bdam_Inf_WS   {  value :  _Pct_Max_WS_Storage / 100 * _Bdam_Inf }

Set  :  _Bdam_WS_Rel /* Calculate the WS outflow to Falls from Beaverdam.  Assume
that the WS amount is simply based on the percent of water supply to total account
storage in Beaverdam */
{     condition  :  elevation230 < 249 /* WS Outflow only if Beaverdam is less than 249ft
(spillage is accounted for in Falls) */
      value      :  flow230.300 * _Pct_Max_WS_Storage / 100

      condition  :  default
      value      :  0
```

```
}

Set : _Bdam_WS_Stor_Init /* Calculate the initial Beaverdam WS storage (before
computing spill to WQ and back*/
{    condition : elevation230 >= 249 or _Bdam_WS_Stor(-1) + _Bdam_Inf_WS -
_Bdam_WS_Rel >= _Bdam_Max_WS_Storage
    value     : _Bdam_Max_WS_Storage

/*   Prevent the water supply account from going negative */
    condition : default
    value     : max {  0, _Bdam_WS_Stor(-1) + _Bdam_Inf_WS - _Bdam_WS_Rel }
}

Set : _Bdam_WStoWQ_Spill /* Calculate the spillover from WS to WQ*/
{    condition : _Bdam_WS_Stor(-1) + _Bdam_Inf_WS - _Bdam_WS_Rel >
_Bdam_Max_WS_Storage /* Spillover only if projected WS storage is more than 100%
*/
    value     : _Bdam_WS_Stor(-1) + _Bdam_Inf_WS - _Bdam_WS_Rel -
_Bdam_Max_WS_Storage

    condition : default
    value     : 0
}

Set : _Bdam_Inf_WQ   {  value : _Pct_Max_WQ_Storage / 100 * _Bdam_Inf }

Set : _Bdam_WQ_Rel /* Calculate the WQ outflow to Falls from Beaverdam */
{    condition : elevation230 < 249 /* WQ Outflow only if Beaverdam is less than 249ft
(spillage is accounted for in Falls) */
    value     : flow230.300 * _Pct_Max_WQ_Storage / 100

    condition : default
    value     : 0
}

Set : _Bdam_WQ_Stor /* Calculate the Beaverdam WQ storage */
{    condition : elevation230 >= 249 or _Bdam_WQ_Stor(-1) + (
_Bdam_WStoWQ_Spill + _Bdam_Inf_WQ - _Bdam_WQ_Rel ) >=
_Bdam_Max_WQ_Storage
    value     : _Bdam_Max_WQ_Storage

/*   Prevent the water quality account from going negative */
    condition : default
    value     : max { 0, _Bdam_WQ_Stor(-1) + ( _Bdam_WStoWQ_Spill +
_Bdam_Inf_WQ - _Bdam_WQ_Rel ) }
}
```

```
Set : _Bdam_WQtoWS_Spill /* Calculate the spillover from WQ to WS */
{    condition : _Bdam_WQ_Stor(-1) + ( _Bdam_WStoWQ_Spill + _Bdam_Inf_WQ -
_Bdam_WQ_Rel ) > _Bdam_Max_WQ_Storage   /* Spillover only if projected WS
storage more than 100% */
    value     : _Bdam_WQ_Stor(-1) + ( _Bdam_WStoWQ_Spill + _Bdam_Inf_WQ -
_Bdam_WQ_Rel ) - _Bdam_Max_WQ_Storage

    condition : default
    value     : 0
}

Set : _Bdam_WS_Stor /* Calculate the final Beaverdam WS storage (after computing
spillage) */
{    condition : _Bdam_WS_Stor_Init + _Bdam_WQtoWS_Spill >=
_Bdam_Max_WS_Storage  /* Set at 100% if at or above full */
    value     : _Bdam_Max_WS_Storage

    condition : default
    value     : _Bdam_WS_Stor_Init + _Bdam_WQtoWS_Spill
}


/*  Falls Accounting  */

/* Falls Lake inflow is Falls gain + upstream flows into Falls*/
Set : _FLake_Inf
{   /* When Beaverdam is full, inflow to Falls account from Beaverdam will simply be
the Beaverdam surplus above 249 (= net inflow).  Note for the general accounting
        (i.e, not the WS/WQ accounting), storage will accumulate in Beaverdam (and
Falls) until the upper rule of 251.5 is reached.
        The point of this is to mirror what historic elevations have been */
        condition : elevation230 >= 249
        value     : inflow300 + flow140.300 + flow162.300 + flow205.300 +
flow250.300 + flow259.300 + flow270.300 + flow705.300 - delivery302 - evap300(0) +
_Bdam_Inf

/* Otherwise, inflow to Falls account from Beaverdam is simply the Beaverdam release
defined in the main.ocl (associated with drought operations).  This will be today's release
since
  this accounting file is included after the solve statement in the main.ocl file */
        condition : default
        value     : inflow300 + flow140.300 + flow162.300 + flow205.300 +
flow250.300 + flow259.300 + flow270.300 + flow705.300 - delivery302 - evap300(0) +
flow230.300
}
```

```
Set : _FLake_Inf_WS { value : _Pct_Max_WS_Storage / 100 * _FLake_Inf } /*
Calculate WS portion of Falls inflow */

/* The water supply release from Falls = the delivery to Raleigh. The delivery is
constrained in the main.ocl file to the amount in the water
        supply storage pool plus the inflow to that account. This way, the delivery will
not be made if the account has emptied */
Set : _Flake_WS_Rel { value : flow300.306 }

Set : _FLake_WS_Stor_Init /* Calculate the initial Falls WS storage (before computing
spill to WQ and back) */
{   condition : elevation300 >= 251.5 or _FLake_WS_Stor(-1) + ( _FLake_Inf_WS -
_Flake_WS_Rel ) >= _FLake_Max_WS_Storage    /* Set at 100% if at or above full */
    value    : _FLake_Max_WS_Storage

        /* Sets the water supply storage floor as zero minus the storage in Beaverdam */
    condition : default
    value    : max { 0 - _Bdam_WS_Stor, _FLake_WS_Stor(-1) + ( _FLake_Inf_WS -
_Flake_WS_Rel ) }
}

Set : _FLake_WStoWQ_Spill /* Calculate the spillover from WS to WQ */
{   condition : _FLake_WS_Stor(-1) + ( _FLake_Inf_WS - _Flake_WS_Rel ) >
[FLake_Max_WQ_WS_Storage] /* Spillover only if projected WS storage is more than
100% */
    value    : _FLake_WS_Stor(-1) + ( _FLake_Inf_WS - _Flake_WS_Rel ) -
[FLake_Max_WQ_WS_Storage]

    condition : default
    value    : 0
}

Set : _FLake_Inf_WQ { value : _Pct_Max_WQ_Storage / 100 * _FLake_Inf } /*
Calculate WQ portion of Falls inflow */

/* The water quality release from Falls = outflow from Falls. The release is constrained
in the main.ocl file to the amount in the water
        quality storage pool plus the inflow to that account. This way, the release will not
be made if the account has emptied. Note the outflow may
        include releasing surplus water above the rule curve, which technically would be
more than the water quality release. But since the account
        is reset to full above the rule curve, it makes no difference. However, to make it
more realistic, set the WQ release to the minimum flow from Falls
        (for simplicity) when the elevation is at 251.5 or above. */
Set : _Flake_WQ_Rel
```

```
{    condition  :  elevation300 >= 251.5
          value     :  min_flow300.310

          condition  :  default
          value     :  flow300.310
}

Set  :  _FLake_WQ_Stor /* Calculate the Falls WQ storage */
{    condition  :  elevation300 >= 251.5 or _FLake_WQ_Stor(-1) + (
_FLake_WStoWQ_Spill + _FLake_Inf_WQ -_Flake_WQ_Rel ) >=
_FLake_Max_WQ_Storage
    value     :  _FLake_Max_WQ_Storage

          /*  Sets the water supply storage floor as zero minus the storage in Beaverdam */
    condition  :  default
    value     :  max { 0 - _Bdam_WQ_Stor, _FLake_WQ_Stor(-1) + (
_FLake_WStoWQ_Spill + _FLake_Inf_WQ -_Flake_WQ_Rel ) } /* Appears to be
counted twice in terry's spreadsheet, but we are not here.*/
}

Set  :  _FLake_WQtoWS_Spill /* Calculate the spillover from WQ to WS */
{    condition  :  _FLake_WQ_Stor(-1) + ( _FLake_WStoWQ_Spill + _FLake_Inf_WQ -
_Flake_WQ_Rel ) > _FLake_Max_WQ_Storage   /* Spillover only if projected WS
storage more than 100% */
    value     :  _FLake_WQ_Stor(-1) + ( _FLake_WStoWQ_Spill + _FLake_Inf_WQ -
_Flake_WQ_Rel ) - _FLake_Max_WQ_Storage

    condition  :  default
    value     :  0
}

Set  :  _FLake_WS_Stor /* Calculate the final Falls WS storage (after computing
spillage) */
{    condition  :  _FLake_WS_Stor_Init + _FLake_WQtoWS_Spill >=
_FLake_Max_WS_Storage /* Set to 100% if more than full */
    value     :  _FLake_Max_WS_Storage

          condition  :  default
    value     :  _FLake_WS_Stor_Init + _FLake_WQtoWS_Spill
}

Set  :  _Total_WS_Stor  {  value  :  _FLake_WS_Stor + _Bdam_WS_Stor  } /* Sum the
total WS storage for Falls + Beaverdam */
```

```
Set : _Total_WQ_Stor { value : _FLake_WQ_Stor + _Bdam_WQ_Stor } /* Sum the total WQ storage for Falls + Beaverdam */


/* Calculate the percent of full for the WS and WQ accounts */

Set : _Bdam_WS_Pct { value : 100 * _Bdam_WS_Stor / _Bdam_Max_WS_Storage }
Set : _Bdam_WQ_Pct { value : 100 * _Bdam_WQ_Stor / _Bdam_Max_WQ_Storage }
Set : _FLake_WS_Pct { value : 100 * _FLake_WS_Stor / _FLake_Max_WS_Storage }
Set : _FLake_WQ_Pct { value : 100 * _FLake_WQ_Stor /
_FLake_Max_WQ_Storage }
Set : _Total_WS_Pct { value : 100 * _Total_WS_Stor / [Total_Max_WS_Storage] }
Set : _Total_WQ_Pct { value : 100 * _Total_WQ_Stor / [Total_Max_WQ_Storage] }
```