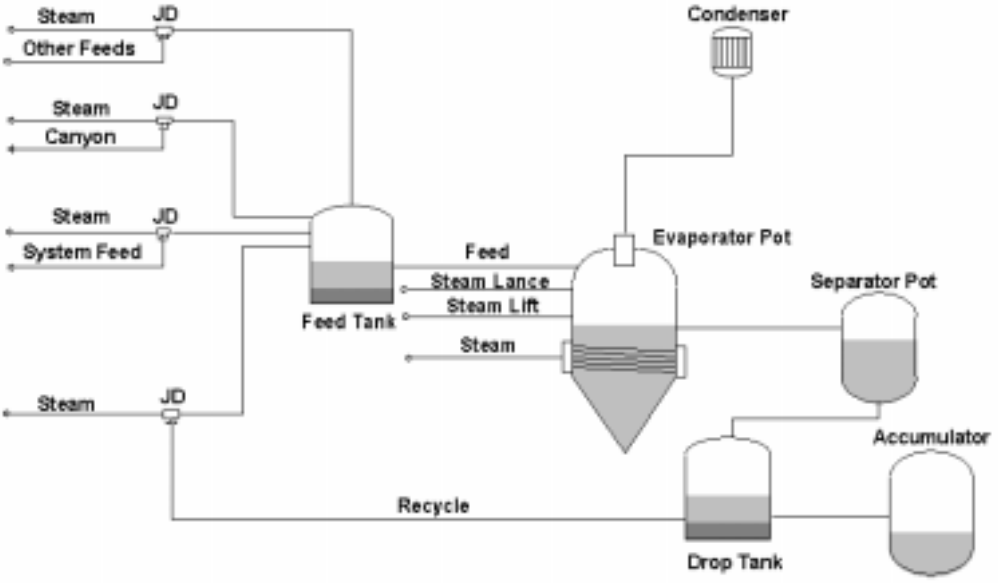
Key Words:
Evaporator,
ACM Modeling

Retention:
Permanent

A DESCRIPTION OF PHASE-1 HIGH LEVEL WASTE EVAPORATOR FLOWSHEET MODELS (U)

Thong Hang

Report Date: June 6, 2002



UNCLASSIFIED
DOES NOT CONTAIN
CLASSIFIED CONTROLLED
NUCLEAR INFORMATION

ADC &
Reviewing
Official: _____
(Name and Title)

Date: _____

Prepared by:
Westinghouse Savannah River Company
Savannah River Site
Aiken, SC 29808

SAFETY • RESPONSIBILITY • SECURITY
SRS
SAVANNAH RIVER SITE

Prepared for the U.S. Department of Energy Under
Contract Number DE-AC09-96SR18500

Key Words:
Evaporator,
ACM Modeling

Retention:
Permanent

A DESCRIPTION OF PHASE-1 HIGH LEVEL WASTE EVAPORATOR FLOWSHEET MODELS (U)

Thong Hang

Report Date: June 6, 2002

Prepared by:
Westinghouse Savannah River Company
Savannah River Site
Aiken, SC 29808

Prepared for the U.S. Department of Energy Under
Contract Number DE-AC09-96SR18500

REVIEWS AND APPROVALS

Thong Hang, Author
Engineering Development Section, SRTC

7/1/02
Date

Michael V. Gregory, Technical Reviewer
Engineering Development Section, SRTC

7/1/02
Date

Cynthia P. Holding-Smith, Manager, Design Agency
Engineering Development Section, SRTC

7/1/02
Date

Glenn A. Taylor, Cognizant Technical Function
Process Engineering, HLWD

7/1/02
Date

Paul D. d'Entremont, Design Authority
Process Engineering, HLWD

7/1/02
Date

REVISIONS

Revision No.	Date	Revision
0	6/6/02	Original Issue

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

Engineering Development Section was requested to develop dynamic flowsheet models for the 2H and 3H Evaporator Systems as the first step in constructing an Integrated Flowsheet Model for the High Level Waste system. Using the Aspen Custom Modeler™ (ACM) software package, the 2H and 3H evaporator flowsheet models are to be developed in two phases. This report describes the Phase-1 3H Evaporator Flowsheet model. The 2H Evaporator Flowsheet model that mainly differs in size from the 3H system will be delivered in Phase 2.

The major accomplishments in development of the Phase-1 3H Evaporator Flowsheet model include:

- Major operation units of the 3H evaporator system included in the current model: (1) Feed system with steam jet dilution to handle system/canyon feeds and concentrated supernate recycle from the drop tank, (2) Feed tank with cooling water, (2) Evaporator pot including steam lance, steam lift, and tube bundle steam, (3) Separator pot, (4) Drop tank with cooling water.

- Thirteen chemical compounds that comprise ~95% of wastes: H_2O, $NaNO_2$, $NaNO_3$, NaOH, Na_2SO_4, Na_2CO_3, Na_3PO_4, NaCl, NaF, $NaAlO_2$, Na_2SiO_3, $CsNO_3$, and PuO_2.

- Overall and component mass balance, and energy balance in each operation unit.

- Precipitation accounted for by chemical equilibrium. Precipitation kinetics are not implemented in Phase 1. Compounds exceeding the solubility limit in the feed and drop tanks are precipitated as solids and leave the supernate, adding to the sludge and salt phases, respectively. In the evaporator pot, solids formation is identified in the true composition calculation, but the concentrate is treated as an “apparent” bulk phase, i.e., there is no separation between the liquid and solid phases.

- Control of level and specific gravity of concentrate in the evaporator pot at user-specified values (Setpoints).

- Event-driven tasks at the ACM Flowsheet level to fill or drain tanks, to make scheduled and unscheduled feed transfers, and to recycle supernate from the drop tank to the feed tank.

1. INTRODUCTION

High Level Waste Process Engineering plans to construct an Integrated Flowsheet Model for the HLW system (Ref. 1). The objective is to provide the HLW Division with a valuable engineering tool that will be used to evaluate planning scenarios and to provide operational guidance.

Engineering Development Section was requested to develop dynamic flowsheet models for the 2H and 3H Evaporator Systems as the first step in constructing the integrated model. Using the Aspen Custom Modeler™ (ACM) software package marketed by Aspen Technology, Inc., the Evaporator Flowsheet Models will be developed in two phases. In Phase 1, a basic model will be built that includes: (1) major unit operations of the Evaporator Systems, (2) a limited number of chemical compounds, (3) salt precipitation, and (4) a limited graphical user interface (GUI) for input/output (I/O). Verification and validation of the models are required to complete Phase 1. In Phase 2, the GUI input/output will be refined. More chemical compounds, detailed chemistry and kinetics, if available, will be added to the models. The evaporator models will be designed such that moving from Phase 1 to Phase 2 is straightforward, requiring minimal revision to the existing portions of the models.

This report provides a detailed description of the ACM modeling portion of the 3H Evaporator Flowsheet Model. The GUI I/O is described in Reference 2. The verification and validation (V&V) of the model are documented in Reference 3. Although this report is limited to the 3H Evaporator Model, the two evaporator systems, 2H and 3H, mainly differ in size that can be accounted for by input variables and parameters. Most underlying block models are identical. The only two differences are in the evaporator pot (Block: Evaporator, Model: Evap): (1) The steam lift correlation, and (2) the level-volume relationship. The Flowsheet Model of the 2H Evaporator System will be delivered in Phase 2.

2. ACM MODELING

The 3H Evaporator Flowsheet Model has been created using the ACM software package marketed by Aspen Technology, Inc. ACM was selected for two reasons. First, ACM as a new generation of SPEEDUP™ provides an excellent platform for chemical process modeling, particularly when dynamic simulation is desired. The new architecture of ACM makes it very well suited to modeling both continuous and/or batch operations. The ACM modeling structure offers the advantage of easy modification to expand capabilities, allowing models to be built in phases. Second, complex and sophisticated SPEEDUP™ and ACM dynamic models have been successfully developed by SRS researchers (Refs. 4, 5, 6, 7, and 8) in the past.

Physical properties of compounds and mixtures in the evaporator models are predicted using Aspen Properties Plus® that is bundled with the ACM package. Aspen Technology also provides an Aspen/OLI interface to let ACM access the OLI database for physical properties calculation. The OLI database handles the electrolyte systems very well and contains many compounds specific to SRS wastes. Reference 9 evaluates both Aspen Properties Plus® and OLI™ in calculation of mixture densities.

3. FUNCTIONS AND REQUIREMENTS

3.1 Evaporator Configuration

The Functions and Requirements (F&R) document (Ref. 1) describes the functions and requirements for the flowsheet model to be constructed for the 2H and 3H evaporator systems. Figure 3-1 below, adapted from the F&R document, shows the required configuration of the evaporator systems.

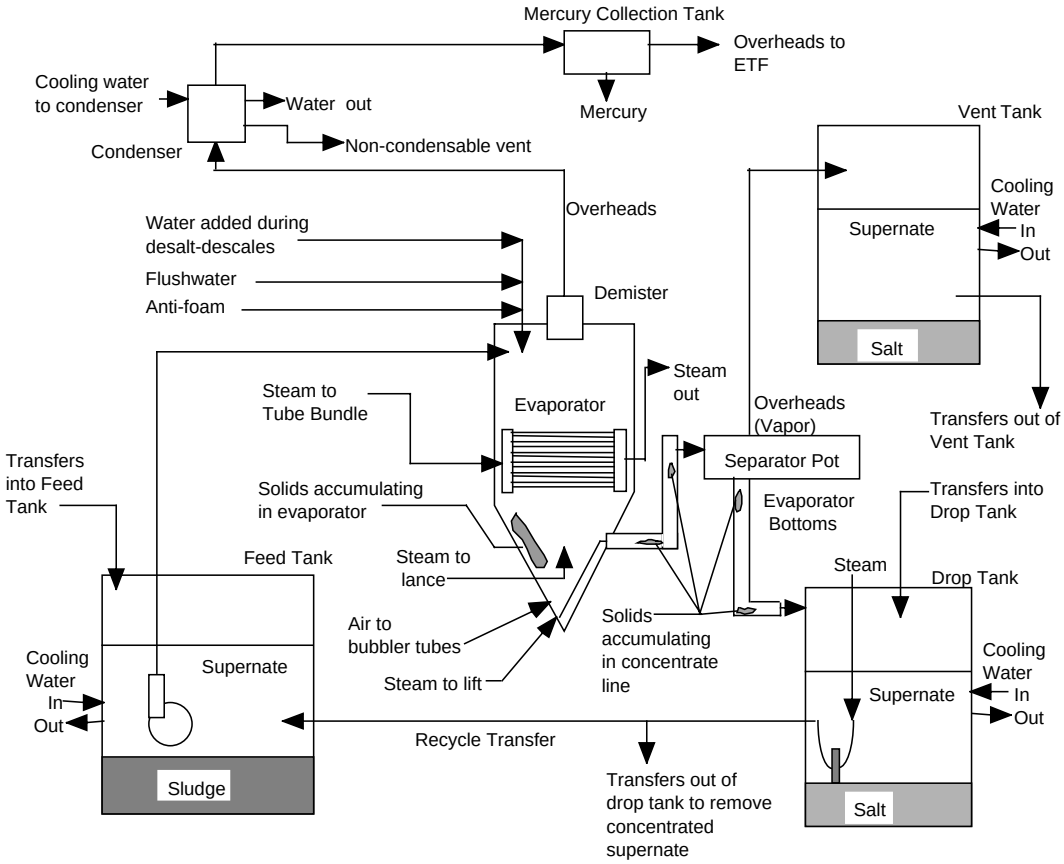


Figure 3-1. Required Evaporator Configuration

Figure 3-2 displays the features of the flowsheet which have been included in the current 3H Evaporator model. The following items have yet to be incorporated in Phase 2: (1) Mercury Collection Tank, (2) Vent Tank, (3) Streams to the evaporator pot (i.e., water during desalt-descales, flush water, and anti-foam), (4) Solids accumulation in evaporator pot and concentrate line, (5) Air to bubbler tubes in evaporator pot, and (6) Transfers into Drop Tank.

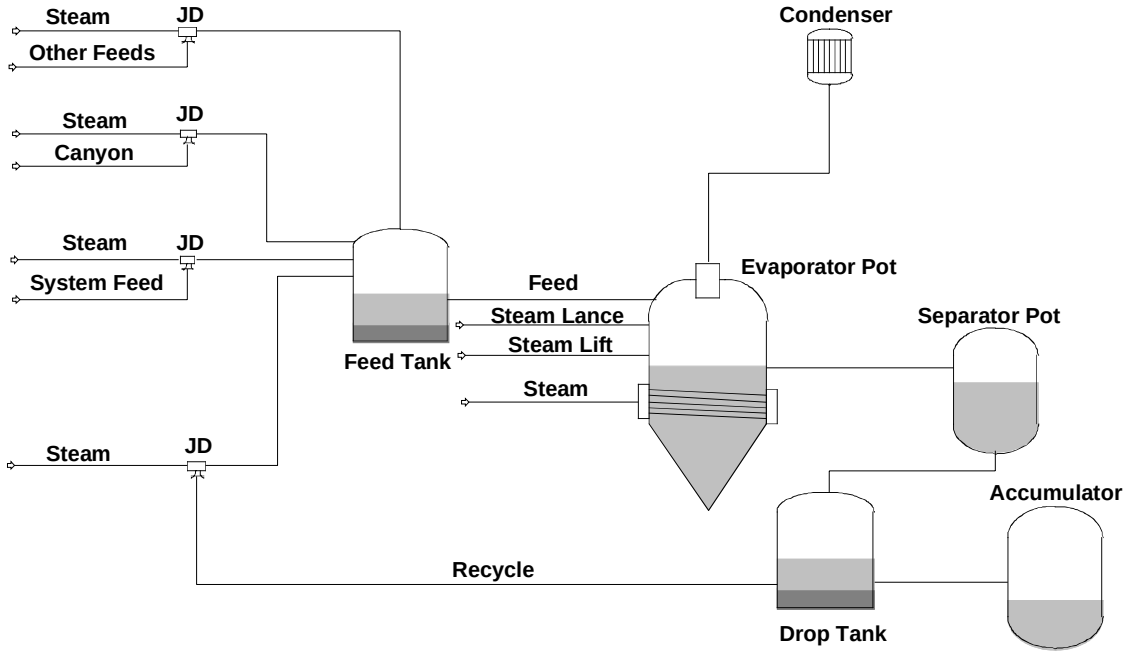


Figure 3-2. Modeled Evaporator Configuration

Note that as shown in Figure 3-2 the 2H/3H evaporator feed system is composed of three sources of feeds:

(1) ***System feed*** for Supernate waste from a designated tank,
(2) ***Canyon feed*** for waste from the canyon, and
(3) ***Other feeds***.

(1) and (2) are considered as scheduled transfers, i.e., feeds are transferred to the feed tank based on a specified schedule. Feeds of type (3) are unscheduled transfers since they are only transferred in the absence of scheduled transfers and when the feed tank liquid level is low. This feed system setup provides the users with a high degree of flexibility to select feeds to be transferred to the evaporator system. In addition to waste feeds, the feed tank also receives the concentrate supernate recycled from the drop tank.

3.2 Required Functions

Table 3-1 summarizes all functions required by the F&R document for the 2H and 3H evaporator systems. Table 1 also shows the functions (marked with X) that are currently included in the 3H Evaporator model. The remaining functions will be modeled in the Phase 2 development. The 13 chemical compounds handled by the 3H Evaporator model are listed in Table 3-2. Other compounds will be added in Phase 2. Reference 2 describes in detail the GUI I/O features of the models.

Table 3-1. Functions Required for the 2H and 3H Evaporator Models

Required Functions	Functions Included in the current 3H Evaporator Model	Remarks
Accept input from the user		
Evaporator conditions	X	
Initial tank conditions	X	
Operating parameters	X	
Time steps and modeling period	X	
Components to be modeled	X	13 compounds are included in Phase 1 (see Table 3-2). No provision for defining and/or selecting predefined sets of compounds
User comments		
User changeable parameters	X	
Predict the system behavior		
Predict flowrates that are not specified	X	
Predict the concentration of each component in each location	X	
Predict where components will exceed solubility limits	X	Instantaneous chemical equilibrium is used to test the solubility limit in the feed and drop tanks. No kinetics for precipitation.
Predict precipitation of compounds based on user preferences		Only use of instantaneous chemical equilibrium for prediction of precipitation behavior. No precipitation kinetics.
Determine how much more space gain can be obtained from a given feed		
Predict layering behavior in waste tanks		
Provide output to users		
Provide graphical output	X	
Provide printed output based on user preferences	X	
Capability to download data	X	

Table 3-2. Chemical Compounds Included in the 3H Evaporator Model

H_2O
Na_2CO_3
Na_2SiO_3
Na_2SO_4
Na_3PO_4
$NaAlO_2$
NaCl
NaF
$NaNO_2$
$NaNO_3$
NaOH
$CsNO_3$
PuO_2

4. MODEL DESCRIPTION

4.1 3H Evaporator Model Flowsheet

The 3H Evaporator ACM Model flowsheet shown in Figure 3-2 above consists of blocks and streams that are based on block or stream models. Table 4-1 lists all the blocks and streams and the underlying models. The following sections discuss all models in details.

Table 4-1. List of Blocks, Streams and Models

Block	Block Model
FeedTank	FeedTank
Evaporator	Evap
Separator	Separator
Condenser	Condenser
DropTank	DropTank
Accumulator	AccumulateTank
OrdFd_JD	JetDilution
Canyon_JD	JetDilution
SysFd_JD	JetDilution
Rec_JD	JetDilution
Stream	**Stream Model**
OrdFdJD_Steam	Steam
CanyonJD_Steam	Steam
SysFdJD_Steam	Steam
RecJD_Steam	Steam
Lance	Steam
Lift	Steam
Steam	Steam
OrderedFeed	Feed
Canyon	Feed
SystemFeed	Feed

4.2 Input Parameters/Variables

Design inputs will be shown in the discussion of individual block and stream models below. In the 3H Evaporator ACM Model, the *Specifications* Table at the Flowsheet level lists all fixed parameters/variables as well as initial variables that need to be specified prior to each simulation run. In Appendix C (Sample Calculation), Table C-1 provides a summary of these design inputs together with their values used in the sample run.

4.3 Component Lists

Six component lists are defined in the 3H Evaporator ACM Model for physical property

calculations using Aspen Properties Plus®:

(1) ***Default*** for all molecular non-radioactive components. Default component list is used in the apparent-component approach.
(2) ***Aqueous*** for all non-radioactive components (molecular, ionic, solid). Aqueous component list is used the true-component approach.
(3) ***Rads*** for all molecular radioactive components.
(4) ***Solids*** for all solid components.
(5) ***Vapor*** for steam component.
(6) ***Water*** for water component.

Table 4-2 provides a detailed list of chemical components currently included in each component list.

Table 4-2. Component Lists

Default	
Name	Formula
H2O	H2O
NA2CO3	Na2CO3
NA2SI03	Na2SiO3
NA2SO4	Na2SO4
NA3PO4	Na3PO4
NAALO2	NaAlO2
NACL	NaCl
NAF	NaF
NANO2	NaNO2
NANO3	NaNO3
NAOH	NaOH
Aqueous	
Name	Formula
ALO2-	AlO2-
CL-	Cl-
CO3--	CO3-2
F-	F-
H2O	H2O
NA+	Na+
NA2CO3	Na2CO3
NA2SI03	Na2SiO3
NA2SO4	Na2SO4
NA3PO4	Na3PO4
NAALO(S)	NaAlO2
NAALO2	NaAlO2
NACL	NaCl
NACL(S)	NaCl

NACO3(S)	Na2CO3
NAF	NaF
NAF(S)	NaF
NANO2	NaNO2
NANO2(S)	NaNO2
NANO3	NaNO3
NANO3(S)	NaNO3
NAOH	NaOH
NAOH(S)	NaOH
NAPO4(S)	Na3PO4
NASO4(S)	Na2SO4
NO2-	NO2-
NO3-	NO3-
OH-	OH-
PO4---	PO4-3
SO4--	SO4-2
Rads	
Name	Formula
CSNO3	CsNO3
PUO2	PuO2
Solids	
Name	Formula
NAALO(S)	NaAlO2
NACL(S)	NaCl
NACO3(S)	Na2CO3
NAF(S)	NaF
NANO2(S)	NaNO2
NANO3(S)	NaNO3
NAOH(S)	NaOH
NAPO4(S)	Na3PO4
NASO4(S)	Na2SO4
Vapor	
Name	Formula
H2O	H2O
Water	
Name	Formula
H2O	H2O

4.4 Ports

In ACM, ports are the I/O anchor points to which streams are connected. Through ports, variables are defined and passed between models. The 3H Evaporator ACM Model defines three types of ports:

(1) ***Main*** is used for a mixed phase of solids and liquids. The phase however is treated as if it were homogeneous, i.e., there is no separation between the liquid and solid phases,
(2) ***MainSimple*** is a simple version of Main primarily used for the steam-lifted concentrate in the evaporator pot, and
(3) ***Vapor*** is used for vapor-phase streams.

Table 4-3 shows the variables defined in each port type.

Table 4-3. Port Types

Main		
Variables	Component List	Description
F		Mass flow rate (kg/hr)
Fmol		Molar flow rate (kmol/hr)
FV		Volumetric flow rate (m^3/hr)
w	Default	Array of mass fraction
z	Default	Array of mole fraction
T		Temperature (°C)
P		Pressure (bar)
h		Molar enthalpy (GJ/kmol)
Frad	Rads	Array of molar flow rate of rad components (kmol/hr)
Radconc	Rads	Array of concentration of rad components (mol/L)
MainSimple		
Variables	Component List	Description
Fmol		Molar flow rate (kmol/hr)
Z	Default	Array of mole fraction
P		Pressure (bar)
H		Molar enthalpy (GJ/kmol)
Frad	Rads	Array of molar flow rate of rad components (kmol/hr)
Vapor		
Variables	Component List	Description
F		Mass flow rate (kg/hr)
Fmol		Molar flow rate (kmol/hr)
FV		Volumetric flow rate (m^3/hr)
w	Default	Array of mass fraction
z	Default	Array of mole fraction
T		Temperature (°C)
P		Pressure (bar)
h		Molar enthalpy (GJ/kmol)

4.5 AccumulateTank Model

The AccumulateTank model represents a simple tank for liquid accumulation. The model calculates total moles, mass and volume of the tank liquid holdup as a function of time.

Model Assumptions

1. Well-mixed vessel.
2. No chemical reactions or solids formation.
3. Tank temperature is specified by users.

Design Inputs/Outputs

- Fixed parameters/variables:

T	Specified tank temperature (°C)
P	Specified tank pressure (bar)
FV	Nominal volumetric outflow (m^3/hr)

- Initial variables:

Volume	Holdup volume (m^3)
z	Array of holdup mole fraction
RadMoles	Array of Rads component holdup moles (kmol)

- Input ports:
  Multiple liquid input ports InPort (Type: Main)

- Output port:
  Liquid output port OutPort (Type: Main)

Procedures Called

ACM Procedures	Calculated Properties
pMolWeights	Component molecular weights
pMolWeight	Average molecular weight of a mixture
pDens_Mass_Liq	Liquid mixture mass density
pEnth_Mol_Liq	Liquid mixture molar enthalpy

Algorithm

The AccumulateTank model describes a simple tank operation to accumulate a liquid holdup over time. The model performs overall and component mole balances to predict total moles, mass, volume as well as the component mole fraction of the holdup.

The overall mole balance is calculated in equation EQ01:

EQ01: $Moles = SIGMA(InPort.Connection.FMol) - OutPort.FMol;

Using average mixture molecular weight MW and liquid mixture mass density DensMass calculated by procedures *pMolWeight* and *pDens_Mass_Liq*, total mole is converted to total mass in equation EQ01a.

EQ01a: Moles * MW = Mass;

Total volume is obtained from equation EQ01b.

EQ01b: Volume * DensMass = Mass;

Equations EQ02 and EQ02a calculate mole and mass balances for non-radioactive components.

```
EQ02:   FOR i IN MainSet DO
          z(i)*$Moles + Moles*$z(i) =
            SIGMA(InPort.Connection.FMol*InPort.Connection.z(i)) -
              OutPort.FMol*OutPort.z(i);
        ENDFOR
```

EQ02a: w * SIGMA(z * MWs) = z * MWs;

Equations EQ02b and EQ02c calculate moles and concentrations for radioactive components.

```
EQ02b:  FOR i IN RadSet DO
          $RadMoles(i) = SIGMA(InPort.Connection.FRad(i)) - OutPort.FRad(i);
        ENDFOR
```

```
EQ02c:  IF Volume > 0 THEN
          radconc * Volume = RadMoles;
        ELSE
          radconc = 0;
        ENDIF
```

The logic in equation EQ02c is to prevent singularity at zero holdup volume.

In equations EQ03 to EQ03i, values of Main port-type variables are calculated and transferred to the output port OutPort.

4.6 Condenser Model

The Condenser model is used to represent the steady-state operation of a condenser.

Model Assumptions

1. Steady-state operation.

2. Well-mixed vessel.
3. Total condensation.
4. Steam is the only component of the vapor stream.

Design Inputs/Outputs

- Fixed parameters/variables:

FV	Volumetric flow of cooling water (m^3/hr)
P_in	Inlet pressure of cooling water (bar)
P_drop	Pressure drop in cooling water tube bundle (bar)
T_in	Inlet temperature of cooling water (°C)

- Input port:
  Vapor input port I1 (Type: Vapor)

- Output port:
  Condensate output port O1 (Type: Vapor)

Procedures Called

ACM Procedures	**Calculated Properties**
pDens_Mass_Liq	Liquid mixture mass density
pEnth_Mol_Liq	Liquid mixture molar enthalpy

Algorithm

The Condenser model describes a steady-state condenser operation in which a total condensation of a vapor stream occurs. Given the volumetric flow rate of cooling water, its inlet temperature and pressure, and the pressure drop in the cooling water tube bundle, the model will predict the outlet temperature of the cooling water.

First, in equations EQ01 to EQ01f, data for the vapor stream are transferred from the input port I1 to the output port O1. Since the vapor stream totally condenses, the only differences between the input and output are the volumetric flow rate (calculated in EQ01b) and the molar enthalpy (provided by EQ020b).

Cooling water calculations are performed by equations EQ02 to EQ02c. The volumetric flow rate is converted to mass flow rate (EQ02c) and to molar flow rate (EQ02). The outlet temperature and pressure are calculated in equations EQ02a and EQ02b, respectively.

The main equation of the condenser operation is the energy balance EQ03 in which the molar condensation heat flux of the vapor stream is transferred to the cooling water as follows:

EQ03: I1.FMol * (I1.h - O1.h) = FMol * (h_out - h_in);

Coupled with equations EQ020c and EQ020d that use ACM procedure *pEnth_Mol_Liq* to

calculate h_in and h_out, the outlet temperature of cooling water T_out is determined.

4.7 DropTank Model

The DropTank model is used to represent the drop tank. The tank content includes both supernate (liquid) and salt (solid). Precipitation and dissolution are accounted for by instantaneous chemical equilibrium.

Model Assumptions

1. Well-mixed vessel.
2. Precipitation accounted for by instantaneous chemical equilibrium.
3. Salt cake is composed of the solid phase only.
4. Instantaneous thermodynamic equilibrium between supernate and salt cake.
5. Negligible decay heat of radioactive components.

Design Inputs/Outputs

- Fixed parameters/variables:

Tank_Factor	Tank volume-to-level factor (m^3/m)
Coil_number	Number of active cooling coils
Heel_Level	Heel level (m)
High_Level_Limit	High level operating limit (m)
Min_Coil_Depth	Minimum coil depth (m)
Min_Supernate_Depth	Minimum supernate depth (m)
Bundle_Area	Area per coil bundle (m^2)
FV_Cooling	Volumetric flow per coil bundle (m^3/hr)
FV_Decant	Nominal decant flow to accumulator (m^3/hr)
FV_Recycle	Nominal recycle flow (m^3/hr)
T_Cooling_In	Cooling water inlet temperature (°C)
P	Operating pressure (bar)
Volume_Cooling	Cooling volume per coil bundle (m^3)
DensMol_Cooling	Cooling molar density (kmol/m^3)
HeatTransCoef	Cooling heat transfer coefficient (kW/m^2/K)
z_Cooling	Array of cooling water mole fraction
Boildownratio_value	Set value of boildown ratio

- Initial variables:

T	Tank holdup temperature (°C)
T_Cooling	Average cooling water temperature (°C)
Volume	Tank holdup volume (m^3)
z	Array of holdup mole fraction
RadMoles	Array of Rads holdup moles (kmol)

- Input ports/signals:
  Multiple liquid input ports InPort (Type: Main)

DestTank_FillStatus_Signal Destination tank fill status signal
EvapFeedFV_Signal Evaporator feed volumetric flow signal;

DestTank_FillStatus_Signal receives the value of the feed tank *RecycleFillStatus_Signal* to check if the feed tank is ready to accept recycle. *EvapFeedFV_Signal* is the value of the volumetric feed flow of the evaporator pot and used for the boil down ratio calculation.

- Output ports/signal:
  Liquid output port OutPort.Connection("Recycle") for recycle stream connection (Type: Main)
  Liquid output port OutPort.Connection("Accumulate") for transfer stream connection to accumulate tank (Type: Main)
  FillStatus_Signal Tank fill status signal

FillStatus_Signal is currently not used since the input stream (*EvaporatorBottoms*) from the separator pot is a continuous stream that is always on.

Submodel

PrecipComp	Get a name set of components to precipitate

Procedures Called

ACM Procedures	**Calculated Properties**
pMolWeights	Component molecular weights
pMolWeight	Average molecular weight of a mixture
pDens_Mass_Liq	Liquid mixture mass density
pDens_Mass_Sol	Solid mixture mass density
pEnth_Mol_Liq	Liquid mixture molar enthalpy
pEnth_Mol_Sol	Solid mixture molar enthalpy
pCp_Mol_Liq	Liquid mixture molar heat capacity
pTrueComp	Convert apparent composition to true composition

Algorithm

The DropTank model describes the drop tank operation in the evaporator system. The model is set up to receive multiple liquid input and output streams using the ACM MultiPort feature. Currently, the *EvaporatorBottoms* liquid stream from the *Separator* pot is the only input stream connected to the drop tank. There are two output streams connected to the output ports *OutPort* of the drop tank: (1) The *Recycle* stream to recycle the concentrate supernate from the drop tank to the feed tank, and (2) the *Accumulate* stream to transfer the drop tank supernate to the *Accumulator* tank. The following calculations are performed within the DropTank model:

Overall and component moles balances
Energy balance

Split of the tank holdup into supernate (liquid phase) and salt (solid phase)
Cooling system
Tank levels
Decant logic
Transfer logic
Output streams

1. Overall and Component Mole Balances

An overall mole balance is given in equation EQ01:

```
EQ01:    $Moles = SIGMA(InPort.Connection.FMol) –
                  SIGMA(OutPort.Connection.FMol) + Volume * SIGMA(Rate);
```

The *Rate* array of reaction rate is set to 0 in the current model version. However, values or equations can be assigned to *Rate* in future upgrade, once kinetic data become available. *Moles* is converted to *Mass* (EQ01a) and further to *Volume* (EQ01b).

Component mole balance calculation is performed for non-radioactive components in equation EQ02 and for radioactive components in equations EQ02a and EQ02b.

```
EQ02:    FOR i IN MainSet DO
           z(i) * $Moles + Moles * $z(i) =
             SIGMA(InPort.Connection.FMol*InPort.Connection.z(i)) -
               SIGMA(OutPort.Connection.FMol*OutPort.Connection.z(i)) + Volume
                 * Rate(i);
         ENDFOR

EQ02a:   FOR i IN RadSet DO
           $RadMoles(i) = SIGMA(InPort.Connection.FRad(i)) –
             SIGMA(OutPort.Connection.FRad(i));
         ENDFOR

EQ02b:   radconc * Volume = RadMoles;
```

2. Energy Balance

Equation EQ03 calculates the overall energy balance of the drop tank:

```
EQ03:    Moles * $h + h * $Moles =
           SIGMA(InPort.Connection.FMol*InPort.Connection.h) -
             SIGMA(OutPort.Connection.FMol*OutPort.Connection.h) +
               SIGMA(RxnH) - Q_Cooling + Q_Gamma;
```

The decay heat *Q_Gamma* of radioactive components is currently assumed to be negligible, and therefore set to 0 in equation EQ03b. The cooling heat *Q_Cooling* is calculated in equation EQ03a.

EQ03a: Q_Cooling = kW2GJperHr * HeatTransCoef * Coil_Area * (T - T_Cooling);

Coil_Area is the coil heat transfer area that is estimated by equation EQ08c, based on Askew's heat transfer model (Ref. 10).

```
EQ08c:  IF (Total_Level <= Min_Coil_Depth) THEN
          Coil_Area = 0.;
        ELSE
          Coil_Area * (High_Level_Limit - Min_Coil_Depth) =
            Bundle_Area * Coil_Number * (Total_Level - Min_Coil_Depth);
        ENDIF;
```

3. Split of Tank Holdup into Supernate (liquid) and Salt (solid)

Instantaneous chemical equilibrium is applied to account for precipitation. Those compounds in the bulk phase of the tank holdup that exceed the solubility limit are precipitated as solids and leave the supernate, adding to the salt phase. ACM procedure *pTrueComp* is used to convert apparent composition to true composition. Note that in the apparent approach only molecular species are considered, while in the true approach molecular, ionic and solid species are all accounted for. Also be aware that to use *pTrueComp* both input and output of the procedure must have the same component list. Hence, *z_apparent* (array of apparent mole fraction) as input to *pTrueComp* and *z_true* (array of true mole fraction) as output from *pTrueComp* all use the *Aqueous* component list.

First, equations EQ04 and EQ04a set *z_apparent* up to match the apparent bulk-phase composition *z* that uses the *Main* component list. Missing species in the *Aqueous* component list are set to 0.

```
EQ04:   z_apparent(MainSet) = z;
EQ04a:  z_apparent(AqueousSet - MainSet) = 0;
```

Calling *pTrueComp* (EQ20k) with *z_apparent*, tank temperature *T*, and tank pressure *P* as inputs provides *z_true*, *sfrac* (true solid to true liquid mole ratio), and *lfrac* (true liquid to apparent liquid mole ratio) as outputs. Species are allocated to the salt cake (solid phase) by equations EQ06 to EQ06f.

```
EQ06:   Moles_Solids = sfrac * lfrac * Moles;
EQ06a:  Mass_Solids = Moles_Solids * MW_Solids;
EQ06b:  Volume_Solids * DensMass_Solids = Mass_Solids;
EQ06c:  FOR i IN SolidsSet DO
          IF sfrac > 1E-6 THEN
            z_Solids(PrecipCompName(i)) * SIGMA(z_true(SolidsSet)) = z_true(i);
```

```
            ELSE
              z_Solids(PrecipCompName(i)) = 0.;
            ENDIF
          ENDFOR
EQ06d:    z_Solids(MainSet-PrecipCompSet) = 0.;
EQ06e:    RadMoles_Solids = RadMoles - RadMoles_Supernate;
EQ06f:    IF sfrac > 1E-6 THEN
            radconc_Solids * Volume_Solids =
              radconc * Volume - radconc_Supernate * Volume_Supernate;
          ELSE
            radconc_Solids = 0.;
          ENDIF
```

The logic in equations EQ06c and EQ06f is to prevent singularity when no solids form. In equation EQ06c, *PrecipCompName* is array of the name of components that precipitate. *PrecipCompName* is obtained from the submodel *PrecipComp*.

The remaining species in the bulk phase of the tank holdup are allocated to the supernate (liquid phase) by equations EQ05 to EQ05e.

```
EQ05:     Moles_Supernate = Moles - Moles_Solids;
EQ05a:    Mass_Supernate = Moles_Supernate * MW_Supernate;
EQ05b:    Volume_Supernate * DensMass_Supernate = Mass_Supernate;
EQ05c:    z_Supernate * Moles_Supernate = z * Moles - z_Solids * Moles_Solids;
EQ05d:    RadMoles_Supernate * Moles  = RadMoles * Moles_Supernate;
EQ05e:    radconc_Supernate * Volume_Supernate = RadMoles_Supernate;
```

Ultimately, both liquid and solid phases must assume the apparent composition for subsequent physical property calculations. Thus, *z_Supernate* and *z_Solids* are array of apparent mole fraction of the supernate and salt cake, respectively, that use the apparent *Main* component list.

4. Cooling System

Adapted from Reference 10, calculations of the water cooling system are carried out in equations EQ07, EQ07a, and EQ07b.

```
EQ07:     FMol_Cooling_Total = Coil_Number * FV_Cooling * DensMol_Cooling;
EQ07a:    DensMol_Cooling * (Volume_Cooling * Coil_Number) *
            (cp_mol_Cooling * 1E-6) * $T_Cooling =
              FMol_Cooling_Total * (cp_mol_Cooling * 1E-6) * (T_Cooling_In –
                T_Cooling_Out) + Q_Cooling;
EQ07b:    T_Cooling = 0.5 * (T_Cooling_In + T_Cooling_Out);
```

5. Tank Levels

Different types of tank level are calculated: Total_Level (EQ08), Supernate_Level (EQ08a), Solids_Level (EQ08b), High_Decant_Level (EQ08d), Low_Decant_Level (EQ08e).

EQ08: Total_Level = Supernate_Level + Solids_Level;
EQ08a: Supernate_Level * Tank_Factor = Volume_Supernate;
EQ08b: Solids_Level * Tank_Factor = Volume_Solids;
EQ08d: High_Decant_Level = High_Level_Limit;
EQ08e: IF Heel_Level > (Solids_Level + Min_Supernate_Depth) THEN
  Low_Decant_Level = Heel_Level;
ELSE
  Low_Decant_Level = Solids_Level + Min_Supernate_Depth;
ENDIF

Low_Decant_Level and *High_Decant_Level* are used for supernate transfer. It is expected that solids will form and accumulate in the drop tank over time. Equation EQ08e ensures that *Low_Decant_Level* is always above the solids level.

6. Decant Logic

Transfer of the drop tank supernate can be activated by turning the Drain_Signal on. Task *DropTankDrain* at the ACM Flowsheet level is designed to turn Drain_Signal on or off. A detailed discussion of tasks used in the 3H Evaporator ACM Model is provided in later sections of this report.

7. Transfer Logic

Supernate can be recycled to the feed tank or transferred to the *Accumulator* tank. The *Boildown_ratio* is used to decide which transfer should be made. Boildown_ratio is the ratio of the drop tank volumetric inlet flow to the Evaporator volumetric feed flow as shown in equation EQ11.

EQ11: IF EvapFeedFV_Signal > 0 THEN
  Boildown_ratio * EvapFeedFV_Signal = SIGMA(InPort.Connection.FV);
ELSE
  Boildown_ratio = 0;
ENDIF

In the current logic, supernate is recycled when Boildown_ratio is less than a pre-specified value *Boildownratio_value* (e.g., 0.95). The logic is captured by equation EQ11a.

EQ11a: IF Boildown_ratio < Boildownratio_value THEN
  RecycleStatus = 1;
ELSE
  RecycleStatus = 0;
ENDIF

8. Output Streams

All *Main*-type variables of the supernate phase are transferred to output ports *OutPort*. Equations EQ09 to EQ09i transfer the variables to the recycle stream and equations EQ10 to EQ10i assign these variables to the transfer stream connected to the *Accumulator* tank. Note that equations EQ09b and EQ10b, as shown below, are designed to make sure that only one output stream can be turned on at any given time.

EQ09b: OutPort.Connection("Recycle").FV =
RecycleStatus * DestTank_FillStatus_Signal * Drain_Signal * FV_Decant;

EQ10b: OutPort.Connection("Accumulate").FV =
(1-RecycleStatus) * Drain_Signal * FV_Decant;

Equation EQ09b shows that the recycle stream is turned on when all three following conditions are met:

1. RecycleStatus = 1: The drop tank is in the recycle mode. Equation EQ11a provides the logic for the recycle mode.
2. DestTank_FillStatus_Signal = 1: DestTank_FillStatus_Signal is an input signal sent from the feed tank to which the concentrated supernate of the drop tank is recycled. Value of 1 indicates that the feed tank is ready to accept the recycle stream.
3. Drain_Signal = 1: The drop tank is in the decant mode.

For the Accumulate stream (EQ10b) there is no signal from the Accumulator tank since this tank serves as a sink with no volume limit and is always ready to accept transfers.

4.8 Evap Model

The Evap model describes the operation of an evaporator pot.

Model Assumptions

1. Well-mixed vessel.
2. Precipitation accounted for by instantaneous chemical equilibrium.
3. Homogeneous bulk phase, i.e., there is no separation between the liquid and solid phases.
4. Thermodynamic equilibrium between the liquid concentrate and the vapor phase.
5. Negligible decay heat of radioactive components in the liquid concentrate.
6. Steam is the only component in the vapor phase.
7. Ideal heat exchange between tube bundle steam and liquid concentrate.
8. Total condensation of steam in tube bundle.
9. No accumulation in the vapor phase.

Design Inputs/Outputs

- Fixed parameters/variables:

FSteamBundle_Manual	Manually set tube bundle steam flow (kg/hr)
FSteamBundle_Min	Minimum tube bundle steam flow (kg/hr)
FSteamBundle_Max	Maximum tube bundle steam flow (kg/hr)
FSteamLift_Min	Minimum steam lift mass flow (kg/hr)
FSteamLift_Max	Maximum steam lift mass flow (kg/hr)
P	Specified operating pressure (bar)
P_drop	Pressure drop in tube bundles (bar)
Rate	Array of component reaction rate in the liquid concentrate (kmol/hr)
RxnH	Array of component reaction heat flow in the liquid concentrate (GJ/hr)
LevelControl_ON	Level control ON/OFF signal
Liquid_Level_LOLO	LO-LO liquid level (m)
Liquid_Level_LO	LO liquid level (m)
Liquid_Level_HI	HI liquid level (m)
Liquid_Level_HIHI	HI-HI liquid level (m)
LiquidLevel_SetPoint	Concentrate level at level setpoint (m)
SpGControl_ON	Specific gravity control ON/OFF signal
SpG_SetPoint	Specific gravity setpoint

- Initial variables:

T	Tank temperature (°C)
Volume	Liquid holdup volume (m^3)
z	Array of liquid holdup mole fraction
RadMoles	Array of Rads component holdup moles (kmol)

- Input ports/signals:
  - Liquid input port I1 for feed stream (Type: Main)
  - Vapor input port I2 for steam lance stream (Type: Vapor)
  - Vapor input port I3 for steam lift stream (Type: Vapor)
  - Vapor input port I4 for steam to tube bundle (Type: Vapor)

- Output ports/signal:
  - Liquid output port O1 for steam-lifted concentrate stream (Type: MainSimple)
  - Vapor output port O2 for vapor overheads (Type: Vapor)

FillStatus_Signal	Evaporator port fill status signal
LiquidLevel_Signal	Concentrate level signal
EvapFeedFV_Signal	Feed volumetric flow signal

FillStatus_Signal is used to indicate if the evaporator pot can receive feed transfer from the feed tank. The value of this signal is passed to *DestTankFillStatus_Signal* in the feed tank. Currently, *FillStatus_Signal* is set to 1. *EvapFeedFV_Signal* is used to pass the value of volumetric feed flow to the drop tank for a boil down ratio calculation. *LiquidLevel_Signal* is equal to *Liquid_Level*.

Procedures Called

ACM Procedures	Calculated Properties
pMolWeights	Component molecular weights
pMolWeight	Average molecular weight of a mixture
pDens_Mass_Liq	Liquid mixture mass density
pDens_Mass_Vap	Vapor mixture mass density
pEnth_Mol_Liq	Liquid mixture molar enthalpy
pEnth_Mol_Vap	Vapor mixture molar enthalpy
pBubt	Bubble point temperature
pVap_Pressures	Component vapor pressures
pAct_Coeff_Liq	Component liquid activity coefficients
pDewt	Dew point temperature
pCp_Mol_Liq	Liquid mixture molar heat capacity
pTrueComp	Convert apparent composition to true composition

Algorithm

The Evap model is used to represent the most important block of the evaporator system, i.e., the evaporator pot. In the current version, the model is designed to have four input streams (feed, steam lance, steam lift, and steam to the tube bundle) and two output streams (steam-lifted concentrate, and vapor overheads). The following calculations are performed within the Evap model:

Overall and component moles balances
Energy balance
Vapor-liquid equilibrium
Steam-lifted concentrate
Vapor overheads
True composition of concentrate
Levels and liquid level control
Specific gravity control
Fill logic

1. Overall and Component Mole Balances

A mole balance for the liquid phase is calculated in equation EQ01:

EQ01: $Moles = I1.FMol + I2.FMol - FMolLift - O2.FMol + SIGMA(Rate);

The *Rate* array of reaction rate is set to 0. However, values or equations can be assigned to *Rate* in future version, once kinetic data become available. *Moles* is converted to *Mass* (EQ01a) and further to *Volume* (EQ01b).

Component mole balance calculations are performed for non-volatile components in equation EQ02, for volatile components in equation EQ02a, and for radioactive components in EQ02c and EQ02d.

Non-volatile components:

EQ02: z(MainSet-GasSet)*$Moles + Moles*$z = I1.FMol*I1.z + Rate - FMolLift*z;

Volatile components:

EQ02a: z(GasSet)*$Moles + Moles*$z =
I1.FMol*I1.z + I2.FMol*I2.y + Rate - FMolLift*z - O2.FMol*O2.y;

Radioactive components:

EQ02c: $RadMoles = I1.FRad - O1.FRad;
EQ02d: radconc * Volume = RadMoles;

Mole fractions are converted to mass fractions in equation EQ02b.

EQ02b: w * SIGMA(z*MWs) = z * MWs;

2. Energy Balance

Equation EQ03 calculate the overall energy balance of the liquid phase:

EQ03: Moles*$h + h*$Moles =
I1.FMol*I1.h + I2.FMol*I2.h + Q + SIGMA(RxnH) – FMolLift*h –
O2.FMol*O2.h;

The *RxnH* array of component reaction heat flow is currently set to 0. Values or equations can be assigned to *RxnH*, once chemistry data become available. The heat flow *Q* of heating steam in the tube bundle is calculated in equation EQ03b.

EQ03b: Q = I4.FMol * (I4.h - hw);

No heat transfer calculation is carried out in the Evap model. Rather, an ideal heat exchange is assumed, in which all the heat released by total condensation of heating steam in the tube bundle serves to heat the evaporator concentrate. hw, molar enthalpy of condensed steam at the tube bundle outlet, is determined at temperature *TS_out* and pressure *PS_out* by calling ACM procedure *pEnth_Mol_Liq* (EQ20k). *PS_out* is calculated in equation EQ03a, and *TS_out* is obtained from a dew point calculation (EQ20j).

From the steady-state energy balance equation EQ04 for the liquid phase the boil-up rate FVap can be determined.

EQ04: FVap*(hv-h) =
(Q + I1.FMol*(I1.h-h) + I2.FMol*(I2.h-h) + SIGMA(RxnH)) * (PVap/P)^3;

The ratio (PVap/P)^3 ensures that the calculated liquid temperature *T* approaches the system boiling temperature *TB* when the total vapor pressure *PVap* approaches the system pressure *P*. Moreover, EQ04 mimics the reality that evaporation (i.e., *FVap* becomes larger than 0) readily starts before *T* reaches *TB*.

3. Vapor-Liquid Equilibrium

Vapor-liquid equilibrium (VLE) is established in equation EQ05a.

EQ05a: y(GasSet) * PVap = VP * gamma * z;

PVap is the total vapor pressure and calculated by EQ05.

EQ05: PVap = SIGMA(VP(GasSet)*gamma(GasSet)*z(GasSet));

where *VP* is array of component vapor pressures and *gamma* is array of component liquid activity coefficients. *VP* and *gamma* are obtained from ACM procedures *pVap_Pressures* (EQ20h) and *pAct_Coeff_Liq* (EQ20i), respectively. Since VLE is assumed, the evaporator pot must have the same temperature and pressure for both vapor and liquid phases.

4. Steam-Lifted Concentrate

A correlation is developed for the steam lift performance of the 3H Evaporator system based on Figure 2.5-1 in Reference 11. Figure 2.5-1 displays concentrate flow (gpm) at different specific gravities as a function of lift steam flow (pph). The steam lift correlation is given in equation EQ06f.

EQ06f: Fcalc = 60.0*3.7854118E-3*((0.327397623716747*SpG –
0.312802419170903)*(I3.F/0.45359237) - 18.4306795860704);

This correlation is applicable only to the 3H Evaporator system. New correlations must be developed for other evaporator systems (e.g., 2H or 2F) unless EQ06f assumes a more generic, non-equipment specific form. Note that the calculated volumetric flow rate *Fcalc* can be negative. Hence, the logic in equation EQ06 is applied to provide a non-negative concentrate lift flow.

EQ06:
```
IF (Fcalc > 0.0) THEN
  FMolLift * MW = Fcalc * DensMass;
  O1.FRad = Fcalc * radconc;
ELSE
  FMolLift = 1E-6;
  O1.FRad = 0.0;
ENDIF
```

In case of a zero or negative *Fcalc*, an arbitrarily small number (1E-6) is assigned to the molar concentrate lift flow *FmolLift* to avoid a convergence problem in the downstream block *Separator*.

The lifted concentrate and lift steam are combined in port O1. The MainSimple-type variables are calculated and passed to O1 in equations EQ06a to EQ06e.

EQ06a: O1.FMol = FMolLift + I3.FMol;
EQ06b: O1.z(GasSet) * O1.FMol = (FMolLift * z + I3.FMol * I3.y);
EQ06c: O1.z(MainSet-GasSet) * O1.FMol = FMolLift * z;
EQ06d: O1.P = P;
EQ06e: O1.h * O1.FMol = (FMolLift * h + I3.FMol * I3.h);

5. Vapor Overheads

The 3H Evaporator ACM Model assumes that there is no accumulation in the vapor space of the evaporator pot. Hence, the boil-up rate is the same as the flow rate of the overheads stream as shown in equation EQ07a. The Vapor-type variables are calculated and transferred to port O2 in equations EQ07 to EQ07g.

EQ07: O2.FMol * MW_Vap = O2.F;
EQ07a: O2.FMol = FVap;
EQ07b: O2.FV * Vap_DensMass = O2.F;
EQ07c: O2.w * SIGMA(y*MWs) = y*MWs;
EQ07d: O2.y = y;
EQ07e: O2.T = T;
EQ07f: O2.P = PVap;
EQ07g: O2.h = hv;

6. True Composition of Concentrate

Precipitating solids are identified by calling ACM procedure *pTrueComp* (EQ20l) to convert apparent composition to true composition. However, concentrate is treated as a homogeneous bulk phase. No attempt to separate the liquid and solid phases as done in the feed tank and drop tank. Calculations of total solids and solid components are given in equations EQ08 to EQ08d.

EQ08: z_Aqueous(MainSet) = z;
EQ08a: z_Aqueous(AqueousSet-MainSet) = 0.;
EQ08b: c_true * MW_true * Volume = Mass * z_true;
EQ08c: Moles_Solids = sfrac * lfrac * Moles;
EQ08d: c_Solids * Volume = Moles_Solids;

7. Levels and Liquid Level Control

The volume-liquid level relationship is expressed by equation EQ11.

EQ11: Volume*(1000/3.785412) = 86.1371*Liquid_Level*(100/2.54) + 1676.2606;

This correlation is developed based on data obtained from Reference 12.

A simple level control scheme is added to the Evap model to maintain the liquid level in the evaporator pot at a user-specified level value (Set point). The scheme facilitates level control without using an actual level controller. Liquid level is controlled by regulation of the lift steam flow (Ref. 11). The control scheme is captured by equations EQ12, EQ12a, and EQ12b.

EQ12:
```
FMolLift_SetPoint =
  LevelControl_ON * (I1.FMol + I2.FMol + SIGMA(Rate) - O2.FMol) *
    (Liquid_Level/LiquidLevel_SetPoint)^3;
```
EQ12a:
```
IF LevelControl_ON > 0 THEN
  (FMolLift_SetPoint * MW / DensMass) =
    60.0*3.7854118E-3*((0.327397623716747*SpG – 0.312802419170903) *
      (FSteamLift/0.45359237) - 18.4306795860704);
ELSE
  FSteamLift = FSteamLift_Min;
ENDIF
```
EQ12b:
```
IF FSteamLift <= FSteamLift_Min THEN
  I3.F = FSteamLift_Min;
ELSEIF FSteamLift >= FSteamLift_Max THEN
  I3.F = FSteamLift_Max;
ELSE
  I3.F = FSteamLift;
ENDIF
```

In equation EQ12, if liquid level is controlled (*LevelControl_ON = 1*), a required value for the molar lifted concentrate flow (*FMolLift_SetPoint*) is determined. The ratio *(Liquid_Level / LiquidLevel_SetPoint)^3* provides the direction for the control action. *FMolLift_SetPoint* will be regulated so that *Liquid_Level* approaches *LiquidLevel_SetPoint.* In equation EQ12a, using the steam lift correlation, the lift steam flow can be calculated once the lifted concentrate flow is known. The lift steam flow assumes a minimum value when no level control is desired. The logic in equation EQ12b ensures that the steam lift mass flow (*I3.F*) is always within the limit values.

There are four fixed input variables to represent four different liquid level conditions in the evaporator pot (Refs. 11, 13). When the liquid level reaches these level conditions, alarm is triggered and control actions are required:

- Liquid_Level_LOLO: Low-low liquid level condition (89 inches), tube bundle steam flow turned off (Interlock 17).
- Liquid_Level_LO: Low liquid level condition (92 inches), alarm generated.
- Liquid_Level_HI: High liquid level condition (98 inches), alarm generated.

- Liquid_Level_HIHI: High-high liquid level condition (101 inches), alarm generated, feed flow turned off (Interlock 3).

These level conditions are not used in the current Evap model. It is suggested that in a future version tasks be set up to detect if these conditions are met and warning messages be written to a warning message log file.

8. Specific gravity control

Specific gravity (SpG) is define by equation EQ10:

EQ10: SpG = DensMass/1000.;

SpG is controlled by regulation of the tube bundle steam flow. SpG increases with increasing steam flow. In the Evap model, SpG is maintained at a user-specified value (Set point) by using a modified proportional controller scheme. It should be noted that this control scheme is to provide the Evap model with a simple way to control SpG. It is not the actual control scheme used in the 3H Evaporator DCS system.

The SpG control is carried out in equations EQ13 to EQ13f.

```
EQ13:   IF SpG > SpG_Max THEN
          PV_Scale = 100;
        ELSEIF (SpG < SpG_Min) THEN
          PV_Scale = 0;
        ELSE
          PV_Scale * (SpG_Max - SpG_Min) = 100 * (SpG - SpG_Min);
        ENDIF
EQ13a:  SP_Scale * (SpG_Max - SpG_Min) = 100 * (SpG_SetPoint - SpG_Min);
EQ13b:  Bias_Scale * (FSteamBundle_Max - FSteamBundle_Min) =
          100 * (FSteamBundle_Manual - FSteamBundle_Min);
EQ13c:  PropTerm = SP_Scale - PV_Scale;
EQ13d:  IF SP_Scale > PV_Scale THEN
          IF PV_Scale > 0 THEN
            OP_Scale = Bias_Scale + Gain * PropTerm * (SP_Scale/PV_Scale)^3;
          ELSE
            OP_Scale = 100;
          ENDIF
        ELSE
          OP_Scale = Bias_Scale + Gain * PropTerm * (PV_Scale/SP_Scale)^3;
        ENDIF
EQ13e:  FSteamBundle =
          (1 - SpGControl_ON) * FSteamBundle_Manual + SpGControl_ON *
            (OP_Scale/100) * (FSteamBundle_Max - FSteamBundle_Min);
EQ13f:  IF (FSteamBundle > FSteamBundle_Max) THEN
          I4.F = FSteamBundle_Max;
```

```
ELSEIF (FSteamBundle < FSteamBundle_Min) THEN
  I4.F = FSteamBundle_Min;
ELSE
  I4.F = FSteamBundle;
ENDIF
```

The ratios *(SP_Scale/PV_Scale)^3* or *(PV_Scale/SP_Scale)^3* are used to accelerate the control action.

9. Fill logic

The fill logic had been commented out in the Evap model. *FillStatus_Signal* is currently set to 1. During testing, the fill logic was found to cause the Evap model to be unstable due to constant switching between Then and Else branches of the logic, especially during the ramp-up period of *SpG*. It is suggested that the fill logic be implemented as a task in a future version.

4.9 FeedTank Model

The FeedTank model represents the feed tank. The tank content includes both supernate (liquid) and solids (salt/sludge). Precipitation and dissolution are accounted for by instantaneous chemical equilibrium.

Model Assumptions

1. Well-mixed vessel.
2. Precipitation accounted for by instantaneous chemical equilibrium.
3. Salt/sludge layer is composed of the solid phase only.
4. Instantaneous thermodynamic equilibrium between supernate and salt/sludge.
5. Negligible decay heat of radioactive components.

Design Inputs/Outputs

- Fixed parameters/variables:

Tank_Factor	Tank volume-to-level factor (m^3/m)
Coil_number	Number of active cooling coils
Heel_Level	Heel level (m)
High_Level_Limit	High level operating limit (m)
Low_Level_Limit	Low level operating limit (m)
Min_Coil_Depth	Minimum coil depth (m)
Min_Supernate_Depth	Minimum supernate depth (m)
WorkingSpace_Volume	Working space volume (m^3)
Bundle_Area	Area per coil bundle (m^2)
FV_Cooling	Volumetric flow per coil bundle (m^3/hr)
FV_Decant	Nominal decant flow to the evaporator pot (m^3/hr)
T_Cooling_In	Cooling water inlet temperature (°C)

P	Operating pressure (bar)
Volume_Cooling	Cooling volume per coil bundle (m^3)
DensMol_Cooling	Cooling molar density ($kmol/m^3$)
HeatTransCoef_Solids	Solids heat transfer coefficient ($kW/m^2/K$)
HeatTransCoef	Cooling heat transfer coefficient ($kW/m^2/K$)
z_Cooling	Array of cooling water mole fraction

- Initial variables:

T	Tank holdup temperature (°C)
T_Cooling	Average cooling water temperature (°C)
Volume	Tank holdup volume (m^3)
z	Array of holdup mole fraction
RadMoles	Array of Rads holdup moles (kmol)

- Input ports/signals:
  - Liquid input port InPort.Connection("JDOrdFeed") for unscheduled feed stream (Type:Main)
  - Liquid input port InPort.Connection("JDCanyon") for scheduled canyon feed stream (Type:Main)
  - Liquid input port InPort.Connection("JDSysFeed") for scheduled system feed stream (Type:Main)
  - Liquid input port InPort.Connection("JDRecycle") for recycle stream (Type:Main)
  - DestTank_FillStatus_Signal Destination tank fill status signal

DestTank_FillStatus_Signal receives the value of the evaporator *FillStatus_Signal* to check if the evaporator pot is ready to accept feed. In the current version *FillStatus_Signal* is set to 1.

- Output ports/signal:
  - Liquid output port OutPort for feed transfer to the evaporator pot (Type: Main)
  - FeedFillStatus_Signal Tank fill status signal for input feed streams
  - RecycleFillStatus_Signal Tank fill status signal for recycle stream

FeedFillStatus_Signal is to indicate if the feed tank is ready to accept feed transfers. *FeedFillStatus_Signal* value is set by *Task FeedTankFeedFill*. *RecycleFillStatus_Signal* is an indicator to show if the feed tank is ready to receive recycle stream from the drop tank. *RecycleFillStatus_Signal* value is set by *Task FeedTankRecycleFill*.

Submodel

PrecipComp	Get a name set of components to precipitate

Procedures Called

ACM Procedures	**Calculated Properties**
pMolWeights	Component molecular weights
pMolWeight	Average molecular weight of a mixture

pDens_Mass_Liq	Liquid mixture mass density
pDens_Mass_Sol	Solid mixture mass density
pEnth_Mol_Liq	Liquid mixture molar enthalpy
pEnth_Mol_Sol	Solid mixture molar enthalpy
pCp_Mol_Liq	Liquid mixture molar heat capacity
pTrueComp	Convert apparent composition to true composition

Algorithm

The FeedTank model is used to describe the feed tank operation in the evaporator system. The model is very similar to the DropTank model discussed above. Hence, only a brief description of the FeedTank model is provided. In the current version, there are four input streams connected to the input ports *InPort* of the feed tank: (1) The unscheduled feed stream, (2) the scheduled feed stream from canyon, (3) the scheduled feed stream from designated tanks, and (4) the recycle stream from the drop tank. This structure is designed to allow the feed tank to handle three sources of input streams, i.e., scheduled feeds, unscheduled feeds, and recycle. The *Feed* liquid stream is the only output stream connected to the output port *OutPort* of the feed tank. The following calculations are performed within the FeedTank model:

Overall and component moles balances
Energy balance
Split of the tank holdup into supernate (liquid phase) and salt/sludge (solid phase)
Cooling system
Tank levels
Decant logic
Fill logic
Output stream

1. *Overall and Component Mole Balances*

Equation EQ01 provides the overall mole balance. The *Rate* array of reaction rate is set to 1. Values or equations can be assigned to *Rate* in future upgrade. *Moles* is converted to *Mass* (EQ01a), and to *Volume* (EQ01b). Equation EQ01c totals all volumetric liquid input flows.

EQ01c: FV_In = SIGMA(InPort.Connection.FV);

Component mole balance calculation is performed for non-radioactive components in equation EQ02 and for radioactive components in equations EQ02a and EQ02b.

2. *Energy Balance*

Equation EQ03 calculates the overall energy balance of the feed tank. The decay heat *Q_Gamma* of radioactive components is assumed to be negligible and set to 0 in equation EQ03b. The cooling heat *Q_Cooling* is calculated in equation EQ03a with the coil heat

transfer area *Coil_Area* estimated by equation EQ08c, based on Askew's heat transfer model (Ref. 10).

3. *Split of Tank Holdup into Supernate (liquid) and Salt/Sludge (solid)*

Instantaneous chemical equilibrium is applied to account for precipitation. Compounds in the bulk phase of the tank holdup that exceed the solubility limit are precipitated as solids and leave the supernate, adding to the salt/sludge phase. ACM procedure *pTrueComp* is used to convert apparent composition to true composition.

First, equations EQ04 and EQ04a set *z_apparent* up to match the apparent bulk-phase composition *z* that uses the Main component list. *z_true*, *sfrac* (true solid to true liquid mole ratio), and *lfrac* (true liquid to apparent liquid mole ratio) obtained from equation EQ20k are used to allocate species to the solid phase (equations EQ06 to EQ06f). The remaining species in the bulk phase of the tank holdup are allocated to the supernate (liquid phase) by equations EQ05 to EQ05e.

4. *Cooling System*

Based on Reference 10, calculations of the water cooling system are carried out in equations EQ07, EQ07a, and EQ07b.

5. *Tank Levels*

Different types of tank level are calculated: *Total_Level* (EQ08), *Supernate_Level* (EQ08a), *Solids_Level* (EQ08b), *High_Decant_Level* (EQ08d), and *Low_Decant_Level* (EQ08e). *Low_Decant_Level* and *High_Decant_Level* are used for supernate transfer. Solids may form and accumulate in the feed tank over time. Equation EQ08e ensures that Low_Decant_Level is always above the solids level. Additionally, a new level, *WorkingSpace_Level*, is introduced and defined by equation EQ08f.

EQ08f: WorkingSpace_Level =
High_Level_Limit - (WorkingSpace_Volume / Tank_Factor);

The working space concept is applied to avoid the volume deadlock case in which both feed and drop tanks are completely full. This would result in a shutdown of the evaporator system, since there is no place to recycle the drop tank concentrate. A pre-defined WorkingSpace_Volume (e.g., 100,000 gals) would reserve a minimum volume that is always available for the recycle stream.

6. *Decant Logic*

Turning the *Drain_Signal* on causes the feed tank supernate to be transferred to the evaporator pot. Task *FeedTankDrain* at the ACM Flowsheet level is designed to turn *Drain_Signal* on or off. Discussion of Task *FeedTankDrain* is provided in later sections.

7. *Fill Logic*

The feed tank receives either waste feed streams or recycle stream from the drop tank. Task *FeedTankFeedFill* turns *FeedFillStatus_Signal* on or off for waste feed transfer. Task *FeedTankRecycleFill* activates *RecycleFillStatus_Signal* for recycle.

8. *Output Stream*

All Main-type variables of the supernate phase are calculated and transferred to the liquid output port *OutPort* by equations EQ09 to EQ09i.

4.10 JetDilution Model

The JetDilution model is used to represent the steady-state operation of a steam jet to transfer a waste solution.

Model Assumptions

1. Steady-state operation.
2. Steam jet dilution process can be represented by a dilution factor.

Design Inputs/Outputs

- Fixed parameters/variables:

Dilution_factor	Specified dilution factor (%)
Jet_ON	JD on/off flag (1: JD on, 0: JD off)

- Input ports:

InPortLiquid	Liquid input port (Type: Main)
InPortSteam	Steam input port (Type: Vapor)

- Output port:

OutPort	Liquid output port (Type: Main)

Procedures Called

ACM Procedures	**Calculated Properties**
pMolWeights	Component molecular weights
pDens_Mass_Liq	Liquid mixture mass density
pEnth_Mol_Liq	Liquid mixture molar enthalpy

Algorithm

The JetDilution model describes the steady-state operation of a steam jet to transfer waste. The model accepts two input streams (liquid and steam) and delivers a mixed liquid output stream. It is assumed that the steam jet operation can be represented by a dilution factor.

First, the mass flow rate of the steam input stream is determined based on the user-specified dilution factor.

EQ01: InPortSteam.F = F_Steam;

EQ01a:
```
IF Jet_ON > 0 THEN
  (dilution_factor/100) * InPortLiquid.F = F_Steam;
ELSE
  F_Steam = 0.;
ENDIF
```

In equation EQ01a, if the *Jet_ON* flag is turned on (i.e., *Jet_On = 1*), the steam jet operation is activated and the steam mass flow is determined. Otherwise, the JetDilution model functions as a transfer pump with zero steam flow.

The overall mole and mass balances are calculated in equations EQ02 and EQ02a, respectively. Equation EQ02b converts mass flow to volumetric flow.

EQ02: OutPort.FMol = InPortLiquid.FMol + InPortSteam.FMol;

EQ02a: OutPort.F = InPortLiquid.F + InPortSteam.F;

EQ02b:
```
IF InPortLiquid.FMol > 0.1 THEN
  OutPort.FV * DensMass = OutPort.F;
ELSE
  OutPort.FV = 0;
ENDIF
```

Equations EQ03 and EQ03a calculate mole and mass balances for non-radioactive components.

EQ03:
```
IF InPortLiquid.FMol > 0.1 THEN
OutPort.z(WaterSet) * OutPort.FMol =
  InPortLiquid.z * InPortLiquid.FMol + InPortSteam.y * InPortSteam.FMol;
OutPort.z(MainSet-WaterSet) * OutPort.FMol =
  InPortLiquid.z * InPortLiquid.FMol;
ELSE
  OutPort.z(WaterSet) = 1.;
  OutPort.z(MainSet-WaterSet) = 0.;
ENDIF
```

EQ03a: OutPort.w * SIGMA(OutPort.z * MWs) = OutPort.z * MWs;

Equations EQ03b and EQ03c calculate moles and concentrations for radioactive components.

EQ03b: OutPort.FRad = InPortLiquid.FRad;

EQ03c:
```
IF InPortLiquid.FMol > 0.1 THEN
  OutPort.radconc * OutPort.FV = OutPort.FRad;
ELSE
  OutPort.radconc = 0.;
ENDIF
```

Energy balance is provided by equation EQ04. Note that the input temperature is simply passed to the output port *OutPort*, if there is no liquid waste flow.

EQ04:
```
IF InPortLiquid.FMol > 0.1 THEN
  OutPort.h * OutPort.FMol =
    InPortLiquid.h * InPortLiquid.FMol +
      InPortSteam.h * InPortSteam.FMol;
ELSE
  OutPort.T = InPortLiquid.T;
ENDIF
```

Pressure in the output port *OutPort* is set equal to pressure of the liquid input stream as indicated by equation EQ04a.

EQ04a: OutPort.P = InPortLiquid.P;

4.11 Separator Model

The separator model describes the steady-state operation of a vapor-liquid separator.

Model Assumptions

1. Steady-state operation.
2. Vapor-liquid (VL) separation accounted for by VL equilibrium (VLE).
3. Adiabatic mixing and separation.

Design Inputs/Outputs

- Fixed parameters/variables:

P	Specified operating pressure (bar)
TRef	Ambient temperature (°C)
UA	Product of (Heat transfer coefficient * Surface area)

- Input port:
  Liquid input port I1 for steam-lifted concentrate stream (Type: MainSimple)

- Output ports:
  Liquid output port O1 for evaporator concentrate stream (Type: Main)

Vapor output port O2 for lift steam stream (Type: Vapor)

Procedures Called

ACM Procedures	Calculated Properties
pMolWeights	Component molecular weights
pMolWeight	Average molecular weight of a mixture
pDens_Mass_Liq	Liquid mixture mass density
pDens_Mass_Vap	Vapor mixture mass density
pFlash	Vapor-liquid flash at given temperature and pressure

Algorithm

The Separator represents a steady-state operation of a VL separator. The phase separation is accomplished by calling the ACM *pFlash* procedure.

In equations EQ01c and EQ01d, a general stream mixing with heat loss is set up. However, in the current evaporator flowsheet, the mixing and separation process is assumed to be adiabatic. Hence, *UA* in EQ01d is set to 0, resulting in zero heat loss.

EQ01c: h * I1.FMol = I1.h * I1.FMol - Q;

EQ01d: Q = UA * (T - TRef);

The overall mole balance is given in equation EQ02.

EQ02: I1.FMol = O1.FMol + O2.FMol;

The energy balance is calculated by equation EQ03.

EQ03: h * I1.FMol = O1.h * O1.FMol + O2.h * O2.FMol;

Equation EQ03 coupled with *pFlash* (EQ020e) determines the equilibrium temperature *T* at which the phase separation occurs.

The liquid phase is allocated to output port O1. Variables of the Main type are calculated and passed to O1 in equations EQ04 to EQ04h.

EQ04: O1.F = O1.FMol * MW;
EQ04a: O1.FV * DensMass = O1.F;
EQ04b: O1.w * SIGMA(x*MWs) = x * MWs;
EQ04c: O1.z = x;
EQ04d: O1.T = T;
EQ04e: O1.P = P;
EQ04f: O1.h = hl;
EQ04g: O1.FRad = I1.FRad;

```
EQ04h:      IF O1.F > 1E-3 THEN
              O1.radconc * O1.FV = O1.FRad;
            ELSE
              O1.radconc = 0.;
            ENDIF
```

The vapor phase is allocated to vapor output port O2 by equations EQ05 to EQ05g.

```
EQ05:       O2.FMol = I1.FMol * vapfrac;
EQ05a:      O2.F = O2.FMol * MW_Vap;
EQ05b:      O2.FV * Vap_DensMass = O2.F;
EQ05c:      O2.w * SIGMA(y*MWs) = y * MWs;
EQ05d:      O2.y = y;
EQ05e:      O2.T = T;
EQ05f:      O2.P = P;
EQ05g:      O2.h = hv;
```

4.12 PrecipComp Submodel

The PrecipComp submodel is used to identify compounds that may form solids. PrecipComp is called in both the FeedTank and DropTank models.

Two sets of strings/parameters are provided when PrecipComp is called: PrecipCompSet, and PrecipCompName.

1. PrecipCompSet is a set of names of components that form solids. This set currently includes NAALO2, NACL, NA2CO3, NAF, NANO2, NANO3, NAOH, NA3PO4, NA2SO4.

2. PrecipCompName is an array of stringparameters that provide the name of components that correspond to the name of solid components. Component names are assigned to the array as follows.

```
PrecipCompName("NAALO(S)"): "NAALO2";
PrecipCompName("NACL(S)"): "NACL";
PrecipCompName("NACO3(S)"): "NA2CO3";
PrecipCompName("NAF(S)"): "NAF";
PrecipCompName("NANO2(S)"): "NANO2";
PrecipCompName("NANO3(S)"): "NANO3";
PrecipCompName("NAOH(S)"): "NAOH";
PrecipCompName("NAPO4(S)"): "NA3PO4";
PrecipCompName("NASO4(S)"): "NA2SO4";
```

It is obvious that PrecipComp submodel be updated when more chemical compounds that may form solids are added to the current component lists.

4.13 Feed Stream Model

The Feed stream model is used to handle waste feed streams.

Model Assumptions

None

Design Inputs/Outputs

- Fixed parameters/variables:

FV_Feed	Nominal volumetric feed flow (m^3/hr)
Feed_Signal	On/Off feed signal
T	Specified feed temperature (°C)
P	Specified feed pressure (bar)
conc	Array of specified feed concentration (mol/L)
radconc	Array of rads concentration (mol/L)
TaskNum	Counter for feed tasks

- Initial variable:

Volume	Transfer volume of feed

- Input port/signal:

DestTank_FillStatus_Signal	Destination tank fill status signal

DestTank_FillStatus_Signal is the fill status signal of a tank to which feed is transferred. The signal indicates if the tank is ready to accept feed transfers. In the evaporator flowsheet, the destination tank for all waste feeds is the feed tank. Hence, *DestTank_FillStatus_Signal* is connected to the feed tank's *FeedFillStatus_Signal*.

- Output port:
  Liquid output port O1 for feed stream (Type: Main)

Procedures Called

ACM Procedures	**Calculated Properties**
pMolWeights	Component molecular weights
pMolWeight	Average molecular weight of a mixture
pDens_Mass_Liq	Liquid mixture mass density
pEnth_Mol_Liq	Liquid mixture molar enthalpy

Algorithm

From the input variables (*FV_Feed, T, P, conc, radconc, Feed_Signal*) the Main-type variables are calculated and passed to the ouput port O1 by equations EQ01 to EQ16.

EQ01: z * SIGMA(conc) = conc;
EQ02: w * SIGMA(z*MWs) = z*MWs;
EQ03: FV = DestTank_FillStatus_Signal * Feed_Signal * FV_Feed;
EQ04: FMass = FV * Liq_DensMass;
EQ05: FMol * MW = FMass;
EQ06: FRad = FV * radconc;
EQ07: O1.F = FMass;
EQ08: O1.FMol = FMol;
EQ09: O1.FV = FV;
EQ10: O1.w = w;
EQ11: O1.z = z;
EQ12: O1.T = T;
EQ13: O1.P = P;
EQ14: O1.h = h;
EQ15: O1.FRad = FRad;
EQ16: O1.radconc = radconc;

As indicated by equation EQ03, three conditions must be met for a waste feed solution to be transferred: (1) Feed tank is ready to accept transfer (*DestTank_FillStatus_Signal = 1*), (2) Feed_Signal = 1, and (3) FV_Feed > 0.

The transfer volume is calculated by equation EQ17.

EQ17: $Volume = FV;

The cumulative transfer volume obtained from EQ17 is used to stop the feed transfer by checking against a user-specified transfer volume. The check is carried out by tasks. The variable *TaskNum* is also required by these tasks in order to activate the feed transfers in sequential order. Note that all the tasks to handle feed transfers (scheduled or unscheduled) are generated by the GUI Input/Output using Microsoft Visual Basic. A detailed discussion of these tasks will be given in a later section.

4.14 Steam Stream Model

The Steam stream model is used to handle steam streams

Model Assumptions

1. Saturated steam.

Design Inputs/Outputs

- Fixed parameters/variables:

FMass	Mass flow, kg/hr
P	Specified steam pressure (bar)
Steam_Signal	On/Off steam signal

- Output port:
  Liquid output port O1 for steam stream (Type: Vapor)

Procedures Called

ACM Procedures	Calculated Properties
pMolWeights	Component molecular weights
pMolWeight	Average molecular weight of a mixture
pDewt	Dew point temperature of a vapor mixture at fixed pressure
pDens_Mass_Vap	Vapor mixture mass density
pEnth_Mol_Vap	Vapor mixture molar enthalpy

Algorithm

From the input variables (*FMass, P, and Steam_Signal*) the Vapor-type variables are calculated and passed to the output port O1 by equations EQ01 to EQ11.

EQ01: y * SIGMA(w/MWs) = w/MWs;
EQ02: FV * Vap_DensMass = FMass;
EQ03: FMol * MW = FMass;
EQ04: O1.F = Steam_Signal * FMass;
EQ05: O1.FV = Steam_Signal * FV;
EQ06: O1.w = w;
EQ07: O1.FMol = Steam_Signal * FMol;
EQ08: O1.y = y;
EQ09: O1.T = T;
EQ10: O1.P = P;
EQ11: O1.h = h;

To activate a steam stream two conditions must be met as shown by equation EQ04: (1) *Steam_Signal = 1*, and (2) *FMass > 0*. Steam temperature *T* is determined from a dew point calculation (EQ20b) at the specified pressure *P*.

4.15 Tasks

Tasks are a major improvement of ACM over SPEEDUP™. Tasks provide an effective structure to handle batch processes and discrete-event simulation. Using tasks, the following actions can be easily implemented during the simulation (Ref. 14): changing the value of some variables, writing messages, suspending the simulation, creating snapshots, invoking scripts etc. Figure 4.1, adopted from Reference 14, illustrates how the task manager interacts with the simulation server and the Graphical User Interface within ACM.

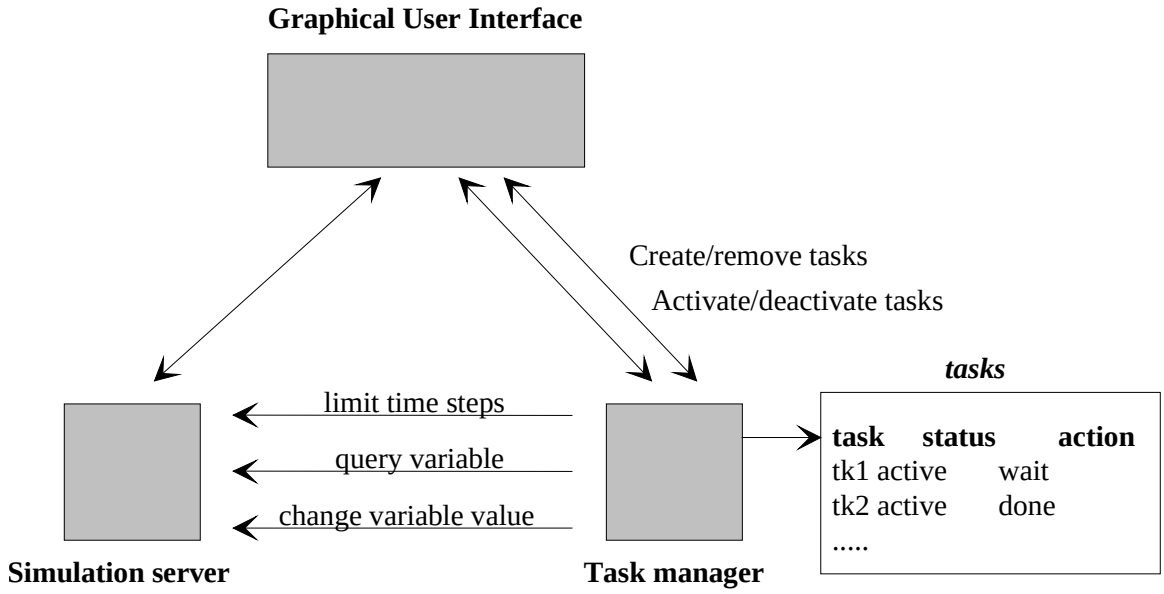


Figure 4-1. ACM Task Manager

The Evaporator Flowsheet models make use of the ACM Task feature. A number of event-driven tasks at the Flowsheet level are created to serve different functions such as filling and draining tanks, making feed transfers, recycling concentrated supernate. Table 4-4 lists all the tasks used in the model. There are two categories of tasks:

1. Fixed tasks are tasks created within ACM. Fixed tasks do not change with simulations.
2. "On-the-fly" tasks are tasks created by the GUI Input/Output (using Microsoft Visual Basic). Form and number of these tasks change in each simulation, depending on user specifications.

Table 4-4. List of Tasks

Fixed Tasks Created within ACM	
Task Name	Description
DropTankDrain	Turn Drain_Signal in the DropTank block On (= 1) or Off (= 0)
FeedTankDrain	Turn Drain_Signal in the FeedTank block On (= 1) or Off (= 0)
FeedTankFeedFill	Turn FeedFillStatus_Signal in the FeedTank block On (= 1) or Off (= 0)
FeedTankRecycleFill	Turn RecycleFillStatus_Signal in the FeedTank block On (= 1) or Off (= 0)
RecycleSchedule	Turn Feed_Signal in SystemFeed, Canyon and OderedFeed streams On (= 1) when Drain_Signal in the DropTank block is 0 and vice versa
"On-The-Fly" Tasks Created by GUI Input/Output (using Microsoft Visual Basic)	
Task Name	Description
OrderedTransfer***i***	Specify the unscheduled feed transfers via the OrderedFeed

	stream. ***i*** is the sequential order number of this type of transfer. For example: OrderedTransfer1, OrderedTransfer2, etc.
SchCanyonTransfer***j***	Specify the scheduled canyon feed transfers via the Canyon stream. ***j*** is the sequential order number of this type of transfer. For example: SchCanyonTransfer1, SchCanyonTransfer2, etc.
SchTankTransfer***k***	Specify the scheduled system (tank) feed transfers via the SystemFeed stream. ***k*** is the sequential order number of this type of transfer. For example: SchTankTransfer1, SchTankTransfer2, etc.

Discussion of each task is given in detail in the following subsections.

4.15.1 DropTankDrain Task

The DropTankDrain task is used to trigger the *Drain_Signal* variable in the drop tank based on the tank level. A listing for the DropTankDrain task is given in Table 4-5 together with line numbers displayed in the first column.

Table 4-5. Listing for DropTankDrain Task

```
1            Task DropTankDrain Runs When Time == 0
2               DrainSignal_ON as realparameter;
3               DrainSignal_OFF as realparameter;
4               DrainSignal_ON: 1;
5               DrainSignal_OFF: 0;
6               If DropTank.Drain_Signal == 0 Then
7                  // Tank level goes from low to high
8                  Wait For DropTank.Total_Level >= (1-DropTank.Drain_Signal) *
                      DropTank.High_Decant_Level;
9                  DropTank.Drain_Signal: DrainSignal_ON;
10                 Restart When DropTank.Total_Level < 1.1 *
                      DropTank.Low_Decant_Level;
11              Else
12                 // Tank level goes from high to low
13                 Wait For DropTank.Total_Level <= DropTank.Drain_Signal *
                      DropTank.Low_Decant_Level + (1-DropTank.Drain_Signal) *
                         DropTank.High_Decant_Level;
14                 DropTank.Drain_Signal: DrainSignal_OFF;
15                 Restart When DropTank.Total_Level > 0.9 *
                      DropTank.High_Decant_Level;
16              EndIf
17           End
```

This task is activated at the beginning of the simulation (Line 1). Lines 2 to 4 define parameters *DrainSignal_ON* and *DrainSignal_OFF* and assign values to them. The Then branch (Lines 6-10) of the If-Else-EndIf conditional is for the non-drain mode (i.e., *DropTank.Drain_Signal = 0*) and the Else branch (Lines 11-16) is for the drain mode (i.e., *DropTank.Drain_Signal = 1*). In ACM, statements with // at the beginning of the line are treated as comments (Lines 7 and 12).

First, in the non-drain mode when the tank level goes from low to high, no action is taken until the tank level becomes larger than or equal to *High_Decant_Level* (Line 8), at which *Drain_Signal* is turned on (Line 9) to start the recycle of concentrated supernate to the feed tank. The (*1-DropTank.Drain_Signal*) group on the right side of the conditional expression on Line 8 is used so that the task will move from Line 8 to Line 9 if the user manually changes the value of *Drain_Signal* from 0 to 1 during the simulation. Without (*1-DropTank.Drain_Signal*), the task would stay idle at Line 8.

Upon switching from non-drain to drain mode (Line 9), when the tank level becomes less than (*1.1 * Low_Decant_Level*), the task is restarted (i.e., going back to Line 2) and then subsequently proceeds to the Else branch (Line 11). No action is taken until the tank level becomes less than or equal to *Low_Decant_Level* (Line 13), at which *Drain_Signal* is turned off (Line 14). Similar to Line 8, the right side of the conditional expression on Line 13 is so formulated that the task can move from Line 13 to Line 14 in case the user manually changes *Drain_Signal* from 1 to 0 during the simulation. A restart is triggered when the tank level exceeds (*0.9 * High_Decant_Level*) as indicated by Line 15.

4.15.2 FeedTankDrain Task

The FeedTankDrain task triggers the *Drain_Signal* variable in the feed tank based on the tank level. Listing for the FeedTankDrain task is given in Table 4.6. Line numbers are displayed in the first column of the Table.

Table 4-6. Listing for FeedTankDrain Task

```
1            Task FeedTankDrain Runs When Time == 0
2               DrainSignal_ON as realparameter;
3               DrainSignal_OFF as realparameter;
4               DrainSignal_ON: 1;
5               DrainSignal_OFF: 0;
6               If FeedTank.Drain_Signal == 0 Then
7                  // Tank level goes from low to high
8                  Wait For FeedTank.Total_Level >= FeedTank.High_Decant_Level;
9                  FeedTank.Drain_Signal: DrainSignal_ON;
10                 Restart When FeedTank.Total_Level < 1.1 *
                      FeedTank.Low_Decant_Level;
11              Else
12                 // Tank level goes from high to low
13                 Wait For FeedTank.Total_Level <= FeedTank.Low_Decant_Level;
14                 FeedTank.Drain_Signal: DrainSignal_OFF;
15                 Restart When FeedTank.Total_Level > 0.9 *
                      FeedTank.High_Decant_Level;
16              EndIf
17           End
```

This task is very similar to the DropTankDrain task discussed in the previous subsection. The task is activated at time 0 of the simulation (Line 1). *DrainSignal_ON* and *DrainSignal_OFF* are defined on Lines 2 and 3, respectively, and values are assigned to them on Lines 4 and 5. The Then branch (Lines 6-10) of the If-Else-EndIf conditional is for the non-drain mode (*FeedTank.Drain_Signal = 0*) while the Else branch (Lines 11-16) is for the drain mode (*FeedTank.Drain_Signal = 1*).

In the non-drain mode when the tank level goes from low to high, no action is taken until the tank level becomes larger than or equal to *High_Decant_Level* (Line 8), at which Drain_Signal is turned on (Line 9) to allow supernate to be transferred to the evaporator pot.

Upon switching from non-drain to drain mode, when the tank level becomes less than (*1.1 * Low_Decant_Level*), the task is restarted (i.e., going back to Line 2) and then subsequently proceeds to the Else branch (Line 11). No action is taken until the tank level becomes less than or equal to *Low_Decant_Level* (Line 13), at which *Drain_Signal* is turned off (Line 14). The task is restarted when the tank level exceeds (*0.9 * High_Decant_Level*) as indicated by Line 15.

4.15.3 FeedTankFeedFill Task

The FeedTankFeedFill task triggers the *FeedFillStatus_Signal* signal in the feed tank based on the tank level. An activated *FeedFillStatus_Signal* indicates that the feed tank is ready to accept feed transfers. Listing for the FeedTankFeedFill task is shown in Table 4-7. Line numbers are displayed in the first column of the Table.

Table 4-7. Listing for FeedTankFeedFill Task

```
1            Task FeedTankFeedFill Runs When Time == 0
2               FillStatus_ON as realparameter;
3               FillStatus_OFF as realparameter;
4               FillStatus_ON: 1;
5               FillStatus_OFF: 0;
6               If FeedTank.FeedFillStatus_Signal == 0 Then
7                  // Tank level goes from high to low
8                  Wait for FeedTank.Total_Level <= FeedTank.Low_Level_Limit;
9                  FeedTank.FeedFillStatus_Signal: FillStatus_ON;
10                 Restart When FeedTank.Total_Level > 0.9 *
                      FeedTank.WorkingSpace_Level;
11              Else
12                 // Tank level goes from low to high
13                 Wait for FeedTank.Total_Level >= FeedTank.WorkingSpace_Level;
14                 FeedTank.FeedFillStatus_Signal: FillStatus_OFF;
15                 Restart When FeedTank.Total_Level < 1.1 *
                      FeedTank.Low_Level_Limit;
16              EndIf
17           End
```

The task is activated at the beginning of the simulation (Line 1). Lines 2-5 define two parameters *FillStatus_ON* and *FillStatus_OFF*, and assign values to them. The Then branch (Lines 6-10) of the If-Else-EndIf conditional is for the non-feed-fill mode (*FeedTank.FeedFillStatus_Signal = 0*) while the Else branch (Lines 11-16) is for the feed-fill mode (*FeedTank.FeedFillStatus_Signal = 1*).

In the non-feed-fill mode when the tank level goes from high to low, no action is taken until the tank level becomes less than or equal to *Low_Level_Limit* (Line 8), at which *FeedFillStatus_Signal* is turned on (Line 9), indicating that the feed tank now can accept feed transfers.

Upon switching from non-feed-fill to feed-fill mode, when the tank level exceeds (*0.9 * WorkingSpace_Level*), the task is restarted and then subsequently proceeds to the Else branch (Line 11). No action is taken until the tank level becomes larger than or equal to *WorkingSpace_Level*, at which *FeedFillStatus_Signal* is turned off (Line 14). On Line 15, the task is restarted when the tank level becomes less than (*1.1 * Low_Level_Limit*).

4.15.4 FeedTankRecycleFill Task

The FeedTankRecycleFill task triggers the *RecycleFillStatus_Signal* signal in the feed tank based on the tank level. An activated *RecycleFillStatus_Signal* indicates that the feed tank is ready to accept supernate recycle from the drop tank. Listing for the FeedTankRecycleFill task is shown in Table 4-8. Line numbers are displayed in the first column of the Table.

Table 4-8. Listing for FeedTankRecycleFill Task

```
1            Task FeedTankRecycleFill Runs When Time == 0
2               FillStatus_ON as realparameter;
3               FillStatus_OFF as realparameter;
4               FillStatus_ON: 1;
5               FillStatus_OFF: 0;
6               If FeedTank.RecycleFillStatus_Signal == 0 Then
7                  // Tank level goes from high to low
8                  Wait for FeedTank.Total_Level <= FeedTank.WorkingSpace_Level;
9                  FeedTank.RecycleFillStatus_Signal: FillStatus_ON;
10                 Restart When FeedTank.Total_Level > 0.9 *
                      FeedTank.High_Level_Limit;
11              Else
12                 // Tank level goes from low to high
13                 Wait for FeedTank.Total_Level >= FeedTank.High_Level_Limit;
14                 FeedTank.RecycleFillStatus_Signal: FillStatus_OFF;
15                 Restart When (DropTank.Total_Level >=
                      DropTank.High_Decant_Level OR
                         FeedTank.Total_Level < 1.01 * FeedTank.Low_Level_Limit) ;
16              EndIf
17           End
```

The task is activated at time 0 of the simulation (Line 1). Lines 2-5 define *FillStatus_ON* and *FillStatus_OFF* and assign values to them. The Then branch (Lines 6-10) of the If-Else-EndIf conditional is for the non-recycle-fill mode (*FeedTank.RecycleFillStatus_Signal = 0*) while the Else branch (Lines 11-16) is for the recycle-fill mode (*FeedTank.RecycleFillStatus_Signal = 1*).

In the non-recycle-fill mode when the tank level goes from high to low, no action is taken until the tank level becomes less than or equal to *WorkingSpace_Level* (Line 8), at which *FeedRecycleFillStatus_Signal* is turned on (Line 9), indicating that the feed tank is now ready to accept feed transfers.

Upon switching from non-recycle-fill to recycle-fill mode, when the tank level exceeds (*0.9 * High_Level_Limit*), the task is restarted and then subsequently proceeds to the Else branch (Line 11). No action is taken until the tank level becomes larger than or equal to

High_Level_Limit, at which RecycleFillStatus_Signal is turned (Line 14). On Line 15, the task is restarted when the tank level EITHER becomes larger than and equal to High_Decant_Level OR less than (*1.01 * FeedTank.Low_Level_Limit*).

4.15.5 RecycleSchedule Task

The RecycleSchedule task carries out the current operating logic that requires all feed transfers to the feed tank be turned off during the recycle of supernate from the drop tank. The RecycleSchedule task is listed in Table 4-9. Line numbers are displayed in the first column of the Table.

Table 4-9. Listing for RecycleSchedule Task

```
1            Task RecycleSchedule runs when Time == 0
2               FeedSignal_ON as realparameter;
3               FeedSignal_OFF as realparameter;
4               FeedSignal_ON: 1;
5               FeedSignal_OFF: 0;
6               SystemFeed.Feed_Signal: FeedSignal_OFF;
7               Canyon.Feed_Signal: FeedSignal_OFF;
8               OrderedFeed.Feed_Signal: FeedSignal_OFF;
9               wait for Rec_JD.OutPort.FV < 0.5;
10              SystemFeed.Feed_Signal: FeedSignal_ON;
11              Canyon.Feed_Signal: FeedSignal_ON;
12              OrderedFeed.Feed_Signal: FeedSignal_ON;
13              restart when Rec_JD.OutPort.FV > 0.5;
14           End
```

The task is activated at time 0 of the simulation (Line 1). Lines 2-5 define *FeedSignal_ON* and *FeedSignal_OFF* and assign values to them. *Feed_Signal* are turned off for feed streams *SystemFeed*, *Canyon*, and *OrderedFeed* on Lines 6, 7, and 8, respectively. No task action is taken until the recycle stream *Rec_JD.OutPort.FV* is off (Line 9). On Lines 10, 11, and 12, *Feed_Signal* are turned back on for all feed streams. The task is restarted when the recycle stream *Rec_JD.OutPort.FV* is on (Line 13).

4.15.6 OrderedTransfer*i* Tasks

The OrderedTransfer*i* tasks are used to specify the unscheduled feed transfers via the *OrderedFeed* stream. *i* is the sequential order number of this type of transfer (e.g., OrderedTransfer1, OrderedTransfer2, etc.). These tasks are created "on-the-fly" by the GUI I/O based on user specifications. Table 4-10 shows a sample of *OrderedTransfer1*. Line numbers are displayed in the first column.

Table 4-10. Listing for OrderedTransfer1 Task

```
1            Task OrderedTransfer1 Runs Once When Time == 0
2              FeedSignal_ON as realparameter;
3              FeedSignal_OFF as realparameter;
4              FeedSignal_ON: 1;
5              FeedSignal_OFF: 0;
6             Wait For OrderedFeed.TaskNum == 1;
7             Wait For FeedTank.Total_Level <= FeedTank.Low_Level_Limit;
```

```
8               OrderedFeed.FV_Feed: 24.983717;
9               OrdFd_JD.Dilution_Factor: 5;
10              OrderedFeed.T: 30;
11              OrderedFeed.conc("H2O"): 50.9653278438005;
12              OrderedFeed.conc("NaOH"): 3.01477887922097;
13              OrderedFeed.conc("NaNO3"): 1.30364213252336;
14              OrderedFeed.conc("NaNO2"): 0.808000413817542;
15              OrderedFeed.conc("Na2CO3"): 7.98614171259721E-02;
16              OrderedFeed.conc("Na2SO4"): 1.98655868766428E-02;
17              OrderedFeed.conc("Na3PO4"): 6.08945373299894E-03;
18              OrderedFeed.conc("NaCl"): 3.89325719531053E-03;
19              OrderedFeed.conc("NaF"): 5.49049122939315E-03;
20              OrderedFeed.conc("NaAlO2"): 8.98444016479622E-02;
21              OrderedFeed.conc("Na2SiO3"): 1.19792538419368E-03;
22              OrderedFeed.radconc("CsNO3"): 0;
23              OrderedFeed.radconc("PuO2"): 0;
24              If (SysFd_JD.OutPort.FV + Canyon_JD.OutPort.FV +
                   Rec_JD.OutPort.FV) < 0.5 Then
25                OrderedFeed.Feed_Signal: FeedSignal_ON;
26              Else
27                OrderedFeed.Feed_Signal: FeedSignal_OFF;
28              EndIf
29              OrderedFeed.Volume: 0.0;
30             Wait For OrderedFeed.Volume >= 378.5412;
31              OrderedFeed.Feed_Signal: FeedSignal_OFF;
32              OrderedFeed.FV_Feed: 0;
33              OrderedFeed.TaskNum: 2;
34            End
```

The task is activated at time 0 of the simulation (Line 1). Lines 2-5 define and assign values to parameters *FeedSignal_ON* and *FeedSignal_OFF*. The task does not proceed until *TaskNum* assumes a correct value (Line 6). This statement ensures that all OrderedTransfer tasks be carried out in sequential order. *TaskNum* must be 1 for *OrderedTransfer1*, 2 for *OrderedTransfer2* and so on. At the beginning of the simulation, the default value for *TaskNum* is 1, which will be increased to 2 at the end of the OrderedTransfer1 task (Line 33). Hence, if more than one OrderedTransfer tasks are present, although all tasks are activated at time 0, *OrderedTransfer2* can not proceed until *OrderedTransfer1* is complete, *OrdererTransfer3* only proceeds upon completion of *OrderedTransfer2*, etc.

No transfer is made until the feed tank level becomes less than or equal to *Low_Level_Limit* (Line 7). The feed transfer characteristics (volumetric flow rate, steam jet dilution factor, feed temperature, and feed concentration) are specified based on user inputs (Lines 8-23). The If-Else-EndIf conditional on Lines 24-28 carries out the operating logic that the unscheduled feed transfers can only be made in the absence of recycle and scheduled feed transfers. Line 29 set the initial transfer volume to 0. When the cumulative transfer volume reaches a user-specified value (Line 30), the transfer is terminated by turning *Feed_Signal* of the *OrdererFeed* stream off (Line 31). Volumetric flow rate is set to 0 (Line 32) and TaskNum is increased to 2 (Line 33).

4.15.7 SchCanyonTransfer*j* Tasks

The SchCanyonTransfer*j* tasks are used to specify the scheduled canyon feed transfers via the *Canyon* stream. *j* is the sequential order number of this type of transfer (e.g., SchCanyonTransfer1, SchCanyonTransfer2, etc.). These tasks are created “on-the-fly” by

the GUI I/O based on user specifications. Table 4-11 shows a sample of *SchCanyonTransfer1*. Line numbers are displayed in the first column.

Table 4-11. Listing for SchCanyonTransfer1 Task

```
1            Task SchCanyonTransfer1 Runs Once When Time == 12
2              FeedSignal_ON as realparameter;
3              FeedSignal_OFF as realparameter;
4              FeedSignal_ON: 1;
5              FeedSignal_OFF: 0;
6             Wait For Canyon.TaskNum == 1;
7              Canyon.FV_Feed: 18.169976;
8              Canyon_JD.Dilution_Factor: 4;
9              Canyon.T: 30;
10             Canyon.conc("H2O"): 51.9448362558336;
11             Canyon.conc("NaOH"): 0.019971731728486;
12             Canyon.conc("NaNO3"): 0.768911668228871;
13             Canyon.conc("NaNO2"): 1.18831803269098;
14             Canyon.conc("Na2CO3"): 1.26820495922221;
15             Canyon.conc("Na2SO4"): 4.18407777889472E-02;
16             Canyon.conc("Na3PO4"): 1.39802121503888E-03;
17             Canyon.conc("NaCl"): 1.39802121502471E-03;
18             Canyon.conc("NaF"): 5.29250888661598E-04;
19             Canyon.conc("NaAlO2"): 9.98586583167687E-03;
20             Canyon.conc("Na2SiO3"): 1.84067533622857E-14;
21             Canyon.radconc("CsNO3"): 0;
22             Canyon.radconc("PuO2"): 0;
23             If DropTank.Drain_Signal < 0.5 Then
24               Canyon.Feed_Signal: FeedSignal_ON;
25             Else
26               Canyon.Feed_Signal: FeedSignal_OFF;
27             EndIf
28           Canyon.Volume: 0.0;
29             OrderedFeed.Feed_Signal: FeedSignal_OFF;
30            Wait For Canyon.Volume >= 189.2706;
31             Canyon.Feed_Signal: 0;
32             Canyon.FV_Feed: 0;
33             Canyon.TaskNum: 2;
34             OrderedFeed.Feed_Signal: FeedSignal_ON;
35           End
```

This task is activated at a user-specified time (Line 1). Lines 2-5 define and assign values to parameters *FeedSignal_ON* and *FeedSignal_OFF*. Similar to the OrderedTransfer tasks, the task does not proceed until *TaskNum* assumes a correct value (Line 6). This statement ensures that all SchCanyonTransfer tasks be carried out in sequential order. A detailed discussion of the use of *TaskNum* feature is given in the subsection of OrderedTransfer*i* Tasks.

The canyon feed characteristics (volumetric flow rate, steam jet dilution factor, feed temperature, and feed concentration) are specified based on user inputs (Lines 7-22). The logic of the If-Else-EndIf conditional on Lines 23-27 ensures that the scheduled canyon feed can only be transferred to the feed tank when the recycle stream is not on. Line 28 set the initial transfer volume to 0. Line 29 turns the unscheduled feed transfer off. When the cumulative transfer volume reaches a user-specified value (Line 30), the transfer is terminated by turning *Feed_Signal* of the *Canyon* stream off (Line 31). Volumetric flow rate

is set to 0 (Line 32) and *TaskNum* is increased to 2 (Line 33). *Feed_Signal* of the *OrderedFeed* stream is turned back on (Line 34).

4.15.8 SchTankTransfer*k* Tasks

The SchTankTransfer*k* tasks are used to specify the scheduled system (tank) feed transfers via the *SystemFeed* stream. *k* is the sequential order number of this type of transfer (e.g., SchTankTransfer1, SchTankTransfer2, etc.). These tasks are created "on-the-fly" by the GUI I/O based on user specifications. Table 4-12 shows a sample of SchTankTransfer1. Line numbers are displayed in the first column.

Table 4-12. Listing for SchTankTransfer1 Task

```
1            Task SchTankTransfer1 Runs Once When Time == 30
2              FeedSignal_ON as realparameter;
3              FeedSignal_OFF as realparameter;
4              FeedSignal_ON: 1;
5              FeedSignal_OFF: 0;
6             Wait For SystemFeed.TaskNum == 1;
7              SystemFeed.FV_Feed: 27.254964;
8              SysFd_JD.Dilution_Factor: 5;
9              SystemFeed.T: 35;
10             SystemFeed.conc("H2O"): 51.8597210008829;
11             SystemFeed.conc("NaOH"): 1.99390066462564E-02;
12             SystemFeed.conc("NaNO3"): 0.767651752568466;
13             SystemFeed.conc("NaNO2"): 1.18637089030676;
14             SystemFeed.conc("Na2CO3"): 1.2661269165097;
15             SystemFeed.conc("Na2SO4"): 4.17722187419746E-02;
16             SystemFeed.conc("Na3PO4"): 1.39573045929256E-03;
17             SystemFeed.conc("NaCl"): 1.39573045927841E-03;
18             SystemFeed.conc("NaF"): 5.28383673986024E-04;
19             SystemFeed.conc("NaAlO2"): 9.9695032906154E-03;
20             SystemFeed.conc("Na2SiO3"): 1.83765926067964E-14;
21             SystemFeed.radconc("CsNO3"): 0;
22             SystemFeed.radconc("PuO2"): 0;
23             If DropTank.Drain_Signal < 0.5 Then
24               SystemFeed.Feed_Signal: FeedSignal_ON;
25             Else
26               SystemFeed.Feed_Signal: FeedSignal_OFF;
27             EndIf
28             SystemFeed.Volume: 0.0;
29             OrderedFeed.Feed_Signal: FeedSignal_OFF;
30            Wait For SystemFeed.Volume >= 378.5412;
31             SystemFeed.Feed_Signal: 0;
32             SystemFeed.FV_Feed: 0;
33             SystemFeed.TaskNum: 2;
34             OrderedFeed.Feed_Signal: FeedSignal_ON;
35           End
```

The structure of this task is identical to the SchCanyonTransfer tasks. The task is activated at a user-specified time (Line 1). Lines 2-5 defined and assign values to parameters *FeedSignal_ON* and *FeedSignal_OFF*. The task does not proceed until *TaskNum* assumes a correct value (Line 6).

The system characteristics (volumetric flow rate, steam jet dilution factor, feed temperature, and feed concentration) are specified based on user inputs (Lines 7-22). The If-Else-EndIf

conditional on Lines 23-27 ensures that the scheduled system feed transfer to the feed tank is made only when the recycle stream is not on. Line 28 set the initial transfer volume to 0. Line 29 turns the unscheduled feed transfer off. When the cumulative transfer volume reaches a user-specified value (Line 30), the transfer is terminated by turning *Feed_Signal* of the *SystemFeed* stream off (Line 31). Volumetric flow rate is set to 0 (Line 32) and *TaskNum* is increased to 2 (Line 33). *Feed_Signal* of the *OrderedFeed* stream is turned back on (Line 34).

5. REFERENCES

1. d'Entremont, P. D. *Functions and Requirements for the 2H and 3H Evaporator Flowsheet Models.* HLW-STE-2001-0003, Savannah River Site, Aiken, SC 29808 (2001).

2. Koffman, L. D. *User Guide for the ACM Evaporator Model User Interface*. WSRC-RP-2002-00324, Savannah River Site, Aiken, SC 29808 (2002).

3. Gorensek, M. B. *Test and Verification Report – HLW Evaporator Flowsheet Models*. Z-VVR-A-00001, Rev. 0, Savannah River Site, Aiken, SC 29808 (2002).

4. Aull, J. E., R. A. Dimenna, M. V. Gregory, T. Hang, P. K. Paul, K. L. Shanahan, and F. G. Smith, III. *A Description of the High Level Waste Integrated Flowsheet Model (HLWIFM)*. WSRC-RP-94-567, Savannah River Site, Aiken, SC 29808 (1994).

5. Smith, F. G., III. "Modeling of Batch Operations in the Defense Waste Processing Facility at the Savannah River Site ". *Proceedings of the 1995 Simulation MultiConference*, Phoenix, AZ, pp 207-212, (1995).

6. Dimenna, R. A., M. V. Gregory, T. Hang, P. K. Paul, F. G. Smith, III, and G. A. Taylor. *Computational Analysis of the SRS Phase III Salt Disposition Alternatives.* WSRC-TR-98-00459, Savannah River Site, Aiken, SC 29808 (1999).

7. Gregory, M. V., T. Hang, and R. A. Dimenna. *Verification Report: Conversion of HLW SPEEDUP Models to Aspen Custom Modeler*. WSRC-TR-2000-00385, Savannah River Site, Aiken, SC 29808 (2000).

8. Hang, T., and D. D. Walker. *Gas Generation and Bubble Formation Model for Crystalline Silicotitanate Ion Exchange Columns*. WSRC-TR-2000-00177, Rev. 0, Savannah River Site, Aiken, SC 29808 (2000).

9. Aleman, S. E. *Calculation of the Mixture Density of Binary Aqueous and Simulated Waste Salt Solutions using Aspen Plus® ElecNRTL and Aspen OLI™ Property Methods.* SRT-EMS-2002-00002, Savannah River Site, Aiken, SC 29808 (2002).

10. Askew, N. M. *Heat Transfer Model Description.* SRT-ATS-2002-00021, Savannah River Site, Aiken, SC 29808 (2002).

11. *Replacement High Level Waste Evaporator - System Design Description/Evaporator System.* G-SYD-H-00026, Rev. 2, Savannah River Site, Aiken, SC 29808 (2002).

12. *Chapter S: 3H Evaporator, Tank and Sump Volumes (November 12, 1999, Rev. 8).* http://shrine.srs.gov/html/hlwmd/wmtc/WMTC.html ("Waste Management Tank Calibrations"), Savannah River Site, Aiken, SC 29808.

13. *Study Guide: CST Operator Training Program – 3H Evaporator Systems.* WHAISC02, Rev. 0, Savannah River Site, Aiken, SC 29808.

14. *Introduction to Aspen Custom Modeler™ 10.1 - Course Notes.* Aspen Technology, Inc., October, 1999.

APPENDIX A. 3H EVAPORATOR STEAM LIFT PERFORMANCE

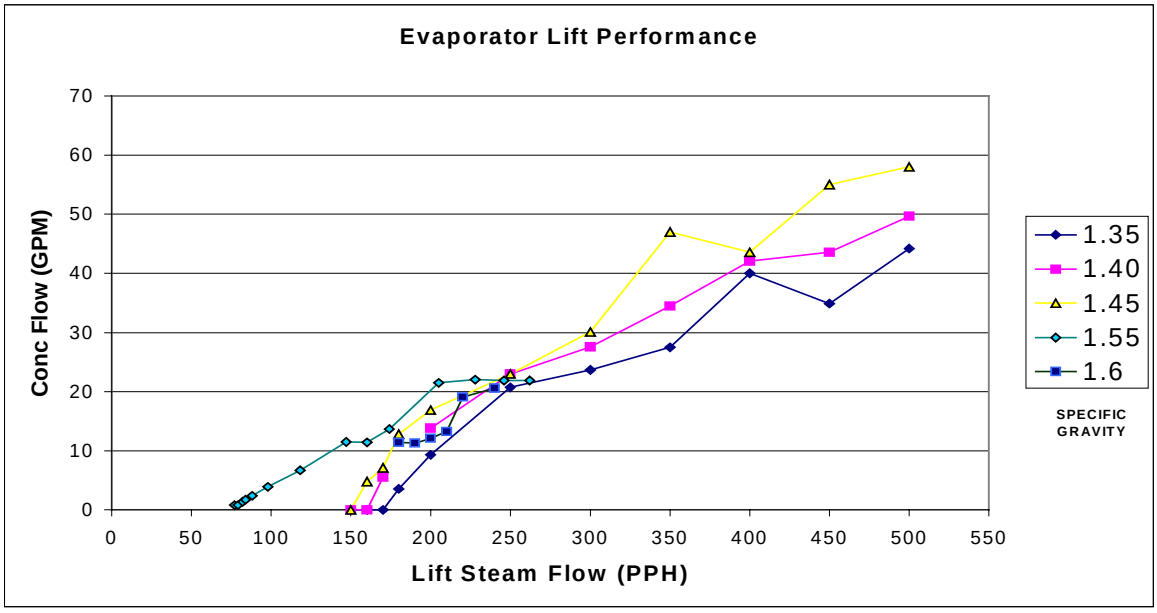


Figure A-1. Steam Lift Performance for the 3H Evaporator System (Ref. 11)

APPENDIX B. 3H EVAPORATOR CALIBRATION

Table B-1. 3H Evaporator Pot Volume Calibration (Ref. 12)

Volume (Gallons)	Corrected Liquid Level (Inches)
2060	1.03
2110	2.43
2160	3.33
2210	4.23
2260	5.03
2310	5.83
2360	6.63
2460	8.23
2560	9.43
2660	11.33
2760	12.83
2860	14.03
2960	15.33
3060	16.83
3160	18.03
3260	19.43
3360	20.63
3460	21.83
3560	23.23
3660	24.43
3760	25.53
3860	26.83
3960	27.83
4060	29.03
4160	30.23
4260	31.33
4360	32.33
4460	33.33
4560	34.43
4660	35.53
4760	36.63
4860	37.53
4960	38.73
5060	39.93
5360	43.13
5660	46.53

5960	49.73
6260	52.93
6560	56.13
6860	59.33
7160	62.63
7460	66.13
7760	69.73
7860	71.33
7960	72.53
8060	73.83
8160	75.23
8260	76.53
8360	77.83
8460	79.13
8560	80.43
8660	81.63
8760	82.83
9060	86.13
9360	89.43
9660	92.53
9960	95.73
10260	99.03
10560	102.13
10860	105.43
11117	108.43

APPENDIX C. SAMPLE CALCULATION

For demonstration a sample run of the 3H Evaporator Flowsheet model is provided in this appendix. The inputs and simulation results are shown. The case presented here corresponds to the following operational sequence:

1. ***Initial tank volumes:*** 400,000 gals supernate in the feed tank and 1,000,000 gals concentrated supernate in the drop tank.
2. ***Scheduled feed transfers:*** 50,000 gals from Canyon (on 2/15/2002 at 12PM), and 200,000 gals from Tank 49 (on 2/28/2002 at 6AM).
3. ***Unscheduled feed transfers:*** 600,000 gals from Tank 50, and 1,000,000 gals from Tank 32.
4. ***Recycle:*** Supernate in the drop tank is recycled to the feed tank whenever the drop tank is full.
5. ***SpG control:*** Concentrate in the evaporator pot is maintained at specific gravity of 1.5
6. ***Process time:*** 1000 hours (Start: 2/15/02 at 0:00 hr; End: 3/28/02 at 16:00 hr)

The input parameters/variables of the run are listed in Table C-1. This table is reproduced from Table Specifications in the ACM Flowsheet. Table C-2 provides a listing of all "on-the-fly" tasks generated by the GUI I/O based on user specifications. The simulation results are shown in Figures C-1 (feed tank), C-2 (evaporator pot), C-3 (separator pot), and C-4 (drop tank).

The results show that the model performs as expected. First, 50,000 gals of the canyon feed are transferred to the feed tank at 12 hours (activated by *Task SchCanyonTransfer1*). At ~59 hours, when the feed tank level reaches the low level operating limit set at 2.54 m, the unscheduled feed transfer from Tank 50 (activated by *Task OrderedTransfer1*) is made since there is no feed transfer scheduled at this time. At 78 hours, *Task SchTankTransfer1* activates the schedule to transfer 200,000 gals of waste from Tank 49. The unscheduled feed transfer is turned off during the scheduled transfer of Tank 49 waste. Upon completion of Tank 49 waste transfer the unscheduled feed transfer resumes its operation. At ~540 hours, another unscheduled feed transfer from Tank 32 is activated by *Task OrderedTransfer2*. This event is triggered by the feed tank level reaching the low level operating limit in the absence of scheduled feed transfers. At ~675 hours, the drop tank is full triggering the recycle of the concentrated supernate (activated by *Task DropTankDrain*). The onset of the recycle stream turns all feed transfers off (activated by *Task RecycleSchedule*). Finally, the recycle stream is terminated when the feed tank level reaches the high level operating limit at 9.144 m (activated by *Task FeedTankRecycleFill*). During the entire operation, the concentrate in the evaporator pot is maintained at specific gravity ~1.5. Specific gravity is controlled within 0.3% (no recycle) and within 1.33% of the setpoint (with recycle).

Table C-1. Input Parameters/Variables

	Value	Units	Spec	Description
Accumulator.FV	0	m3/hr	Fixed	Nominal volumetric outflow
Accumulator.P	1.0133	Bar	Fixed	Tank pressure
Accumulator.RadMoles("CSNO3")	0	Kmol	Initial	Rad holdup moles
Accumulator.RadMoles("PUO2")	0	Kmol	Initial	Rad holdup moles
Accumulator.T	30	C	Fixed	Tank temperature
Accumulator.Volume	1.00E-06	m3	Initial	Holdup volume
Accumulator.z("H2O")	1	kmol/kmol	Initial	Holdup mole fraction
Accumulator.z("NA2CO3")	0	kmol/kmol	Initial	Holdup mole fraction
Accumulator.z("NA2SIO3")	0	kmol/kmol	Initial	Holdup mole fraction
Accumulator.z("NA2SO4")	0	kmol/kmol	Initial	Holdup mole fraction
Accumulator.z("NA3PO4")	0	kmol/kmol	Initial	Holdup mole fraction
Accumulator.z("NAALO2")	0	kmol/kmol	Initial	Holdup mole fraction
Accumulator.z("NACL")	0	kmol/kmol	Initial	Holdup mole fraction
Accumulator.z("NAF")	0	kmol/kmol	Initial	Holdup mole fraction
Accumulator.z("NANO2")	0	kmol/kmol	Initial	Holdup mole fraction
Accumulator.z("NANO3")	0	kmol/kmol	Initial	Holdup mole fraction
Accumulator.z("NAOH")	0	kmol/kmol	Initial	Holdup mole fraction
Condenser.FV	50	m3/hr	Fixed	Volumetric flow of cooling water
Condenser.P_drop	0	bar	Fixed	Pressure drop in tubes
Condenser.P_in	1	bar	Fixed	Inlet pressure of cooling water
Condenser.T_in	30	C	Fixed	Inlet temperature of cooling water
Canyon.conc("H2O")	51.9448	mol/L	Fixed	Component concentration
Canyon.conc("NA2CO3")	1.2682	mol/L	Fixed	Component concentration
Canyon.conc("NA2SIO3")	1.84E-14	mol/L	Fixed	Component concentration
Canyon.conc("NA2SO4")	0.0418408	mol/L	Fixed	Component concentration
Canyon.conc("NA3PO4")	0.00139802	mol/L	Fixed	Component concentration
Canyon.conc("NAALO2")	0.00998587	mol/L	Fixed	Component concentration
Canyon.conc("NACL")	0.00139802	mol/L	Fixed	Component concentration
Canyon.conc("NAF")	5.29E-04	mol/L	Fixed	Component concentration
Canyon.conc("NANO2")	1.18832	mol/L	Fixed	Component concentration
Canyon.conc("NANO3")	0.768912	mol/L	Fixed	Component concentration
Canyon.conc("NAOH")	0.0199717	mol/L	Fixed	Component concentration
Canyon.Feed_Signal	1		Fixed	ON/OFF Feed signal
Canyon.FV_Feed	0	m3/hr	Fixed	Nominal volumetric flow
Canyon.P	1.0133	bar	Fixed	Pressure
Canyon.radconc("CSNO3")	0	mol/L	Fixed	Rad concentration
Canyon.radconc("PUO2")	0	mol/L	Fixed	Rad concentration
Canyon.T	30	C	Fixed	Temperature
Canyon.Volume	199.87	m3	Initial	Holdup volume
CanyonJD_Steam.P	11.356	bar	Fixed	Pressure
CanyonJD_Steam.Steam_Signal	1		Fixed	ON/OFF steam signal
Canyon_JD.dilution_factor	4		Fixed	Dilution factor, %
Canyon_JD.Jet_ON	1		Fixed	JD ON/OFF flag
DropTank.Boildownratio_value	0.95		Fixed	Set value of boildown ratio
DropTank.Bundle_Area	42.96394	m2	Fixed	Area per coil bundle
DropTank.Coil_number	5			Number of active cooling coils
DropTank.Drain_Signal	1		Fixed	ON/OFF Drain signal

DropTank.FV_Cooling	2.3848094	m3/hr	Fixed	Volumetric flow per cooling bundle
DropTank.FV_Decant	27.254964	m3/hr	Fixed	Nominal decant flow
DropTank.HeatTransCoef	0.03774	kW/m2/K	Fixed	Cooling heat transfer coefficient
DropTank.Heel_Level	1.9812	m	Fixed	Heel level
DropTank.High_Level_Limit	9.1948	m	Fixed	High level operating limit
DropTank.Min_Coil_Depth	0.2286	m	Fixed	Minimum coil depth
DropTank.Min_Supernate_Depth	0.1524	m	Fixed	Minimum supernate depth
DropTank.P	1.0133	bar	Fixed	Operating pressure
DropTank.RadMoles("CSNO3")	0.032605581	kmol	Initial	Holdup rad holdup moles
DropTank.RadMoles("PUO2")	0	kmol	Initial	Holdup rad holdup moles
DropTank.T	64.018037	C	Initial	Tank holdup temperature
DropTank.T_Cooling	33.058933	C	Initial	Average cooling water temperature
DropTank.T_Cooling_In	25	C	Fixed	Cooling water inlet temperature
DropTank.Tank_Factor	523.10221			m3/m
DropTank.Volume	4291.1597	m3	Initial	Tank holdup volume
DropTank.Volume_Cooling	0.489032	m3	Fixed	Cooling volume per coil bundle
DropTank.z("H2O")	0.7709223	kmol/kmol	Initial	Tank holdup mole fraction
DropTank.z("NA2CO3")	0.007075401	kmol/kmol	Initial	Tank holdup mole fraction
DropTank.z("NA2SIO3")	3.45E-05	kmol/kmol	Initial	Tank holdup mole fraction
DropTank.z("NA2SO4")	0.001116652	kmol/kmol	Initial	Tank holdup mole fraction
DropTank.z("NA3PO4")	2.46E-04	kmol/kmol	Initial	Tank holdup mole fraction
DropTank.z("NAALO2")	0.009759058	kmol/kmol	Initial	Tank holdup mole fraction
DropTank.z("NACL")	1.78E-04	kmol/kmol	Initial	Tank holdup mole fraction
DropTank.z("NAF")	2.35E-04	kmol/kmol	Initial	Tank holdup mole fraction
DropTank.z("NANO2")	0.045718164	kmol/kmol	Initial	Tank holdup mole fraction
DropTank.z("NANO3")	0.055779988	kmol/kmol	Initial	Tank holdup mole fraction
DropTank.z("NAOH")	0.10893532	kmol/kmol	Initial	Tank holdup mole fraction
Lance.FMass	566.9905	kg/hr	Fixed	Mass flow
Lance.P	2.7369427	bar	Fixed	Pressure
Lance.Steam_Signal	1		Fixed	ON/OFF steam signal
Lift.P	2.7369427	bar	Fixed	Pressure
Lift.Steam_Signal	1		Fixed	ON/OFF steam signal
Steam.P	23.421214	bar	Fixed	Pressure
Steam.Steam_Signal	1		Fixed	ON/OFF steam signal
Evaporator.FSteambundle_Manual	7500.0006	kg/hr	Fixed	Manually set tube bundle steam mass flow
Evaporator.FSteamBundle_Max	9071.848	kg/hr	Fixed	Maximum tube bundle steam mass flow
Evaporator.FSteamBundle_Min	0	kg/hr	Fixed	Minimum tube bundle steam mass flow
Evaporator.FSteamLift_Max	226.7962	kg/hr	Fixed	Maximum steam lift mass flow
Evaporator.FSteamLift_Min	29.483506	kg/hr	Fixed	Minimum steam lift mass flow
Evaporator.Liquid_Level	2.4129262	m	Initial	Concentrate level
Evaporator.LiquidLevel_SetPoint	2.413	m	Fixed	Concentrate level at level setpoint
Evaporator.P	1.0133	bar	Fixed	Operating pressure
Evaporator.P_drop	1.3789514	bar	Fixed	Pressure drop in tubes
Evaporator.RadMoles("CSNO3")	2.00E-04	kmol	Initial	Rad holdup moles
Evaporator.RadMoles("PUO2")	0	kmol	Initial	Rad holdup moles
Evaporator.SpG_SetPoint	1.5		Fixed	Specific gravity setpoint
Evaporator.SpGControl_ON	1			SpG control ON/OFF signal
Evaporator.T	138.84124	C	Initial	Solution temperature
Evaporator.z("H2O")	0.66842843	kmol/kmol	Initial	Holdup mole fraction
Evaporator.z("NA2CO3")	0.009830854	kmol/kmol	Initial	Holdup mole fraction

Evaporator.z("NA2SIO3")	6.09E-05	kmol/kmol	Initial	Holdup mole fraction
Evaporator.z("NA2SO4")	0.001588521	kmol/kmol	Initial	Holdup mole fraction
Evaporator.z("NA3PO4")	3.27E-04	kmol/kmol	Initial	Holdup mole fraction
Evaporator.z("NAALO2")	0.010000572	kmol/kmol	Initial	Holdup mole fraction
Evaporator.z("NACL")	2.53E-04	kmol/kmol	Initial	Holdup mole fraction
Evaporator.z("NAF")	3.43E-04	kmol/kmol	Initial	Holdup mole fraction
Evaporator.z("NANO2")	0.060558348	kmol/kmol	Initial	Holdup mole fraction
Evaporator.z("NANO3")	0.081698452	kmol/kmol	Initial	Holdup mole fraction
Evaporator.z("NAOH")	0.16691138	kmol/kmol	Initial	Holdup mole fraction
FeedTank.Bundle_Area	42.96394	m2	Fixed	Area per coil bundle
FeedTank.Coil_number	5			Number of active cooling coils
FeedTank.Drain_Signal	1		Fixed	ON/OFF Drain signal
FeedTank.FeedFillStatus_Signal	0		Fixed	Tank feed fill status signal
FeedTank.FV_Cooling	2.3848094	m3/hr	Fixed	Volumetric flow per cooling bundle
FeedTank.FV_Decant	6.813741	m3/hr	Fixed	Nominal supernate decant flow
FeedTank.HeatTransCoef	0.037739915	kW/m2/K	Fixed	Cooling heat transfer coefficient
FeedTank.Heel_Level	2.2606	m	Fixed	Heel level
FeedTank.High_Level_Limit	9.144	m	Fixed	High level operating limit
FeedTank.Low_Level_Limit	2.54	m	Fixed	Low level operating limit
FeedTank.Min_Coil_Depth	0.2286	m	Fixed	Minimum coil depth
FeedTank.Min_Supernate_Depth	0.1524	m	Fixed	Minimum supernate depth
FeedTank.P	1.0133	bar	Fixed	Operating pressure
FeedTank.RadMoles("CSNO3")	0.006222696	kmol	Initial	Holdup rad holdup moles
FeedTank.RadMoles("PUO2")	0	kmol	Initial	Holdup rad holdup moles
FeedTank.RecycleFillStatus_Signal	0		Fixed	Tank recycle fill status signal
FeedTank.T	53.692961	C	Initial	Tank holdup temperature
FeedTank.T_Cooling	29.244881	C	Initial	Average cooling water temperature
FeedTank.T_Cooling_In	25	C	Fixed	Cooling water inlet temperature
FeedTank.Tank_Factor	523.10221			m3/m
FeedTank.Volume	2885.2935	m3	Initial	Tank holdup volume
FeedTank.Volume_Cooling	0.489032	m3	Fixed	Cooling volume per coil bundle
FeedTank.WorkingSpace_Volume	378.5412	m3	Fixed	Working space volume
FeedTank.z("H2O")	0.88554	kmol/kmol	Initial	Tank holdup mole fraction
FeedTank.z("NA2CO3")	0.003393655	kmol/kmol	Initial	Tank holdup mole fraction
FeedTank.z("NA2SIO3")	2.10E-05	kmol/kmol	Initial	Tank holdup mole fraction
FeedTank.z("NA2SO4")	5.48E-04	kmol/kmol	Initial	Tank holdup mole fraction
FeedTank.z("NA3PO4")	1.13E-04	kmol/kmol	Initial	Tank holdup mole fraction
FeedTank.z("NAALO2")	0.003452243	kmol/kmol	Initial	Tank holdup mole fraction
FeedTank.z("NACL")	8.72E-05	kmol/kmol	Initial	Tank holdup mole fraction
FeedTank.z("NAF")	1.18E-04	kmol/kmol	Initial	Tank holdup mole fraction
FeedTank.z("NANO2")	0.020905015	kmol/kmol	Initial	Tank holdup mole fraction
FeedTank.z("NANO3")	0.028202674	kmol/kmol	Initial	Tank holdup mole fraction
FeedTank.z("NAOH")	0.057618559	kmol/kmol	Initial	Tank holdup mole fraction
OrderedFeed.conc("H2O")	50.9653	mol/L	Fixed	Component concentration
OrderedFeed.conc("NA2CO3")	0.0798614	mol/L	Fixed	Component concentration
OrderedFeed.conc("NA2SIO3")	0.00119793	mol/L	Fixed	Component concentration
OrderedFeed.conc("NA2SO4")	0.0198656	mol/L	Fixed	Component concentration
OrderedFeed.conc("NA3PO4")	0.00608945	mol/L	Fixed	Component concentration
OrderedFeed.conc("NAALO2")	0.0898444	mol/L	Fixed	Component concentration
OrderedFeed.conc("NACL")	0.00389326	mol/L	Fixed	Component concentration

OrderedFeed.conc("NAF")	0.00549049	mol/L	Fixed	Component concentration
OrderedFeed.conc("NANO2")	0.808	mol/L	Fixed	Component concentration
OrderedFeed.conc("NANO3")	1.30364	mol/L	Fixed	Component concentration
OrderedFeed.conc("NAOH")	3.01478	mol/L	Fixed	Component concentration
OrderedFeed.Feed_Signal	1		Fixed	ON/OFF Feed signal
OrderedFeed.FV_Feed	22.7125	m3/hr	Fixed	Nominal volumetric flow
OrderedFeed.P	1.0133	bar	Fixed	Pressure
OrderedFeed.radconc("CSNO3")	0	mol/L	Fixed	Rad concentration
OrderedFeed.radconc("PUO2")	0	mol/L	Fixed	Rad concentration
OrderedFeed.T	30	C	Fixed	Temperature
OrderedFeed.Volume	3134.5521	m3	Initial	Holdup volume
OrdFdJD_Steam.P	11.356	bar	Fixed	Pressure
OrdFdJD_Steam.Steam_Signal	1		Fixed	ON/OFF steam signal
OrdFd_JD.dilution_factor	4		Fixed	Dilution factor, %
OrdFd_JD.Jet_ON	1		Fixed	JD ON/OFF flag
RecJD_Steam.P	11.356	bar	Fixed	Pressure
RecJD_Steam.Steam_Signal	1		Fixed	ON/OFF steam signal
Rec_JD.dilution_factor	4		Fixed	Dilution factor, %
Rec_JD.Jet_ON	1		Fixed	JD ON/OFF flag
Separator.P	1.0133	bar	Fixed	Pressure
Separator.TRef	25	C	Fixed	Ambient temperature
Separator.UA	0		Fixed	Heat transfer coefficent * Area
SystemFeed.conc("H2O")	51.8597	mol/L	Fixed	Component concentration
SystemFeed.conc("NA2CO3")	1.26613	mol/L	Fixed	Component concentration
SystemFeed.conc("NA2SIO3")	1.84E-14	mol/L	Fixed	Component concentration
SystemFeed.conc("NA2SO4")	0.0417722	mol/L	Fixed	Component concentration
SystemFeed.conc("NA3PO4")	0.00139573	mol/L	Fixed	Component concentration
SystemFeed.conc("NAALO2")	0.0099695	mol/L	Fixed	Component concentration
SystemFeed.conc("NACL")	0.00139573	mol/L	Fixed	Component concentration
SystemFeed.conc("NAF")	5.28E-04	mol/L	Fixed	Component concentration
SystemFeed.conc("NANO2")	1.18637	mol/L	Fixed	Component concentration
SystemFeed.conc("NANO3")	0.767652	mol/L	Fixed	Component concentration
SystemFeed.conc("NAOH")	0.019939	mol/L	Fixed	Component concentration
SystemFeed.Feed_Signal	1		Fixed	ON/OFF Feed signal
SystemFeed.FV_Feed	0	m3/hr	Fixed	Nominal volumetric flow
SystemFeed.P	1.0133	bar	Fixed	Pressure
SystemFeed.radconc("CSNO3")	0	mol/L	Fixed	Rad concentration
SystemFeed.radconc("PUO2")	0	mol/L	Fixed	Rad concentration
SystemFeed.T	35	C	Fixed	Temperature
SystemFeed.Volume	763.14	m3	Initial	Holdup volume
SysFdJD_Steam.P	11.356	bar	Fixed	Pressure
SysFdJD_Steam.Steam_Signal	1		Fixed	ON/OFF steam signal
SysFd_JD.dilution_factor	5		Fixed	Dilution factor, %
SysFd_JD.Jet_ON	1		Fixed	JD ON/OFF flag

Table C-2. "On-The-Fly" Tasks Generated For The Sample Calculation

OrdererTransfer1 Task

```
1            Task OrderedTransfer1 Runs Once When Time == 0
2              FeedSignal_ON as realparameter;
3              FeedSignal_OFF as realparameter;
4              FeedSignal_ON: 1;
5              FeedSignal_OFF: 0;
6             Wait For OrderedFeed.TaskNum == 1;
7             Wait For FeedTank.Total_Level <= FeedTank.Low_Level_Limit;
8              OrderedFeed.FV_Feed: 24.983717;
9              OrdFd_JD.Dilution_Factor: 5;
10             OrderedFeed.T: 30;
11             OrderedFeed.conc("H2O"): 50.9653278438005;
12             OrderedFeed.conc("NaOH"): 3.01477887922097;
13             OrderedFeed.conc("NaNO3"): 1.30364213252336;
14             OrderedFeed.conc("NaNO2"): 0.808000413817542;
15             OrderedFeed.conc("Na2CO3"): 7.98614171259721E-02;
16             OrderedFeed.conc("Na2SO4"): 1.98655868766428E-02;
17             OrderedFeed.conc("Na3PO4"): 6.08945373299894E-03;
18             OrderedFeed.conc("NaCl"): 3.89325719531053E-03;
19             OrderedFeed.conc("NaF"): 5.49049122939315E-03;
20             OrderedFeed.conc("NaAlO2"): 8.98444016479622E-02;
21             OrderedFeed.conc("Na2SiO3"): 1.19792538419368E-03;
22             OrderedFeed.radconc("CsNO3"): 0;
23             OrderedFeed.radconc("PuO2"): 0;
24             If (SysFd_JD.OutPort.FV + Canyon_JD.OutPort.FV +
                    Rec_JD.OutPort.FV) < 0.5 Then
25               OrderedFeed.Feed_Signal: FeedSignal_ON;
26             Else
27               OrderedFeed.Feed_Signal: FeedSignal_OFF;
28             EndIf
29             OrderedFeed.Volume: 0.0;
30            Wait For OrderedFeed.Volume >= 2271.2472;
31             OrderedFeed.Feed_Signal: FeedSignal_OFF;
32             OrderedFeed.FV_Feed: 0;
33             OrderedFeed.TaskNum: 2;
34           End
```

OrdererTransfer2 Task

```
1            Task OrderedTransfer2 Runs Once When Time == 0
2              FeedSignal_ON as realparameter;
3              FeedSignal_OFF as realparameter;
4              FeedSignal_ON: 1;
5              FeedSignal_OFF: 0;
6             Wait For OrderedFeed.TaskNum == 2;
7             Wait For FeedTank.Total_Level <= FeedTank.Low_Level_Limit;
8              OrderedFeed.FV_Feed: 22.71247;
9              OrdFd_JD.Dilution_Factor: 4;
10             OrderedFeed.T: 30;
11             OrderedFeed.conc("H2O"): 50.9653278438005;
12             OrderedFeed.conc("NaOH"): 3.01477887922097;
13             OrderedFeed.conc("NaNO3"): 1.30364213252336;
```

```
14              OrderedFeed.conc("NaNO2"): 0.808000413817542;
15              OrderedFeed.conc("Na2CO3"): 7.98614171259721E-02;
16              OrderedFeed.conc("Na2SO4"): 1.98655868766428E-02;
17              OrderedFeed.conc("Na3PO4"): 6.08945373299894E-03;
18              OrderedFeed.conc("NaCl"): 3.89325719531053E-03;
19              OrderedFeed.conc("NaF"): 5.49049122939315E-03;
20              OrderedFeed.conc("NaAlO2"): 8.98444016479622E-02;
21              OrderedFeed.conc("Na2SiO3"): 1.19792538419368E-03;
22              OrderedFeed.radconc("CsNO3"): 0;
23              OrderedFeed.radconc("PuO2"): 0;
24              If (SysFd_JD.OutPort.FV + Canyon_JD.OutPort.FV +
                     Rec_JD.OutPort.FV) < 0.5 Then
25                OrderedFeed.Feed_Signal: FeedSignal_ON;
26              Else
27                OrderedFeed.Feed_Signal: FeedSignal_OFF;
28              EndIf
29              OrderedFeed.Volume: 0.0;
30             Wait For OrderedFeed.Volume >= 3785.412;
31              OrderedFeed.Feed_Signal: FeedSignal_OFF;
32              OrderedFeed.FV_Feed: 0;
33              OrderedFeed.TaskNum: 3;
34            End
```

SchCanyonTransfer1 Task

```
1             Task SchCanyonTransfer1 Runs Once When Time == 12
2               FeedSignal_ON as realparameter;
3               FeedSignal_OFF as realparameter;
4               FeedSignal_ON: 1;
5               FeedSignal_OFF: 0;
6              Wait For Canyon.TaskNum == 1;
7               Canyon.FV_Feed: 18.169976;
8               Canyon_JD.Dilution_Factor: 4;
9               Canyon.T: 30;
10              Canyon.conc("H2O"): 51.9448362558336;
11              Canyon.conc("NaOH"): 0.019971731728486;
12              Canyon.conc("NaNO3"): 0.768911668228871;
13              Canyon.conc("NaNO2"): 1.18831803269098;
14              Canyon.conc("Na2CO3"): 1.26820495922221;
15              Canyon.conc("Na2SO4"): 4.18407777889472E-02;
16              Canyon.conc("Na3PO4"): 1.39802121503888E-03;
17              Canyon.conc("NaCl"): 1.39802121502471E-03;
18              Canyon.conc("NaF"): 5.29250888661598E-04;
19              Canyon.conc("NaAlO2"): 9.98586583167687E-03;
20              Canyon.conc("Na2SiO3"): 1.84067533622857E-14;
21              Canyon.radconc("CsNO3"): 0;
22              Canyon.radconc("PuO2"): 0;
23              If DropTank.Drain_Signal < 0.5 Then
24                Canyon.Feed_Signal: FeedSignal_ON;
25              Else
26                Canyon.Feed_Signal: FeedSignal_OFF;
27              EndIf
28              Canyon.Volume: 0.0;
```

```
29              OrderedFeed.Feed_Signal: FeedSignal_OFF;
30             Wait For Canyon.Volume >= 189.2706;
31              Canyon.Feed_Signal: 0;
32              Canyon.FV_Feed: 0;
33              Canyon.TaskNum: 2;
34              OrderedFeed.Feed_Signal: FeedSignal_ON;
35            End
```

SchTankTransfer1 Task

```
1             Task SchTankTransfer1 Runs Once When Time == 78
2               FeedSignal_ON as realparameter;
3               FeedSignal_OFF as realparameter;
4               FeedSignal_ON: 1;
5               FeedSignal_OFF: 0;
6              Wait For SystemFeed.TaskNum == 1;
7               SystemFeed.FV_Feed: 27.254964;
8               SysFd_JD.Dilution_Factor: 5;
9               SystemFeed.T: 35;
10              SystemFeed.conc("H2O"): 51.8597210008829;
11              SystemFeed.conc("NaOH"): 1.99390066462564E-02;
12              SystemFeed.conc("NaNO3"): 0.767651752568466;
13              SystemFeed.conc("NaNO2"): 1.18637089030676;
14              SystemFeed.conc("Na2CO3"): 1.2661269165097;
15              SystemFeed.conc("Na2SO4"): 4.17722187419746E-02;
16              SystemFeed.conc("Na3PO4"): 1.39573045929256E-03;
17              SystemFeed.conc("NaCl"): 1.39573045927841E-03;
18              SystemFeed.conc("NaF"): 5.28383673986024E-04;
19              SystemFeed.conc("NaAlO2"): 9.9695032906154E-03;
20              SystemFeed.conc("Na2SiO3"): 1.83765926067964E-14;
21              SystemFeed.radconc("CsNO3"): 0;
22              SystemFeed.radconc("PuO2"): 0;
23              If DropTank.Drain_Signal < 0.5 Then
24                SystemFeed.Feed_Signal: FeedSignal_ON;
25              Else
26                SystemFeed.Feed_Signal: FeedSignal_OFF;
27              EndIf
28              SystemFeed.Volume: 0.0;
29              OrderedFeed.Feed_Signal: FeedSignal_OFF;
30             Wait For SystemFeed.Volume >= 757.0824;
31              SystemFeed.Feed_Signal: 0;
32              SystemFeed.FV_Feed: 0;
33              SystemFeed.TaskNum: 2;
34              OrderedFeed.Feed_Signal: FeedSignal_ON;
35            End
```

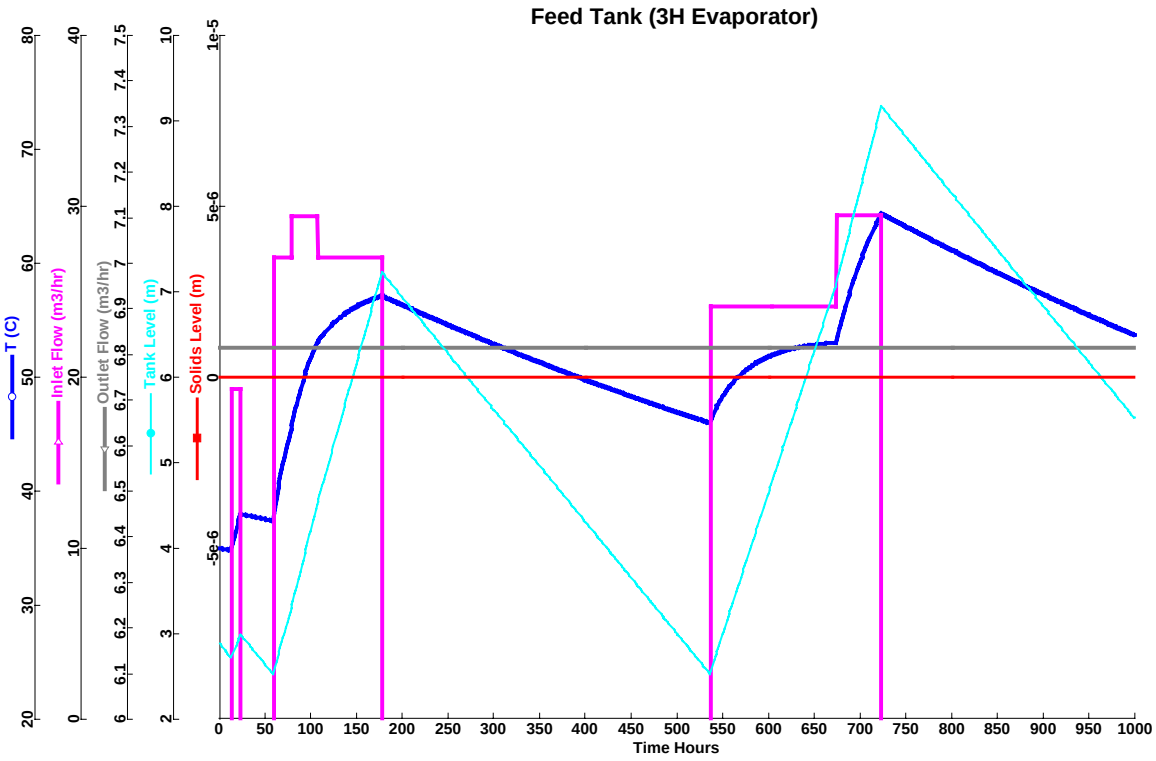


Figure C-1. Result of the Feed Tank for the Sample Run

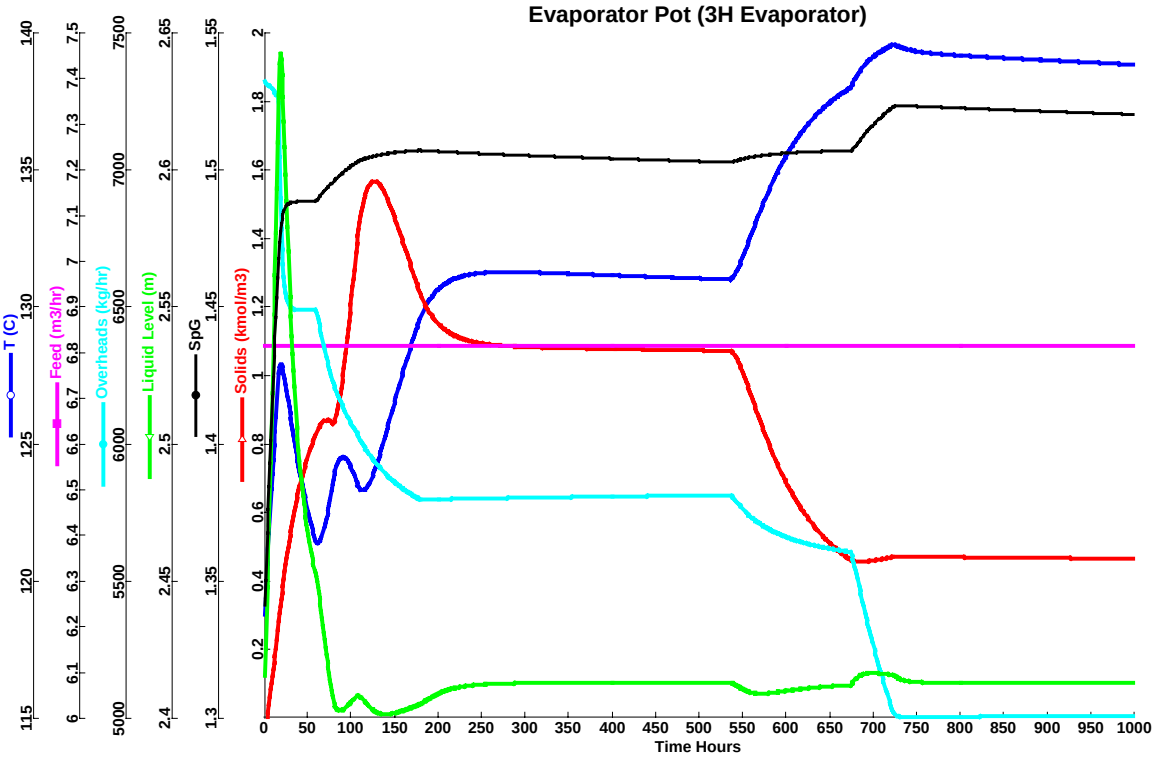


Figure C-2. Result of the Evaporator Pot for the Sample Run

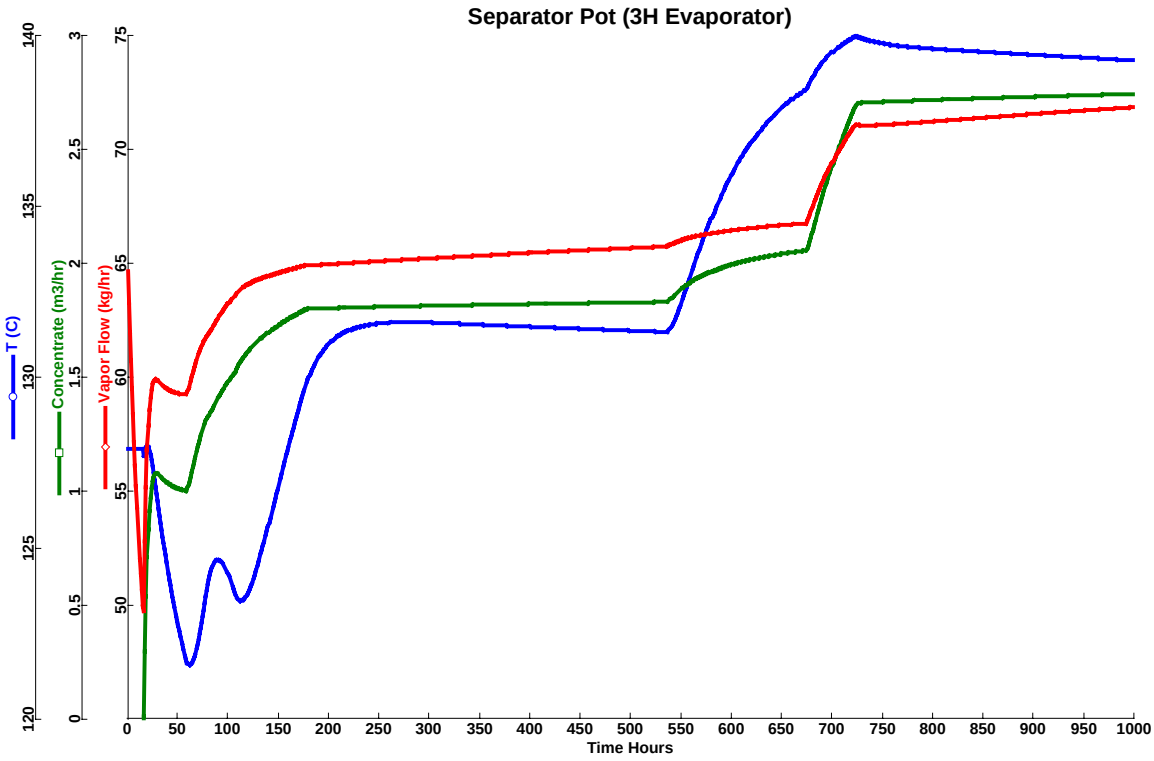


Figure C-3. Result of the Separator Pot for the Sample Run

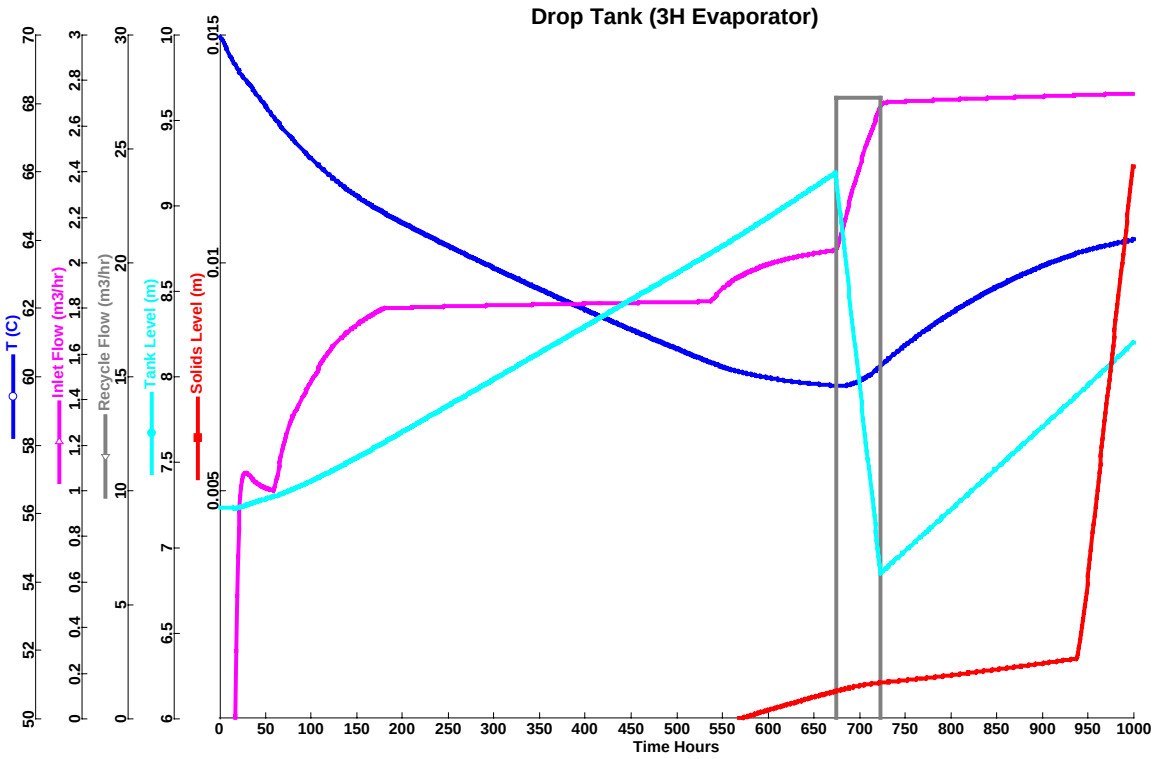


Figure C-4. Result of the Drop Tank for the Sample Run

APPENDIX D. PROGRAM LISTING

AccumulateTank Model

```
Model AccumulateTank
/*     WESTINGHOUSE SAVANNAH RIVER COMPANY
       INTEGRATED FLOWSHEET MODEL FOR THE HLW SYSTEM
       THE 2H & 3H EVAPORATOR FLOWSHEET MODELS

       Author:      T. Hang, SRTC
       Version Date: 28 February 2002
       Description:  This module models a simple tank for liquid accumulation

       Design Inputs/Outputs:
          Fixed Parameters/Variables
              T            Specified tank temperature, C
              P            Specified tank pressure, bar
              FV           Nominal volumetric outflow, m3/hr

          Initial Variables
              Volume       Holdup volume, m3
              z            Array of holdup mole fraction
              RadMoles     Array of Rads component holdup moles, kmol

          Input Streams
              InPort       Multiple liquid inlet streams (MultiPort)

          Output Streams
              OutPort            Multiple liquid outlet stream (MultiPort)

       Revision History:
              Rev No Date         Author       Description
              ------ ----------   --------------      ------------------------------------------------------------------------

*/

// PARAMETERS & VARIABLES

       MainSet             AS  ComponentListName("Default");       // Main (apparent) Component List
       RadSet              AS  ComponentListName("Rads");          // Rads Component List

       FV                  AS  flow_vol        (Description:"Nominal volumetric outflow", Units:"m3/hr", Fixed, 0);

       T                   As  temperature     (Description:"Tank temperature", Units:"C", Fixed, 30);
       P                   AS  pressure        (Description:"Tank pressure", Units:"bar", Fixed, 1.0133);
```

```
      DensMass            As  dens_mass             (Description:"Holdup mass density", Units:"kg/m3");
      Mass                AS  holdup_mass           (Description:"Holdup mass", Units:"kg", Upper:1E10);
      Moles               AS  holdup_mol            (Description:"Holdup moles", Units:"kmol", Upper:1E10);
      Volume                     AS  volume         (Description:"Holdup volume", Units:"m3", Initial, 1E-6);

      MW                  AS  molweight       (Description:"Holdup MW");
      h                   AS  enth_mol        (Description:"Holdup molar enthalpy", Units:"GJ/kmol");

      w(MainSet)          AS  massfraction    (Description:"Holdup mass fraction", Units:"kg/kg");
      z(MainSet)          AS  molefraction    (Description:"Holdup mole fraction", Units:"kmol/kmol", Initial);
      MWs(MainSet)        AS  HIDDEN molweight      (Description:"Component MW");

      RadMoles(RadSet)    AS  holdup_mol            (Description:"Rad holdup moles", Units:"kmol", Initial, 0);
      radconc(RadSet)     AS  conc_mole       (Description:"Rad concentration", Units:"mol/L");

// PORTS
      InPort                     AS INPUT MULTIPORT OF Main (Description:"Multiple liquid inlet streams");
      OutPort             as OUTPUT Main      (Description:"Multiple liquid outlet stream", ComponentList: MainSet);

// EQUATIONS

// Overall mole balance
EQ01:  $Moles = SIGMA(InPort.Connection.FMol) - OutPort.FMol;
EQ01a:  Moles * MW = Mass;
EQ01b: Volume * DensMass = Mass;

// Component mole balances
EQ02:  FOR i IN MainSet DO
         z(i)*$Moles + Moles*$z(i) =
            SIGMA(InPort.Connection.FMol*InPort.Connection.z(i)) - OutPort.FMol*OutPort.z(i);
       ENDFOR
EQ02a:  w * SIGMA(z * MWs) = z * MWs;

       // Rad components
EQ02b: FOR i IN RadSet DO
         $RadMoles(i) = SIGMA(InPort.Connection.FRad(i)) - OutPort.FRad(i);
       ENDFOR
EQ02c:  IF Volume > 0 THEN
         radconc * Volume = RadMoles;
       ELSE
         radconc = 0;
       ENDIF
```

```
// Output stream
EQ03:  OutPort.F = FV * DensMass;
EQ03a: OutPort.FMol * MW = OutPort.F;
EQ03b: OutPort.FV = FV;
EQ03c: OutPort.w = w;
EQ03d: OutPort.z = z;
EQ03e: OutPort.T = T;
EQ03f: OutPort.P = P;
EQ03g: OutPort.h = h;
EQ03h: OutPort.FRad  = radconc * FV;
EQ03i: OutPort.radconc = radconc;

// PROCEDURES
EQ20:  CALL(MWs) = pMolWeights() MainSet;                // Component MWs
EQ20a: CALL(MW) = pMolWeight(z);                         // Apparent-component MW
EQ20b: CALL(DensMass) = pDens_Mass_Liq(T, P, z);         // Liquid mass density
EQ20c: CALL(h) = pEnth_Mol_Liq(T, P, z);                 // Liquid molar enthalpy

End
```

Condenser Model

```
Model Condenser
/*     WESTINGHOUSE SAVANNAH RIVER COMPANY
       INTEGRATED FLOWSHEET MODEL FOR THE HLW SYSTEM
       THE 2H & 3H EVAPORATOR FLOWSHEET MODELS

       Author:      T. Hang, SRTC
       Version Date: 9 January 2002
       Description:  This module models the steady-state operation of a condenser

       Design Inputs/Outputs:
          Fixed Parameters/Variables
              FV           Volumetric flow of cooling water, m3/hr
              P_in         Inlet pressure of cooling water, bar
              P_drop       Pressure drop in cooling water tube bundle, bar
              T_in         Inlet temperature of cooling water, C

          Input Streams
              I1           Vapor inlet stream

          Output Streams
              O1           Condensate stream

       Revision History:
              Rev No Date         Author              Description
              ------ ----------   --------------      ------------------------------------------------------------------
              001    02/26/2002   T. Hang             Specify volume flow of cooling water; T_rise is calculated

*/

// PARAMETERS & VARIABLES

       GasSet              AS  ComponentListName("Vapor");         // Vapor Component List
       WaterSet            AS  ComponentListName ("Water");        // Water component set

       F                   AS  flow_mass       (Description:"Mass flow of cooling water", Units:"kg/hr");
       FV                  AS  flow_vol        (Description:"Volumetric flow of cooling water", Units:"m3/hr", Fixed,
                                                      50);         // Rev 001
       FMol                AS  flow_mol        (Description:"Mole flow of cooling water", Units:"kmol/hr");

       DensMass            AS  dens_mass       (Description:"Condensate density", Units:"kg/m3");
       DensMass_H2O        AS  dens_mass       (Description:"Cooling water density", Units:"kg/m3", Fixed,1000); // Rev 001
```

```
      P_in              AS  pressure       (Description:"Inlet pressure of cooling water", Units:"bar", Fixed);
      P_out             AS  pressure       (Description:"Outlet pressure of cooling water", Units:"bar");
      P_drop            AS  press_drop     (Description:"Pressure drop in tubes", Units:"bar", Fixed);

      T_in              AS  temperature    (Description:"Inlet temperature of cooling water", Units:"C", Fixed, 30);
      T_out             AS  temperature    (Description:"Outlet temperature of cooling water", Units:"C");
      //T_rise          AS  temp_rise      (Description:"Inlet temperature of cooling water", Units:"C", Fixed);
      T_rise            AS  temp_rise      (Description:"Inlet temperature of cooling water", Units:"C",
                                                  Upper:1000); // Rev 001

      h_in              AS  enth_mol       (Description:"Inlet molar enthalpy of cooling water", Units:"GJ/kmol");
      h_out             AS  enth_mol       (Description:"Outlet molar enthalpy of cooling water", Units:"GJ/kmol");

      MW                AS  molweight      (Description:"MW of cooling water", Fixed, 18.0);

      z(WaterSet)       AS  molefraction   (Description:"Mole fraction of cooling water", Units:"kmol/kmol", Fixed,
                                                  1.0);

// PORTS

      I1                AS  INPUT Vapor    (Description:"Vapor inlet", ComponentList: GasSet);
      O1                AS  OUTPUT Vapor   (Description:"Condensate outlet", ComponentList: GasSet);

// EQUATIONS

// Material balance
EQ01:   I1.F = O1.F;
EQ01a: I1.FMol = O1.FMol;
EQ01b: O1.FV * DensMass = O1.F;
EQ01c: I1.w = O1.w;
EQ01d: I1.y = O1.y;
EQO1e: I1.T = O1.T;
EQ01f: I1.P = O1.P;

// Cooling water
EQ02:  FMol * MW = F;
EQ02a: T_out = T_in + T_rise;
EQ02b: P_out = P_in - P_drop;
EQ02c: FV * DensMass_H2O = F;                                                                    //
Rev 001

// Energy Balance
EQ03:  I1.FMol * (I1.h - O1.h) = FMol * (h_out - h_in);
```

```
// PROCEDURES
EQ020a:       CALL(DensMass) = pDens_Mass_Liq(O1.T, O1.P, O1.y); // Condensate mass density
EQ020b:       CALL(O1.h) = pEnth_Mol_Liq(O1.T, O1.P, O1.y);      // Condensate molar enthalpy
Eq020c:       CALL(h_in) = pEnth_Mol_Liq(T_in, P_in, z);         // Inlet molar enthalpy of cooling water
Eq020d:       CALL(h_out) = pEnth_Mol_Liq(T_out, P_out, z);      // Outlet molar enthalpy of cooling water

End
```

DropTank Model

```
Model DropTank
/*     WESTINGHOUSE SAVANNAH RIVER COMPANY
       INTEGRATED FLOWSHEET MODEL FOR THE HLW SYSTEM
       THE 2H & 3H EVAPORATOR FLOWSHEET MODELS

       Authors:      T. Hang and M.B. Gorensek, SRTC
       Version Date: 1 April 2002
       Description:  This module models the drop tank with content including supernate (liquid) and salt (solid).
                     Precipitation and dissolution are accounted for.

       Design Inputs/Outputs:
          Fixed Parameters/Variables
              Tank_Factor         Tank factor, m3/m
              Coil_number         Number of active cooling coils
              High_Level_Limit    High level operating limit, m
              Min_Coil_Depth      Minimum coil depth, m
              Min_Supernate_Depth Minimum supernate depth, m
              Bundle_Area         Area per coil bundle, m2
              FV_Cooling          Volumetric flow per coil bundle, m3/hr
              FV_Decant           Nominal decant flow to accumulator, m3/hr
              FV_Recycle          Nominal recycle flow, m3/hr
              T_Cooling_In        Cooling water inlet temperature, C
              P                   Operating pressure, bar
              Volume_Cooling      Cooling volume per coil bundle, m3
              DensMol_Cooling     Cooling molar density, kmol/m3
              HeatTransCoef_Solids       Solids heat transfer coefficient, kW/m2/K
              HeatTransCoef       Cooling heat transfer coefficient, kW/m2/K
              z_Cooling           Array of cooling water mole fraction, kmol/kmol
              Boildownratio_value Set value of boildown ratio (0.95)

          Initial Variables
              T                   Tank holdup temperature, C
              T_Cooling           Average cooling water temperature, C
              Volume              Tank holdup volume, m3
              z                   Array of holdup mole fraction, kmol/kmol
              RadMoles            Array of rad holdup moles, kmol

          Input Stream/Signal
              InPort              Multiple inlet streams (MultiPort)
              DestTank_FillStatus_Signal
                                  Destination tank fill status signal
```

```
          Output Stream/Signal
              OutPort.Connection("Recycle")           Recycle stream
              OutPort.Connection("Accumulate") Transfer stream to accumulate tank
              FillStatus_Signal   Tank fill status signal

       Revision History:
              Rev No Date         Author       Description
              ------ ----------   --------------      ------------------------------------------------------------------------

*/

// PARAMETERS & VARIABLES

       MainSet             AS  ComponentListName("Default");// Apparent Component List
       AqueousSet          AS  ComponentListName("Aqueous");       // True Component List
       SolidsSet           AS  ComponentListName("Solids");        // Solids Component List
       RadSet              AS  ComponentListName("Rads");          // Rads Component List
       WaterSet            AS  ComponentListName("Water");         // Vapor Component List
       PrecipCompSet       AS  StringSet(Description:"Set of precipitation component names");
       PrecipCompName(SolidsSet) AS  StringParameter  (Description:"Name of precipitation components");
       kW2GJperHr          AS  RealParameter   (Description:"kW ---> GJ/hr");
       Tank_Factor         AS  RealParameter   (Description:"m3/m");
       Coil_number         AS  IntegerParameter(Description:"Number of active cooling coils");

       Total_Level         AS  length          (Description:"Total tank level", Units:"m");
       High_Level_Limit    AS  length          (Description:"High level operating limit", Units:"m", Fixed, 9.1948);
       Heel_Level          AS  length          (Description:"Heel level", Units:"m", Fixed, 0);
       Supernate_Level     AS  length          (Description:"Supernate level", Units:"m");
       Solids_Level        AS  length          (Description:"Solids level", Units:"m");
       Min_Coil_Depth      AS  length          (Description:"Minimum coil depth", Units:"m", Fixed, 0.2286);
       Min_Supernate_Depth AS  length          (Description:"Minimum supernate depth", Units:"m", Fixed, 0.1524);
       High_Decant_Level   AS  length          (Description:"Full Tank level", Units:"m");
       Low_Decant_Level    AS  length          (Description:"Minimum decant level", Units:"m");

       Coil_Area           AS  area            (Description:"Coil surface area", Units:"m2");
       Bundle_Area         AS  area            (Description:"Area per coil bundle", Units:"m2", Fixed, 42.964);

       FMol_Cooling_Total  AS  flow_mol        (Description:"Total cooling molar flow", Units:"kmol/hr");
       FV_Cooling          AS  flow_vol        (Description:"Volumetric flow per cooling bundle", Units:"m3/hr", Fixed,
                                                     2.3846);
       FV_Decant           AS  flow_vol        (Description:"Nominal decant flow", Units:"m3/hr", Fixed, 27.254965);
       FV_Recycle          AS  flow_vol        (Description:"Nominal recycle flow", Units:"m3/hr", Fixed, 27.254965);
```

```
T                   AS  temperature     (Description:"Tank holdup temperature", Units:"C", Initial, 75);
T_Cooling           AS  temperature     (Description:"Average cooling water temperature", Units:"C", Initial, 25);
T_Cooling_In        AS  temperature     (Description:"Cooling water inlet temperature", Units:"C", Fixed, 25);
T_Cooling_Out       AS  temperature     (Description:"Cooling water outlet temperature", Units:"C");
T_Min               AS  temperature     (Description:"Supernate minimum temperature", Units:"C", Fixed, 65);
T_Max               AS  temperature     (Description:"Supernate maximum temperature", Units:"C", Fixed, 85);

P                   AS  pressure        (Description:"Operating pressure", Units:"bar", Fixed, 1.0133);

Mass                AS  holdup_mass     (Description:"Tank holdup mass", Units:"kg");
Moles               AS  holdup_mol      (Description:"Tank holdup moles", Units:"kmol");
Volume              AS  volume          (Description:"Tank holdup volume", Units:"m3", Initial, 1000);

Mass_Supernate      AS  holdup_mass     (Description:"Supernate holdup mass", Units:"kg");
Moles_Supernate     AS  holdup_mol      (Description:"Supernate holdup moles", Units:"kmol");
Volume_Supernate    AS  volume          (Description:"Supernate holdup volume", Units:"m3");

Mass_Solids         AS  holdup_mass     (Description:"Solids holdup mass", Units:"kg");
Moles_Solids        AS  holdup_mol      (Description:"Solids holdup moles", Units:"kmol");
Volume_Solids       AS  volume          (Description:"Solids holdup volume", Units:"m3");

Volume_Cooling      AS  volume          (Description:"Cooling volume per coil bundle", Units:"m3", Fixed, 0.489);

DensMass            AS  dens_mass       (Description:"Tank holdup density", Units:"kg/m3");
DensMass_Supernate  AS  dens_mass       (Description:"Supernate density", Units:"kg/m3");
DensMass_Solids     AS  dens_mass       (Description:"Solids density", Units:"kg/m3");

DensMol_Cooling     AS  dens_mol        (Description:"Cooling molar density", Units:"kmol/m3", Fixed, 55.56);

Q_Cooling           AS  enthflow        (Description:"Cooling heat", Units:"GJ/hr");
Q_Gamma             AS  enthflow        (Description:"Radioactive decay heat", Units:"GJ/hr");

cp_mol_Cooling      AS  cp_mol          (Description:"Cooling molar heat capacity", Units:"kJ/kmol/K");

MW                  AS  molweight       (Description:"Holdup MW");
MW_Supernate        AS  molweight       (Description:"Supernate MW");
MW_Solids           AS  molweight       (Description:"Solids MW");

h                   AS  enth_mol        (Description:"Tank holdup molar enthalpy", Units:"GJ/kmol");
h_Supernate         AS  enth_mol        (Description:"Supernate molar enthalpy", Units:"GJ/kmol");
h_Solids            AS  enth_mol        (Description:"Solids molar enthalpy", Units:"GJ/kmol");

HeatTransCoef_Solids AS  heat_trans_coeff (Description:"Solids heat transfer coefficient", Units:"kW/m2/K", Fixed,
                                           0.16981);
```

```
      HeatTransCoef       AS  heat_trans_coeff      (Description:"Cooling heat transfer coefficient", Units:"kW/m2/K",
                                                    Fixed, 0.03774);

      sfrac               AS  ratio_          (Description:"True solid to true liquid mole ratio");
      lfrac               AS  ratio_          (Description:"True liquid to apparent liquid mole ratio");
      Boildown_ratio      AS  ratio_          (Description:"Boildown ratio");
      Boildownratio_value AS  ratio_          (Description:"Set value of boildown ratio", Fixed, 0.95);

      Drain_Signal        AS  notype          (Description:"ON/OFF Drain signal", Fixed, 1);
      Feed_Flag           AS  notype          (Description:"Feed flag");
      RecycleStatus       AS  notype          (Description:"Recycle ready flag");

      Rate(MainSet)       AS  reaction        (Description:"Molar reaction rate", Units:"kmol/hr/m3", Fixed, 0);
      RxnH(MainSet)       AS  enthflow        (Description:"Component reaction heat flow", Units:"GJ/hr", Fixed, 0);

      z(MainSet)          AS  molefraction    (Description:"Tank holdup mole fraction", Units:"kmol/kmol", Initial);
      z_Supernate(MainSet)AS  molefraction    (Description:"Supernate mole fraction", Units:"kmol/kmol");
      z_Solids(MainSet)   AS  molefraction    (Description:"Solids mole fraction", Units:"kmol/kmol");

      z_apparent(AqueousSet)AS  molefraction  (Description:"Apparent tank holdup mole fraction", Units:"kmol/kmol");
      z_true(AqueousSet)  AS  molefraction    (Description:"True tank holdup mole fraction", Units:"kmol/kmol");

      z_Cooling(WaterSet) AS  molefraction    (Description:"Cooling mole fraction", Units:"kmol/kmol", Fixed, 1);

      MWs(AqueousSet)     AS  HIDDEN molweight(Description:"Component MW");

      RadMoles(RadSet)    AS  holdup_mol      (Description:"Holdup rad holdup moles", Units:"kmol", Initial, 0);
      RadMoles_Supernate(RadSet) AS  holdup_mol (Description:"Supernate rad holdup moles", Units:"kmol");
      RadMoles_Solids(RadSet)    AS  holdup_mol (Description:"Solids rad holdup moles", Units:"kmol");

      radconc(RadSet)     AS  conc_mole       (Description:"Tank holdup rad concentration", Units:"mol/L");
      radconc_Supernate(RadSet) AS conc_mole  (Description:"Supernate rad concentration", Units:"mol/L");
      radconc_Solids(RadSet) AS  conc_mole    (Description:"Solids rad concentration", Units:"mol/L");

// SUBMODEL
      Precip              AS  PrecipComp      (Description:"Get a name set of components to precipitate");

// PORTS

      InPort              AS  INPUT MULTIPORT OF Main  (Description:"Multiple liquid inlet streams");
      OutPort             AS  OUTPUT MULTIPORT OF Main (Description:"Multiple liquid outlet streams");

      DestTank_FillStatus_Signal AS INPUT control_signal (Description:"Destination tank fill status signal", Fixed);
      FillStatus_Signal   AS  OUTPUT control_signal  (Description:"Tank fill status signal", Fixed);
```

```
      EvapFeedFV_Signal   AS  INPUT control_signal   (Description:"Evaporator feed vol flow signal", Fixed);

// ASSIGNMENTS

PrecipCompSet :      Precip.PrecipCompSet;
PrecipCompName       :      Precip.PrecipCompName;
kW2GJperHr    :      3.6E-3;
Tank_Factor   :      523.;
Coil_number   :      5;
   // Free DestTank_FillStatus_Signal if it is connected
   If DestTank_FillStatus_Signal.IsConnected Then
      DestTank_FillStatus_Signal.Spec : Free;
   Endif
   // Free EvapFeedFV_Signal if it is connected
   If EvapFeedFV_Signal.IsConnected Then
      EvapFeedFV_Signal.Spec : Free;
   Endif

// EQUATIONS

// Mole balance
EQ01:  $Moles = SIGMA(InPort.Connection.FMol) - SIGMA(OutPort.Connection.FMol) + Volume * SIGMA(Rate);
EQ01a: Mass = Moles * MW;
EQ01b: Volume * DensMass = Mass;

// Component mole balance
EQ02:  FOR i IN MainSet DO
          z(i) * $Moles + Moles * $z(i) =
             SIGMA(InPort.Connection.FMol*InPort.Connection.z(i)) -
                SIGMA(OutPort.Connection.FMol*OutPort.Connection.z(i)) + Volume * Rate(i);
       ENDFOR
       // Rad components
EQ02a: FOR i IN RadSet DO
          $RadMoles(i) = SIGMA(InPort.Connection.FRad(i)) - SIGMA(OutPort.Connection.FRad(i));
       ENDFOR
EQ02b: radconc * Volume = RadMoles;

// Energy Balance
EQ03:  Moles * $h + h * $Moles =
          SIGMA(InPort.Connection.FMol*InPort.Connection.h) -
             SIGMA(OutPort.Connection.FMol*OutPort.Connection.h) + SIGMA(RxnH) -
                Q_Cooling + Q_Gamma;
EQ03a: Q_Cooling = kW2GJperHr * HeatTransCoef * Coil_Area * (T - T_Cooling);
EQ03b: Q_Gamma = 0;
```

```
// Calculation to split holdup into Supernate (Liquid phase) and Salt (Solid phase)

       // Set up z_apparent to match z, but add missing species from "Aqueous" component list
       // with zero mole fraction
EQ04:  z_apparent(MainSet)  =  z;
EQ04a: z_apparent(AqueousSet - MainSet)  =  0;

       // Supernate phase
EQ05:  Moles_Supernate = Moles - Moles_Solids;
EQ05a: Mass_Supernate = Moles_Supernate * MW_Supernate;
EQ05b: Volume_Supernate * DensMass_Supernate = Mass_Supernate;
EQ05c: z_Supernate * Moles_Supernate = z * Moles - z_Solids * Moles_Solids;
EQ05d: RadMoles_Supernate * Moles  = RadMoles * Moles_Supernate;
EQ05e: radconc_Supernate * Volume_Supernate = RadMoles_Supernate;

       // Solids phase
EQ06:  Moles_Solids = sfrac * lfrac * Moles;
EQ06a: Mass_Solids = Moles_Solids * MW_Solids;
EQ06b: Volume_Solids * DensMass_Solids = Mass_Solids;
EQ06c: FOR i IN SolidsSet DO
          IF sfrac > 1E-6 THEN
             z_Solids(PrecipCompName(i)) * SIGMA(z_true(SolidsSet)) = z_true(i);
          ELSE
             z_Solids(PrecipCompName(i)) = 0.;
          ENDIF
       ENDFOR
EQ06d: z_Solids(MainSet-PrecipCompSet) = 0.;
EQ06e: RadMoles_Solids = RadMoles - RadMoles_Supernate;
EQ06f: IF sfrac > 1E-6 THEN
          radconc_Solids * Volume_Solids = radconc * Volume - radconc_Supernate * Volume_Supernate;
       ELSE
          radconc_Solids = 0.;
       ENDIF

// Cooling system
EQ07:  FMol_Cooling_Total = Coil_Number * FV_Cooling * DensMol_Cooling;
EQ07a: DensMol_Cooling * (Volume_Cooling * Coil_Number) * (cp_mol_Cooling * 1E-6) * $T_Cooling =
          FMol_Cooling_Total * (cp_mol_Cooling * 1E-6) * (T_Cooling_In - T_Cooling_Out) + Q_Cooling;
EQ07b: T_Cooling = 0.5 * (T_Cooling_In + T_Cooling_Out);

// Level and heat transfer area calculations
EQ08:  Total_Level = Supernate_Level + Solids_Level;
EQ08a: Supernate_Level * Tank_Factor = Volume_Supernate;
```

```
EQ08b: Solids_Level * Tank_Factor = Volume_Solids;
       // Coil heat transfer area
EQ08c: IF (Total_Level <= Min_Coil_Depth) THEN
          Coil_Area = 0.;
       ELSE
          Coil_Area * (High_Level_Limit - Min_Coil_Depth) =
             Bundle_Area * Coil_Number * (Total_Level - Min_Coil_Depth);
       ENDIF;
EQ08d: High_Decant_Level = High_Level_Limit;
EQ08e: IF Heel_Level > (Solids_Level + Min_Supernate_Depth) THEN
          Low_Decant_Level = Heel_Level;
       ELSE
          Low_Decant_Level = Solids_Level + Min_Supernate_Depth;
       ENDIF

// Decant logic
// Drain_Signal is set by Task DropTankDrain

// Outlet stream
       // Supernate recycle stream
EQ09:  OutPort.Connection("Recycle").FMol * MW_Supernate = OutPort.Connection("Recycle").F;
EQ09a: OutPort.Connection("Recycle").F = OutPort.Connection("Recycle").FV * DensMass_Supernate;
EQ09b: OutPort.Connection("Recycle").FV = RecycleStatus * DestTank_FillStatus_Signal * Drain_Signal * FV_Decant;
EQ09c: OutPort.Connection("Recycle").z = z_Supernate;
EQ09d: OutPort.Connection("Recycle").w * SIGMA(z_Supernate*MWs) = z_Supernate * MWs;
EQ09e: OutPort.Connection("Recycle").T = T;
EQ09f: OutPort.Connection("Recycle").P = P;
EQ09g: OutPort.Connection("Recycle").h = h_Supernate;
EQ09h: OutPort.Connection("Recycle").FRad = OutPort.Connection("Recycle").FV * radconc_Supernate;
EQ09i: OutPort.Connection("Recycle").radconc = radconc_Supernate;

       // Supernate transfer stream to accumulate tank
EQ10:  OutPort.Connection("Accumulate").FMol * MW_Supernate = OutPort.Connection("Accumulate").F;
EQ10a: OutPort.Connection("Accumulate").F = OutPort.Connection("Accumulate").FV * DensMass_Supernate;
EQ10b: OutPort.Connection("Accumulate").FV = (1-RecycleStatus) * Drain_Signal * FV_Decant;
EQ10c: OutPort.Connection("Accumulate").z = z_Supernate;
EQ10d: OutPort.Connection("Accumulate").w * SIGMA(z_Supernate*MWs) = z_Supernate * MWs;
EQ10e: OutPort.Connection("Accumulate").T = T;
EQ10f: OutPort.Connection("Accumulate").P = P;
EQ10g: OutPort.Connection("Accumulate").h = h_Supernate;
EQ10h: OutPort.Connection("Accumulate").FRad = OutPort.Connection("Accumulate").FV * radconc_Supernate;
EQ10i: OutPort.Connection("Accumulate").radconc = radconc_Supernate;

// Transfer logic (Destination tank: Feed tank or Accumulate tank) based on Boildown_ratio
```

```
      // Boildown_ratio = (Drop tank vol. Inflow) / (Evap. vol. feed flow)
EQ11:  IF EvapFeedFV_Signal > 0 THEN
          Boildown_ratio * EvapFeedFV_Signal = SIGMA(InPort.Connection.FV);
       ELSE
          Boildown_ratio = 0;
       ENDIF
EQ11a: IF Boildown_ratio < Boildownratio_value THEN
          RecycleStatus = 1;
       ELSE
          RecycleStatus = 0;
       ENDIF

// PROCEDURES
EQ20:  CALL(MWs) = pMolWeights() AqueousSet;                              // Component MWs
EQ20a: CALL(MW) = pMolWeight(z) MainSet;                                  // Tank holdup apparent molecular weight
EQ20b: CALL(MW_Supernate) = pMolWeight(z_Supernate) MainSet;              // Supernate molecular weight
EQ20c: CALL(MW_Solids) = pMolWeight(z_Solids) MainSet;                    // Solids molecular weight
EQ20d: CALL(DensMass) = pDens_Mass_Liq(T, P, z) MainSet;                  // Tank holdup density
EQ20e: CALL(DensMass_Supernate) = pDens_Mass_Liq(T, P, z_Supernate) MainSet;    // Supernate density
EQ20f: CALL(DensMass_Solids) = pDens_Mass_Sol(T, P, z_Solids) MainSet;    // Solids density
EQ20g: CALL(h) = pEnth_Mol_Liq(T, P, z) MainSet;                          // Tank holdup molar enthalpy
EQ20h: CALL(h_Supernate) = pEnth_Mol_Liq(T, P, z_Supernate) MainSet;      // Supernate molar enthalpy
EQ20i: CALL(h_Solids) = pEnth_Mol_Sol(T, P, z_Solids) MainSet;            // Solids molar enthalpy
EQ20j: CALL(cp_mol_Cooling) = pCp_Mol_Liq(T_Cooling, P, z_Cooling);       // Cooling molar heat capacity
EQ20k: CALL(z_true, sfrac, lfrac) = pTrueComp (T, P, z_apparent) AqueousSet; // Convert tank holdup from apparent to true
                                                                          // components

End
```

Evap Model

```
Model Evap
/*     WESTINGHOUSE SAVANNAH RIVER COMPANY
       INTEGRATED FLOWSHEET MODEL FOR THE HLW SYSTEM
       THE 2H & 3H EVAPORATOR FLOWSHEET MODELS

       Author:      T. Hang and M.B. Gorensek, SRTC
       Version Date: 25 April 2002
       Description:  This module models the operation of an evaporator

       Design Inputs/Outputs:
          Fixed Parameters/Variables
              FSteamBundle_Manual Manually set tube bundle steam flow, kg/hr (4536)
              FSteamBundle_Min    Minimum tube bundle steam flow, kg/hr (2722)
              FSteamBundle_Max    Maximum tube bundle steam flow, kg/hr (6804
              FSteamLift_Min      Minimum steam lift mass flow, kg/hr (29.483504)
              FSteamLift_Max      Maximum steam lift mass flow, kg/hr (227)
              P                   Specified operating pressure, bar
              P_drop              Pressure drop in tube bundles, bar
              Rate                Array of component reaction rate in waste solution, kmol/hr
              RxnH                Array of component reaction heat flow in waste solution, GJ/hr
              LevelControl_ON     Level control ON/OFF signal (1: control ON; 0: control OFF)
              Liquid_Level_LOLO   LO-LO liquid level, m (2.2606)
              Liquid_Level_LO     LO liquid level, m (2.3368)
              Liquid_Level_HI     HI liquid level, m (2.4892)
              Liquid_Level_HIHI   HI-HI liquid level, m (2.5654)
              LiquidLevel_SetPoint      Concentrate level at level setpoint, m (2.413)
              SpGControl_ON       SpG control ON/OFF signal (1: control ON, 0: control OFF)
              SpG_SetPoint        SpG setpoint, C (1.3)

          Initial Variables
              T                   Solution temperature, C
              Volume              Holdup volume, m3
              z                   Array of holdup mole fraction
              RadMoles            Array of Rads component holdup moles, kmol

          Input Streams/Variables
              I1                  Feed stream
              I2                  Steam lance inlet stream
              I3                  Steam lift inlet stream
              I4                  Tube bundle steam inlet stream
```

```
          Output Streams/Variables
              O1                  Evaporator concentrate stream
              O2                  Evaporator overheads stream
              O3                  Steam lift outlet stream
              FillStatus_Signal   Evaporator fill status signal
              LiquidLevel_Signal  Liquid level signal

       Revision History:
              Rev No Date         Author       Description
              ------ ----------   --------------      ------------------------------------------------------------------------

*/

// PARAMETERS & VARIABLES

       MainSet             AS  ComponentListName("Default");       // Main (apparent) Component List
       GasSet              AS  ComponentListName("Vapor");         // Vapor Component List
       AqueousSet          AS  ComponentListName("Aqueous");       // True Component List
       RadSet              AS  ComponentListName("Rads");          // Rads Component List
       LevelControl_ON     AS  RealParameter   (Description:"Level control ON/OFF signal");
       SpGControl_ON       AS  RealParameter   (Description:"SpG control ON/OFF signal");

       LiquidLevel_SetPoint AS  length         (Description:"Concentrate level at level setpoint", Units:"m", Fixed,
                                                      2.413);
       Liquid_Level        AS  length          (Description:"Concentrate level", Units:"m", Initial);
       Liquid_Level_LO     AS  length          (Description:"LO concentrate level", Units:"m", Fixed, 2.3368);
       Liquid_Level_LOLO   AS  length          (Description:"LO-LO concentrate level", Units:"m", Fixed, 2.2606);
       Liquid_Level_HI     AS  length          (Description:"HI concentrate level", Units:"m", Fixed, 2.4892);
       Liquid_Level_HIHI   AS  length          (Description:"HI-HI concentrate level", Units:"m", Fixed, 2.5654);

       FSteamBundle        AS  flow_mass_      (Description:"Tube bundle steam mass flow", Units:"kg/hr");
       FSteamBundle_Manual AS  flow_mass       (Description:"Manually set tube bundle steam mass flow", Units:"kg/hr",
                                                      Fixed, 7500);
       FSteamBundle_Min    AS  flow_mass       (Description:"Minimum tube bundle steam mass flow", Units:"kg/hr", Fixed,
                                                      0);
       FSteamBundle_Max    AS  flow_mass       (Description:"Maximum tube bundle steam mass flow", Units:"kg/hr", Fixed,
                                                      9072);
       FSteamLift          AS  flow_mass_      (Description:"Steam lift mass flow", Units:"kg/hr");
       FSteamLift_Min      AS  flow_mass       (Description:"Minimum steam lift mass flow", Units:"kg/hr", Fixed,
                                                      29.483504);
       FSteamLift_Max      AS  flow_mass       (Description:"Maximum steam lift mass flow", Units:"kg/hr", Fixed, 227);

       FVap                AS  flow_mol        (Description:"Evaporation molar flow", Units:"kmol/hr");
       FMolLift_SetPoint   AS  flow_mol_       (Description:"Concentrate molar lift rate at level setpoint",
```

```
                                                Units:"kmol/hr");
FMolLift            AS  flow_mol        (Description:"Concentrate molar lift rate", Units:"kmol/hr");

Fcalc               AS  flow_vol        (Description:"Calculated concentrate flow rate", Units:"m3/hr", Lower:-1E3);

T                   AS  temperature     (Description:"Solution temperature", Units:"C", Initial);
TB                  AS  temperature     (Description:"Solution boiling point", Units:"C");
TS_out              AS  temperature     (Description:"Temperature at tube bundle outlet", Units:"C");

P                   AS  pressure        (Description:"Operating pressure", Units:"bar", Fixed, 1.0133);
P_drop              AS  press_drop      (Description:"Pressure drop in tubes", Units:"bar", Fixed, 1.379 );
PS_out              AS  pressure        (Description:"Pressure at tube bundle outlet", Units:"bar");
PVap                AS  pressure        (Description:"Total vapor pressure", Units:"bar");

Mass                AS  holdup_mass     (Description:"Liquid holdup mass", Units:"kg", Upper:1E10);
Moles               AS  holdup_mol      (Description:"Liquid holdup moles", Units:"kmol", Upper:1E10);
Volume              AS  volume          (Description:"Holdup volume", Units:"m3");
DensMass            AS  dens_mass       (Description:"Holdup density", Units:"kg/m3");

Vap_DensMass        AS  dens_mass       (Description:"Vapor density", Units:"kg/m3");

Moles_Solids        AS  holdup_mol      (Description:"Solids holdup moles", Units:"kmol");
c_Solids            AS  conc_mole       (Description:"Solids concentration", Units:"kmol/m3");

Q                   AS  enthflow        (Description:"Steam heat flow", Units:"GJ/hr");

MW                  AS  molweight       (Description:"Holdup MW");
MW_Vap              AS  molweight       (Description:"Vapor MW");
MW_true             AS  molweight       (Description:"Holdup true-comp MW");

hv                  AS  enth_mol_vap    (Description:"Vapor molar enthalpy", Units:"GJ/kmol");
h                   AS  enth_mol        (Description:"Solution molar enthalpy", Units:"GJ/kmol");
hw                  AS  enth_mol        (Description:"Steam condensate molar enthalpy", Units:"GJ/kmol");

sfrac               AS  ratio_          (Description:"True solid to true liquid mole ratio");
lfrac               AS  ratio_          (Description:"True liquid to apparent liquid mole ratio");

c_true(AqueousSet)  AS  conc_mole       (Description:"Holdup true component concentration", Units:"kmol/m3");

SpG                 AS  pos_small       (Description:"Holdup specific gravity", Upper:10.0);
SpG_SetPoint        AS  pos_small       (Description:"Specific gravity setpoint", Upper:10.0, Fixed, 1.3);
SpG_Min             AS  pos_small       (Description:"Minimum specific gravity", Upper:10, Fixed, 0);
SpG_Max             AS  pos_small       (Description:"Maximium specific gravity", Upper:10, Fixed, 2);
```

```
      Gain                AS  pos_small       (Description:"SpG control gain", Fixed, 10);
      PV_Scale            AS  percentu        (Description:"Scaled input for SpG control", Units:"%");
      SP_Scale            AS  percentu        (Description:"Scaled set point for SpG control", Units:"%");
      Bias_Scale          AS  percentu        (Description:"Scaled bias for SpG control", Units:"%");
      OP_Scale            AS  percentu        (Description:"Scaled output for SpG control", Units:"%");
      PropTerm            AS  percentu        (Description:"Proportional term for SpG control");

      Rate(MainSet)       AS  flow_mol        (Description:"Component reaction rate", Units:"kmol/hr", Fixed, 0.);
      RxnH(MainSet)       AS  enthflow        (Description:"Component reaction heat flow", Units:"GJ/hr", Fixed, 0.);
      VP(MainSet)         AS  pressure        (Description:"Component vapor pressure", Lower:0, Units:"bar");
      gamma(MainSet)      AS  act_coeff_liq   (Description:"Component liquid activity coefficient");

      y(GasSet)           AS  molefraction    (Description:"Vapor mole fraction", Units:"kmol/kmol");

      w(MainSet)          AS  massfraction    (Description:"Holdup mass fraction", Units:"kg/kg");
      z(MainSet)          As  molefraction    (Description:"Holdup mole fraction", Units:"kmol/kmol", Initial);

      z_Aqueous(AqueousSet)     AS  molefraction    (Description:"Holdup app-comp mole fraction", Units:"kmol/kmol");
      z_true(AqueousSet)  AS  molefraction    (Description:"Holdup true-comp mole fraction", Units:"kmol/kmol");
      MWs(AqueousSet)     AS  HIDDEN molweight(Description:"Component MW");

      RadMoles(RadSet)    AS  holdup_mol      (Description:"Rad holdup moles", Units:"kmol", Initial);
      radconc(RadSet)     AS  conc_mole       (Description:"Rad concentration", Units:"mol/L");

// PORTS
      I1                  AS  INPUT Main            (Description:"Feed", ComponentList: MainSet);
      I2                  AS  INPUT Vapor           (Description:"Steam lance inlet", ComponentList: GasSet);
      I3                  AS  INPUT Vapor           (Description:"Steam lift inlet", ComponentList: GasSet);
      I4                  AS  INPUT Vapor           (Description:"Steam to tube bundle inlet", ComponentList: GasSet);

      O1                  AS  OUTPUT MainSimple     (Description:"Steam-lifted concentrate", ComponentList: MainSet);
      O2                  AS  OUTPUT Vapor    (Description:"Vapor overheads", ComponentList: GasSet);

      FillStatus_Signal   AS  OUTPUT control_signal  (Description:"Evaporator fill status signal");
      LiquidLevel_Signal  AS  OUTPUT control_signal  (Description:"Concentrate level as control signal");
      EvapFeedFV_Signal   AS  OUTPUT control_signal  (Description:"Feed vol flow as control signal");

// EQUATIONS

// Mole balances
EQ01:   $Moles = I1.FMol + I2.FMol - FMolLift - O2.FMol + SIGMA(Rate);
EQ01a:  Moles * MW = Mass;
EQ01b:  Volume * DensMass = Mass;
```

```
// Component mole balances
        // Non-volatile components
EQ02:   z(MainSet-GasSet)*$Moles + Moles*$z = I1.FMol*I1.z + Rate - FMolLift*z;

        // Volatile components
EQ02a:  z(GasSet)*$Moles + Moles*$z = I1.FMol*I1.z + I2.FMol*I2.y + Rate - FMolLift*z - O2.FMol*O2.y;
EQ02b:  w * SIGMA(z*MWs) = z * MWs;

        // Rad components
EQ02c:  $RadMoles = I1.FRad - O1.FRad;
EQ02d:  radconc * Volume = RadMoles;

// Energy balances
EQ03:   Moles*$h + h*$Moles = I1.FMol*I1.h + I2.FMol*I2.h + Q + SIGMA(RxnH) - FMolLift*h - O2.FMol*O2.h;
        // Heat flow from steam in tube bundles
        // Assuming total condensation of steam in tubes
EQ03a: PS_out = I4.P - P_drop;
EQ03b:  Q = I4.FMol * (I4.h - hw);

// Boilup rate
// Use of control function (PVap/P)^3 to ensure PVap --> P and T --> TB
EQ04:  FVap*(hv-h) = (Q + I1.FMol*(I1.h-h) + I2.FMol*(I2.h-h) + SIGMA(RxnH)) * (PVap/P)^3;

// Vapor-Liquid equilibrium
EQ05:   PVap = SIGMA(VP(GasSet)*gamma(GasSet)*z(GasSet));
EQ05a:  y(GasSet) * PVap = VP * gamma * z;

// Output streams

       // Steam-lifted concentrate stream
       // If there is insufficient steam flow to maintain lift, the steam exits by itself
       // Use steam lift correlation to determine concentrate, rads flow rates
EQ06:  IF (Fcalc > 0.0) THEN
          FMolLift * MW = Fcalc * DensMass;
          O1.FRad = Fcalc * radconc;
       ELSE
          // Set FMolLift to 1E-6 to avoid convergence problem in Module Separator
          FMolLift = 1E-6;
          O1.FRad = 0.0;
       ENDIF
       // Combine lifted concentrate and lift steam
EQ06a:  O1.FMol = FMolLift + I3.FMol;
       // Calculate component mole fractions in combined stream
EQ06b:  O1.z(GasSet) * O1.FMol = (FMolLift * z + I3.FMol * I3.y);
```

```
EQ06c:  O1.z(MainSet-GasSet) * O1.FMol = FMolLift * z;
EQ06d:  O1.P = P;
       // Calculate molar enthalpy of combined stream
EQ06e:  O1.h * O1.FMol = (FMolLift * h + I3.FMol * I3.h);
       // Steam lift correlation for 3H Evaporator
       // Ref.: "Replacement High Level Waste Evaporator - System Design Description/Evaporator System"
       //       G-SYD-H-00026, Rev. 2
       //       Figure 2.5-1: "Steam Lift Performance", Page 52
EQ06f: Fcalc = 60.0*3.7854118E-3*((0.327397623716747*SpG - 0.312802419170903)*(I3.F/0.45359237) - 18.4306795860704);

        // Vapor stream
EQ07:   O2.FMol * MW_Vap = O2.F;
EQ07a:  O2.FMol = FVap;
EQ07b:  O2.FV * Vap_DensMass = O2.F;
EQ07c:  O2.w * SIGMA(y*MWs) = y*MWs;
EQ07d:  O2.y = y;
EQ07e:  O2.T = T;
EQ07f:  O2.P = PVap;
EQ07g:  O2.h = hv;

// True components calculation
EQ08:  z_Aqueous(MainSet) = z;
EQ08a: z_Aqueous(AqueousSet-MainSet) = 0.;
EQ08b:        c_true * MW_true * Volume = Mass * z_true;
       // Total solids
EQ08c: Moles_Solids = sfrac * lfrac * Moles;
EQ08d: c_Solids * Volume = Moles_Solids;

// Fill logic
//EQ09:       IF (Liquid_Level > Liquid_Level_HIHI
//        OR (Liquid_Level >= Liquid_Level_LOLO
//           AND FillStatus_Signal < 0.5)) THEN
//        FillStatus_Signal = 0.;
//     ELSE
//        FillStatus_Signal = 1.;
//     ENDIF
EQ09:  FillStatus_Signal = 1.;

// Definition of specific gravity
EQ10:   SpG = DensMass/1000.;

// Volume and liquid level relationship
// Correlation:     V = 86.1371 L + 1676.2606
// with V (gals), L (inch) and 1.03 <= L <= 108.43
```

```
EQ11:  Volume * (1000/3.785412) = 86.1371 * Liquid_Level * (100/2.54) + 1676.2606;
EQ11a: LiquidLevel_Signal = Liquid_Level;

// Add liquid level control (without actual use of a level controller)
EQ12:  FMolLift_SetPoint = LevelControl_ON * (I1.FMol + I2.FMol + SIGMA(Rate) - O2.FMol) *
             (Liquid_Level/LiquidLevel_SetPoint)^3;
EQ12a: IF LevelControl_ON > 0 THEN
          (FMolLift_SetPoint * MW / DensMass) =
             60.0*3.7854118E-3*((0.327397623716747*SpG - 0.312802419170903)*(FSteamLift/0.45359237) - 18.4306795860704);
       ELSE
          FSteamLift = FSteamLift_Min;
       ENDIF
EQ12b: IF FSteamLift <= FSteamLift_Min THEN
          I3.F = FSteamLift_Min;
       ELSEIF FSteamLift >= FSteamLift_Max THEN
          I3.F = FSteamLift_Max;
       ELSE
          I3.F = FSteamLift;
       ENDIF

// Add SpG control by using a modified Proportional controller
EQ13:  IF SpG > SpG_Max THEN
          PV_Scale = 100;
       ELSEIF (SpG < SpG_Min) THEN
          PV_Scale = 0;
       ELSE
          PV_Scale * (SpG_Max - SpG_Min) = 100 * (SpG - SpG_Min);
       ENDIF
EQ13a: SP_Scale * (SpG_Max - SpG_Min) = 100 * (SpG_SetPoint - SpG_Min);
EQ13b: Bias_Scale * (FSteamBundle_Max - FSteamBundle_Min) = 100 * (FSteamBundle_Manual - FSteamBundle_Min);
EQ13c: PropTerm = SP_Scale - PV_Scale;
EQ13d: IF SP_Scale > PV_Scale THEN
          IF PV_Scale > 0 THEN
             OP_Scale = Bias_Scale + Gain * PropTerm * (SP_Scale/PV_Scale)^3;
          ELSE
             OP_Scale = 100;
          ENDIF
       ELSE
          OP_Scale = Bias_Scale + Gain * PropTerm * (PV_Scale/SP_Scale)^3;
       ENDIF
EQ13e: FSteamBundle =
          (1 - SpGControl_ON) * FSteamBundle_Manual +
             SpGControl_ON * (OP_Scale/100) * (FSteamBundle_Max - FSteamBundle_Min);
EQ13f: IF (FSteamBundle > FSteamBundle_Max) THEN
```

```
         I4.F = FSteamBundle_Max;
      ELSEIF (FSteamBundle < FSteamBundle_Min) THEN
         I4.F = FSteamBundle_Min;
      ELSE
         I4.F = FSteamBundle;
      ENDIF

// Set feed volumetric flow as output control signal
EQ14:  EvapFeedFV_Signal = I1.FV;

// PROCEDURES
EQ20:  CALL(MWs) = pMolWeights() AqueousSet;              // Component MWs
EQ20a: CALL(MW) = pMolWeight(z);                          // Apparent-component MW
EQ20b: CALL(DensMass) = pDens_Mass_Liq(T, P, z);          // Liquid mass density
EQ20c: CALL(h) = pEnth_Mol_Liq(T, P, z);                  // Liquid molar enthalpy
EQ20d: CALL(Vap_DensMass) = pDens_Mass_Vap(O2.T, O2.P, y); // Vapor mass density
EQ20e: CALL(hv) = pEnth_Mol_Vap(T, PVap, y);              // Vapor molar enthalpy
EQ20f: CALL(MW_Vap) = pMolWeight(y);                      // Vapor molecular weight
EQ20g: CALL(TB) = pBubt(P, z);                            // Bubble point temperature
EQ20h: CALL(VP) = pVap_Pressures(T);                      // Component vapor pressures
EQ20i: CALL(gamma) = pAct_Coeff_Liq(T, P, z);             // Component liquid activity coefficients
EQ20j: CALL(TS_out) = pDewt(PS_out, I4.y);                // Temperature at tube bundle outlet
EQ20k: CALL(hw) = pEnth_Mol_Liq(TS_out, PS_out, I4.y);    // Steam molar enthalpy at tuble bundle outlet
EQ20l: CALL(z_true, sfrac, lfrac) = pTrueComp (T, P, z_Aqueous) AqueousSet;  // Convert to true components
EQ20m: CALL(MW_true) = pMolWeight(z_true) AqueousSet;     // True-component MW

End
```

FeedTank Model

```
Model FeedTank
/*     WESTINGHOUSE SAVANNAH RIVER COMPANY
       INTEGRATED FLOWSHEET MODEL FOR THE HLW SYSTEM
       THE 2H & 3H EVAPORATOR FLOWSHEET MODELS

       Author:      T. Hang and M.B. Gorensek, SRTC
       Version Date:1 April 2002
       Description:  This module models a feed tank with content including supernate (liquid) and solids
                     (salt/sludge).  Precipitation and dissolution are accounted for.

       Design Inputs/Outputs:
          Fixed Parameters/Variables
              Tank_Factor         Tank factor, m3/m
              Coil_number         Number of active cooling coils
              High_Level_Limit    High level operating limit, m
              Low_Level_Limit     Low level operating limit, m
              Min_Coil_Depth      Minimum coil depth, m
              Min_Supernate_Depth Minimum supernate depth, m
              Bundle_Area         Area per coil bundle, m2
              FV_Cooling          Volumetric flow per coil bundle, m3/hr
              FV_Decant           Nominal supernate decant flow, m3/hr
              T_Cooling_In        Cooling water inlet temperature, C
              P                   Operating pressure, bar
              Volume_Cooling      Cooling volume per coil bundle, m3
              DensMol_Cooling     Cooling molar density, kmol/m3
              HeatTransCoef_Solids       Solids heat transfer coefficient, kW/m2/K
              HeatTransCoef       Cooling heat transfer coefficient, kW/m2/K
              z_Cooling           Array of cooling water mole fraction, kmol/kmol

          Initial Variables
              T                   Tank holdup temperature, C
              T_Cooling           Average cooling water temperature, C
              Volume              Tank holdup volume, m3
              z                   Array of holdup mole fraction, kmol/kmol
              RadMoles            Array of rad holdup moles, kmol

          Input Stream/Signal
              InPort              Multiple liquid inlet streams (MultiPort)
              DestTank_FillStatus_Signal
                                  Destination tank fill status signal
```

```
         Output Stream/Signal
             OutPort                    Supernate transfer stream
             FeedFillStatus_Signal      Tank fill status signal for feed streams
             RecycleFillStatus_Signal   Tank fill status signal for recycle streams

       Revision History:
             Rev No Date          Author       Description
             ------ ----------    --------------      ------------------------------------------------------------------------

*/

// PARAMETERS & VARIABLES

       MainSet             AS  ComponentListName("Default");// Apparent Component List
       AqueousSet          AS  ComponentListName("Aqueous");// True Component List
       SolidsSet           AS  ComponentListName("Solids");  // Solids Component List
       RadSet              AS  ComponentListName("Rads");    // Rads Component List
       WaterSet            AS  ComponentListName("Water");   // Vapor Component List
       PrecipCompSet       AS  StringSet       (Description:"Set of precipitation component names");
       PrecipCompName(SolidsSet) AS  StringParameter  (Description:"Name of precipitation components");
       kW2GJperHr          AS  RealParameter   (Description:"kW ---> GJ/hr");
       Tank_Factor         AS  RealParameter   (Description:"m3/m");
       Coil_number         AS  IntegerParameter(Description:"Number of active cooling coils");

       Total_Level         AS  length          (Description:"Total tank level", Units:"m");
       High_Level_Limit    AS  length          (Description:"High level operating limit", Units:"m", Fixed, 9.1948);
       Low_Level_Limit     AS  length          (Description:"Low level operating limit", Units:"m", Fixed, 0);
       WorkingSpace_Level  AS  length          (Description:"Working space level", Units:"m");
       Heel_Level          AS  length          (Description:"Heel level", Units:"m", Fixed, 0);
       Supernate_Level     AS  length          (Description:"Supernate level", Units:"m");
       Solids_Level        AS  length          (Description:"Solids level", Units:"m");
       Min_Coil_Depth      AS  length          (Description:"Minimum coil depth", Units:"m", Fixed, 0.2286);
       Min_Supernate_Depth AS  length          (Description:"Minimum supernate depth", Units:"m", Fixed, 0.1524);
       High_Decant_Level   AS  length          (Description:"Full Tank level", Units:"m");
       Low_Decant_Level    AS  length          (Description:"Minimum decant level", Units:"m");

       Coil_Area           AS  area            (Description:"Coil surface area", Units:"m2");
       Bundle_Area         AS  area            (Description:"Area per coil bundle", Units:"m2", Fixed, 42.964);

       FV_In               AS  flow_vol        (Description:"Total volumetric flow of feeds", Units:"m3/hr");
       FMol_Cooling_Total  AS  flow_mol        (Description:"Total cooling molar flow", Units:"kmol/hr");
       FV_Cooling          AS  flow_vol        (Description:"Volumetric flow per cooling bundle", Units:"m3/hr", Fixed,
2.3846);
       FV_Decant           AS  flow_vol        (Description:"Nominal supernate decant flow", Units:"m3/hr", Fixed, 27.252);
```

```
T                   AS  temperature     (Description:"Tank holdup temperature", Units:"C", Initial, 40);
T_Cooling           AS  temperature     (Description:"Average cooling water temperature", Units:"C", Initial, 25);
T_Cooling_In        AS  temperature     (Description:"Cooling water inlet temperature", Units:"C", Fixed, 25);
T_Cooling_Out       AS  temperature     (Description:"Cooling water outlet temperature", Units:"C");
T_Min               AS  temperature     (Description:"Supernate minimum temperature", Units:"C", Fixed, 65);
T_Max               AS  temperature     (Description:"Supernate maximum temperature", Units:"C", Fixed, 85);

P                   AS  pressure        (Description:"Operating pressure", Units:"bar", Fixed, 1.0133);
WorkingSpace_Volume AS  volume          (Description:"Working space volume", Units:"m3", Fixed, 378.54118);

Mass                AS  holdup_mass     (Description:"Tank holdup mass", Units:"kg");
Moles               AS  holdup_mol      (Description:"Tank holdup moles", Units:"kmol");
Volume              AS  volume          (Description:"Tank holdup volume", Units:"m3", Initial, 1000);

Mass_Supernate      AS  holdup_mass     (Description:"Supernate holdup mass", Units:"kg");
Moles_Supernate     AS  holdup_mol      (Description:"Supernate holdup moles", Units:"kmol");
Volume_Supernate    AS  volume          (Description:"Supernate holdup volume", Units:"m3");

Mass_Solids         AS  holdup_mass     (Description:"Solids holdup mass", Units:"kg");
Moles_Solids        AS  holdup_mol      (Description:"Solids holdup moles", Units:"kmol");
Volume_Solids       AS  volume          (Description:"Solids holdup volume", Units:"m3");

Volume_Cooling      AS  volume          (Description:"Cooling volume per coil bundle", Units:"m3", Fixed, 0.489);

DensMass            AS  dens_mass       (Description:"Tank holdup density", Units:"kg/m3");
DensMass_Supernate  AS  dens_mass       (Description:"Supernate density", Units:"kg/m3");
DensMass_Solids     AS  dens_mass       (Description:"Solids density", Units:"kg/m3");

DensMol_Cooling     AS  dens_mol        (Description:"Cooling molar density", Units:"kmol/m3", Fixed, 55.56);

Q_Cooling           AS  enthflow        (Description:"Cooling heat", Units:"GJ/hr");
Q_Gamma             AS  enthflow        (Description:"Radioactive decay heat", Units:"GJ/hr");

cp_mol_Cooling      AS  cp_mol          (Description:"Cooling molar heat capacity", Units:"kJ/kmol/K");

MW                  AS  molweight       (Description:"Holdup MW");
MW_Supernate        AS  molweight       (Description:"Supernate MW");
MW_Solids           AS  molweight       (Description:"Solids MW");

h                   AS  enth_mol        (Description:"Tank holdup molar enthalpy", Units:"GJ/kmol");
h_Supernate         AS  enth_mol        (Description:"Supernate molar enthalpy", Units:"GJ/kmol");
h_Solids            AS  enth_mol        (Description:"Solids molar enthalpy", Units:"GJ/kmol");
```

```
      HeatTransCoef_Solids AS  heat_trans_coeff(Description:"Solids heat transfer coefficient", Units:"kW/m2/K", Fixed,
                                                0.16981);
      HeatTransCoef       AS  heat_trans_coeff(Description:"Cooling heat transfer coefficient", Units:"kW/m2/K", Fixed,
                                                0.03774);

      sfrac               AS  ratio_          (Description:"True solid to true liquid mole ratio");
      lfrac               AS  ratio_          (Description:"True liquid to apparent liquid mole ratio");

      Drain_Signal        AS  notype          (Description:"ON/OFF Drain signal", Fixed, 1);
      Feed_Flag           AS  notype          (Description:"Feed flag");

      Rate(MainSet)       AS  reaction        (Description:"Molar reaction rate", Units:"kmol/hr/m3", Fixed, 0);
      RxnH(MainSet)       AS  enthflow        (Description:"Component reaction heat flow", Units:"GJ/hr", Fixed, 0);

      z(MainSet)          AS  molefraction    (Description:"Tank holdup mole fraction", Units:"kmol/kmol", Initial);
      z_Supernate(MainSet) AS  molefraction   (Description:"Supernate mole fraction", Units:"kmol/kmol");
      z_Solids(MainSet)   AS  molefraction    (Description:"Solids mole fraction", Units:"kmol/kmol");

      z_apparent(AqueousSet)AS  molefraction  (Description:"Apparent tank holdup mole fraction", Units:"kmol/kmol");
      z_true(AqueousSet)  AS  molefraction    (Description:"True tank holdup mole fraction", Units:"kmol/kmol");

      z_Cooling(WaterSet) AS  molefraction    (Description:"Cooling mole fraction", Units:"kmol/kmol", Fixed, 1);

      MWs(AqueousSet)     AS HIDDEN molweight (Description:"Component MW");

      RadMoles(RadSet)    AS  holdup_mol      (Description:"Holdup rad holdup moles", Units:"kmol", Initial, 0);
      RadMoles_Supernate(RadSet) AS holdup_mol      (Description:"Supernate rad holdup moles", Units:"kmol");
      RadMoles_Solids(RadSet)    AS  holdup_mol     (Description:"Solids rad holdup moles", Units:"kmol");

      radconc(RadSet)     AS  conc_mole       (Description:"Tank holdup rad concentration", Units:"mol/L");
      radconc_Supernate(RadSet) AS conc_mole        (Description:"Supernate rad concentration", Units:"mol/L");
      radconc_Solids(RadSet)     AS  conc_mole      (Description:"Solids rad concentration", Units:"mol/L");

// SUBMODEL
      Precip              AS  PrecipComp      (Description:"Get a name set of components to precipitate");

// PORTS
      InPort              AS  INPUT MULTIPORT OF Main      (Description:"Multiple liquid feed streams");
      OutPort             AS  OUTPUT Main     (Description:"Supernate transfer stream", ComponentList: MainSet);

      DestTank_FillStatus_Signal AS INPUT control_signal (Description:"Destination tank fill status signal", Fixed);
      FeedFillStatus_Signal      AS  OUTPUT control_signal  (Description:"Tank feed fill status signal", Fixed, 1);
      RecycleFillStatus_Signal AS OUTPUT control_signal  (Description:"Tank recycle fill status signal", Fixed, 1);
```

```
// ASSIGNMENTS

PrecipCompSet :      Precip.PrecipCompSet;
PrecipCompName       :      Precip.PrecipCompName;
kW2GJperHr    :      3.6E-3;
Tank_Factor   :      523.;
Coil_number   :      5;
   // Free DestTank_FillStatus_Signal if it is connected
   If DestTank_FillStatus_Signal.IsConnected Then
      DestTank_FillStatus_Signal.Spec : Free;
   Endif

// EQUATIONS

// Mole balance
EQ01:  $Moles = SIGMA(InPort.Connection.FMol) - OutPort.FMol + Volume * SIGMA(Rate);
EQ01a: Mass = Moles * MW;
EQ01b: Volume * DensMass = Mass;
EQ01c: FV_In = SIGMA(InPort.Connection.FV);

// Component mole balance
EQ02:  FOR i IN MainSet DO
          z(i) * $Moles + Moles * $z(i) =
             SIGMA(InPort.Connection.FMol * InPort.Connection.z(i)) -
                (OutPort.FMol * OutPort.z(i)) + Volume * Rate(i);
       ENDFOR
       // Rad components
EQ02a: FOR i IN RadSet DO
          $RadMoles(i) = SIGMA(InPort.Connection.FRad(i)) - OutPort.FRad(i);
       ENDFOR
EQ02b: radconc * Volume = RadMoles;

// Energy Balance
EQ03:  Moles * $h + h * $Moles =
          SIGMA(InPort.Connection.FMol*InPort.Connection.h) -
             (OutPort.FMol * OutPort.h) + SIGMA(RxnH) - Q_Cooling + Q_Gamma;
EQ03a: Q_Cooling = kW2GJperHr * HeatTransCoef * Coil_Area * (T - T_Cooling);
EQ03b: Q_Gamma = 0;

// Calculation to split holdup into Supernate (Liquid phase) and Salt (Solid phase)

       // Set up z_apparent to match z, but add missing species from "Aqueous" component list
       // with zero mole fraction
EQ04:  z_apparent(MainSet)  =  z;
```

```
EQ04a: z_apparent(AqueousSet - MainSet)  =  0;

       // Supernate phase
EQ05:  Moles_Supernate = Moles - Moles_Solids;
EQ05a: Mass_Supernate = Moles_Supernate * MW_Supernate;
EQ05b: Volume_Supernate * DensMass_Supernate = Mass_Supernate;
EQ05c: z_Supernate * Moles_Supernate = z * Moles - z_Solids * Moles_Solids;
EQ05d: RadMoles_Supernate * Moles  = RadMoles * Moles_Supernate;
EQ05e: radconc_Supernate * Volume_Supernate = RadMoles_Supernate;

       // Solids phase
EQ06:  Moles_Solids = sfrac * lfrac * Moles;
EQ06a: Mass_Solids = Moles_Solids * MW_Solids;
EQ06b: Volume_Solids * DensMass_Solids = Mass_Solids;
EQ06c: FOR i IN SolidsSet DO
          IF sfrac > 1E-6 THEN
             z_Solids(PrecipCompName(i)) * SIGMA(z_true(SolidsSet)) = z_true(i);
          ELSE
             z_Solids(PrecipCompName(i)) = 0.;
          ENDIF
       ENDFOR
EQ06d: z_Solids(MainSet-PrecipCompSet) = 0.;
EQ06e: RadMoles_Solids = RadMoles - RadMoles_Supernate;
EQ06f: IF sfrac > 1E-6 THEN
          radconc_Solids * Volume_Solids = radconc * Volume - radconc_Supernate * Volume_Supernate;
       ELSE
          radconc_Solids = 0.;
       ENDIF

// Cooling system
EQ07:  FMol_Cooling_Total = Coil_Number * FV_Cooling * DensMol_Cooling;
EQ07a: DensMol_Cooling * (Volume_Cooling * Coil_Number) * (cp_mol_Cooling * 1E-6) * $T_Cooling =
          FMol_Cooling_Total * (cp_mol_Cooling * 1E-6) * (T_Cooling_In - T_Cooling_Out) + Q_Cooling;
EQ07b: T_Cooling = 0.5 * (T_Cooling_In + T_Cooling_Out);

// Level and heat transfer area calculations
EQ08:  Total_Level = Supernate_Level + Solids_Level;
EQ08a: Supernate_Level * Tank_Factor = Volume_Supernate;
EQ08b: Solids_Level * Tank_Factor = Volume_Solids;
       // Coil heat transfer area
EQ08c: IF (Total_Level <= Min_Coil_Depth) THEN
          Coil_Area = 0.;
       ELSE
          Coil_Area * (High_Level_Limit - Min_Coil_Depth) =
```

```
            Bundle_Area * Coil_Number * (Total_Level - Min_Coil_Depth);
       ENDIF;
EQ08d: High_Decant_Level = High_Level_Limit;
EQ08e: IF Heel_Level > (Solids_Level + Min_Supernate_Depth) THEN
          Low_Decant_Level = Heel_Level;
       ELSE
          Low_Decant_Level = Solids_Level + Min_Supernate_Depth;
       ENDIF
EQ08f: WorkingSpace_Level = High_Level_Limit - (WorkingSpace_Volume / Tank_Factor);

// Decant logic
// Drain_Signal is set by Task FeedTankDrain

// Fill logic
// FeedFillStatus_Signal is set by Task FeedTankFeedFill
// RecycleFillStatus_Signal is set by Tank FeedTankRecycleFill

// Outlet stream
       // Supernate transfer stream
EQ09:  OutPort.FMol * MW_Supernate = OutPort.F;
EQ09a: OutPort.F = OutPort.FV * DensMass_Supernate;
EQ09b: OutPort.FV = DestTank_FillStatus_Signal * Drain_Signal * FV_Decant;
EQ09c: OutPort.z = z_Supernate;
EQ09d:  OutPort.w * SIGMA(z_Supernate*MWs) = z_Supernate * MWs;
EQ09e: OutPort.T = T;
EQ09f: OutPort.P = P;
EQ09g: OutPort.h = h_Supernate;
EQ09h: OutPort.FRad = OutPort.FV * radconc_Supernate;
EQ09i: OutPort.radconc = radconc_Supernate;


// PROCEDURES
EQ20:   CALL(MWs) = pMolWeights() AqueousSet;                              // Component MWs
EQ20a:       CALL(MW) = pMolWeight(z) MainSet;                             // Tank holdup apparent molecular weight
EQ20b: CALL(MW_Supernate) = pMolWeight(z_Supernate) MainSet;               // Supernate molecular weight
EQ20c: CALL(MW_Solids) = pMolWeight(z_Solids) MainSet;                     // Solids molecular weight
EQ20d: CALL(DensMass) = pDens_Mass_Liq(T, P, z) MainSet;                   // Tank holdup density
EQ20e: CALL(DensMass_Supernate) = pDens_Mass_Liq(T, P, z_Supernate) MainSet;    // Supernate density
EQ20f: CALL(DensMass_Solids) = pDens_Mass_Sol(T, P, z_Solids) MainSet;     // Solids density
EQ20g: CALL(h) = pEnth_Mol_Liq(T, P, z) MainSet;                           // Tank holdup molar enthalpy
EQ20h: CALL(h_Supernate) = pEnth_Mol_Liq(T, P, z_Supernate) MainSet;       // Supernate molar enthalpy
EQ20i: CALL(h_Solids) = pEnth_Mol_Sol(T, P, z_Solids) MainSet;             // Solids molar enthalpy
EQ20j: CALL(cp_mol_Cooling) = pCp_Mol_Liq(T_Cooling, P, z_Cooling);        // Cooling molar heat capacity
EQ20k:       CALL(z_true, sfrac, lfrac) = pTrueComp (T, P, z_apparent) AqueousSet;    // Convert tank holdup from apparent
```

```
                                        // to true components

End
```

JetDilution Model

```
Model JetDilution
/*     WESTINGHOUSE SAVANNAH RIVER COMPANY
       INTEGRATED FLOWSHEET MODEL FOR THE HLW SYSTEM
       THE 2H & 3H EVAPORATOR FLOWSHEET MODELS

       Author:      T. Hang, SRTC
       Version Date: 18 February 2002
       Description:  This module models the operation of a steam jet to transfer waste

       Design Inputs/Outputs:
          Fixed Parameters/Variables
              Diltion_factor      Specified dilution factor, %
              Jet_ON              JD ON/OFF flag (1: JD ON, 0: JD OFF)

          Input Streams
              InPortLiquid        Liquid inlet stream
              InPortSteam         Steam inlet stream

          Output Streams
              OutPort             Mixed stream

       Revision History:
             Rev No Date          Author              Description
             ------ ----------    --------------      ------------------------------------------------------------------------

*/

// PARAMETERS & VARIABLES

       MainSet             AS  ComponentListName("Default");// Main component list
       WaterSet            AS  ComponentListName("Water");         // Water component list
       RadSet              AS  ComponentListName("Rads");          // Rads component list

       F_Steam             AS  flow_mass       (Description:"Steam mass flow", Units:"kg/hr");
       DensMass            AS  dens_mass       (Description:"Mixture density", Units:"kg/m3");
       dilution_factor     AS  ratio_          (Description:"Dilution factor, %", Fixed, 4);
       Jet_ON              AS  notype          (Description:"JD ON/OFF flag", Fixed, 1);

       MWs(MainSet)        AS  HIDDEN molweight (Description:"Component MWs");

// PORTS
```

```
      InPortLiquid        AS  INPUT Main      (Description:"Liquid stream", ComponentList: MainSet);
      InPortSteam         AS  INPUT Vapor     (Description:"Steam", ComponentList: WaterSet);
      OutPort             AS  OUTPUT Main     (Description:"Mixed stream", ComponentList: MainSet);

// EQUATIONS

// Determine mass flow rate of steam jet based on dilution factor
// Jet_ON =  1 (Steam jet on)
//           0 (Steam jet off ---> use pump only, no jet dilution)
EQ01:  InPortSteam.F = F_Steam;
EQ01a: IF Jet_ON > 0 THEN
          (dilution_factor/100) * InPortLiquid.F = F_Steam;
       ELSE
          F_Steam = 0.;
       ENDIF

// Overall mole balance
EQ02:  OutPort.FMol = InPortLiquid.FMol + InPortSteam.FMol;
EQ02a:  OutPort.F = InPortLiquid.F + InPortSteam.F;
EQ02b: IF InPortLiquid.FMol > 0.1 THEN
          OutPort.FV * DensMass = OutPort.F;
       ELSE
          OutPort.FV = 0;
       ENDIF

// Component mole balances
EQ03:  IF InPortLiquid.FMol > 0.1 THEN
          OutPort.z(WaterSet) * OutPort.FMol = InPortLiquid.z * InPortLiquid.FMol + InPortSteam.y * InPortSteam.FMol;
          OutPort.z(MainSet-WaterSet) * OutPort.FMol = InPortLiquid.z * InPortLiquid.FMol;
       ELSE
          OutPort.z(WaterSet) = 1.;
          OutPort.z(MainSet-WaterSet) = 0.;
       ENDIF
EQ03a:  OutPort.w * SIGMA(OutPort.z * MWs) = OutPort.z * MWs;

       // Rad Components
EQ03b:  OutPort.FRad = InPortLiquid.FRad;
EQ03c:       IF InPortLiquid.FMol > 0.1 THEN
          OutPort.radconc * OutPort.FV = OutPort.FRad;
       ELSE
          OutPort.radconc = 0.;
       ENDIF
```

```
// Energy balance
EQ04:  IF InPortLiquid.FMol > 0.1 THEN
          OutPort.h * OutPort.FMol = InPortLiquid.h * InPortLiquid.FMol + InPortSteam.h * InPortSteam.FMol;
       ELSE
          OutPort.T = InPortLiquid.T;
       ENDIF
EQ04a: OutPort.P = InPortLiquid.P;

// Output stream

// PROCEDURES
EQ20:  CALL(MWs) = pMolWeights() MainSet;                                    // Component MWs
EQ20a: CALL(DensMass) = pDens_Mass_Liq(OutPort.T, OutPort.P, OutPort.z);     // Liquid mass density
EQ20b: CALL(OutPort.h) = pEnth_Mol_Liq(OutPort.T, OutPort.P, OutPort.z);     // Liquid molar enthalpy

End
```

Separator Model

```
Model Separator
/*     WESTINGHOUSE SAVANNAH RIVER COMPANY
       INTEGRATED FLOWSHEET MODEL FOR THE HLW SYSTEM
       THE 2H & 3H EVAPORATOR FLOWSHEET MODELS

       Author:      T. Hang, SRTC
       Version Date:14 December 2001
       Description: This module models the operation of the VL separator pot

       Design Inputs/Outputs:
          Fixed Parameters/Variables
              P            Specified operating pressure, bar
              TRef         Ambient temperature, C
              UA           Heat transfer coefficient * Surface area

          Input Streams
              I1           Evaporator concentrate stream
              I2           Steam lift stream

          Output Streams
              O1           Liquid stream
              O2           Vapor stream

       Revision History:
              Rev No Date         Author              Description
              ------ ----------   --------------      ------------------------------------------------------------------
              001    02/09/2002   M.B. Gorensek       Combine concentrate and steam in input port I1, eliminating input
                                                         port I2; change I1 to MainSimple type; add explanatory comments
                                                         throughout
              002    02/27/2002   T. Hang             Revise EQ04h to avoid indeterminate problem when O1.FV equals 0

*/

// PARAMETERS & VARIABLES

       MainSet             AS  ComponentListName("Default");       // Default component list
       GasSet              AS  ComponentListName("Vapor");         // Vapor component list
       AqueousSet          AS  ComponentListName("Aqueous");       // True component list
       RadSet              AS  ComponentListName("Rads");          // Rads component list

       //FMol              AS  flow_mol       (Description:"Molar flow", Units:"kmol/hr");               // Rev 001
```

```
      T                   AS  temperature     (Description:"Temperature", Units:"C");
      TRef                AS  temperature     (Description:"Ambient temperature", Units:"C", Fixed, 25);

      P                   AS  pressure        (Description:"Pressure", Units:"bar", Fixed, 1.0133);

      DensMass            AS  dens_mass       (Description:"Liquid density", Units:"kg/m3");
      Vap_DensMass        AS  dens_mass       (Description:"Vapor density", Units:"kg/m3");

      Q                   AS  enthflow        (Description:"Heat loss", Units:"GJ/hr");

      h                   AS  enth_mol        (Description:"Misture molar enthalpy", Units:"GJ/kmol");
      hl                  AS  enth_mol_liq    (Description:"Liquid molar enthalpy", Units:"GJ/kmol");
      hv                  AS  enth_mol_vap    (Description:"Vapor molar enthalpy", Units:"GJ/kmol");

      MW                  AS  molweight       (Description:"Liquid MW");
      MW_Vap              AS  molweight       (Description:"Vapor MW");

      vapfrac             AS  vapfraction     (Description:"Vapor fraction", Units:"kmol/kmol");

      UA                  AS  notype          (Description:"Heat transfer coefficent * Area", Fixed, 0.005);

      x(ComponentList)    AS  molefraction    (Description:"Liquid mole fraction", Units:"kmol/kmol");
      y(ComponentList)    AS  molefraction    (Description:"Vapor mole fraction", Units:"kmol/kmol");
      //z(ComponentList)  AS  molefraction    (Description:"Mixture mole fraction", Units:"kmol/kmol");   // Rev 001
      w(ComponentList)    AS  massfraction    (Description:"Liquid mass fraction", Units:"kg/kg");

      MWs(ComponentList)  AS  HIDDEN molweight      (Description:"Component MW");

// PORTS
      //I1                AS  INPUT Main        (Description:"Evaporator concentrate", ComponentList: MainSet);// Rev 001
      I1                  AS  INPUT MainSimple  (Description:"Evaporator concentrate", ComponentList: MainSet);// Rev 001
      //I2                AS  INPUT Vapor           (Description:"Steam lift", ComponentList: GasSet);        // Rev 001
      O1                  AS  OUTPUT Main           (Description:"Liquid outlet", ComponentList: MainSet);
      O2                  AS  OUTPUT Vapor    (Description:"Vapor", ComponentList: GasSet);

// EQUATIONS

// Stream mixing with heat loss
//EQ01:   FMol = I1.FMol + I2.FMol;                                                          // Rev 001
//EQ01a:  z(ComponentList-GasSet) * FMol = I1.z * I1.FMol;                                   // Rev 001
//EQ01b:  z(GasSet) * FMol = I1.z * I1.FMol + I2.y * I2.FMol; */                             // Rev 001
//EQ01c:  h * FMol = I1.h * I1.FMol + I2.h * I2.FMol - Q;                                    // Rev 001
EQ01c: h * I1.FMol = I1.h * I1.FMol - Q;                                                   // Rev 001
```

```
EQ01d: Q = UA * (T - TRef);

// Overall mole balance
//EQ02:   FMol = O1.FMol + O2.FMol;                                          // Rev 001
EQ02:  I1.FMol = O1.FMol + O2.FMol;                                          // Rev 001

// Energy balance
//EQ03:      h * FMol = O1.h * O1.FMol + O2.h * O2.FMol;
EQ03:  h * I1.FMol = O1.h * O1.FMol + O2.h * O2.FMol;                        // Rev 001

// Output streams
        // Liquid stream
EQ04:  O1.F = O1.FMol * MW;
EQ04a: O1.FV * DensMass = O1.F;
EQ04b: O1.w * SIGMA(x*MWs) = x * MWs;
EQ04c: O1.z = x;
EQ04d: O1.T = T;
EQ04e: O1.P = P;
EQ04f: O1.h = hl;
EQ04g: O1.FRad = I1.FRad;
//EQ04h:  O1.radconc * O1.FV = O1.FRad;
EQ04h: IF O1.F > 1E-3 THEN                                                   // Rev 002
         O1.radconc * O1.FV = O1.FRad;                                       // Rev 002
       ELSE                                                                  // Rev 002
         O1.radconc = 0.;                                                    // Rev 002
       ENDIF                                                                 // Rev 002

        // Vapor stream
//EQ05:      O2.FMol = FMol * vapfrac;                                       // Rev 001
EQ05:  O2.FMol = I1.FMol * vapfrac;                                          // Rev 001
EQ05a: O2.F = O2.FMol * MW_Vap;
EQ05b: O2.FV * Vap_DensMass = O2.F;
EQ05c: O2.w * SIGMA(y*MWs) = y * MWs;
EQ05d: O2.y = y;
EQ05e: O2.T = T;
EQ05f: O2.P = P;
EQ05g: O2.h = hv;

// PROCEDURES
EQ020:  CALL(MWs) = pMolWeights();
EQ020a: CALL(MW) = pMolWeight(x);                        // Liquid molecular weight
EQ020b: CALL(DensMass) = pDens_Mass_Liq(T, P, x);        // Liquid mass density
EQ020c: CALL(MW_Vap) = pMolWeight(y);                    // Vapor molecular weight
EQ020d: CALL(Vap_DensMass) = pDens_Mass_Vap(T, P, y);    // Vapor mass density
```

```
//EQ020e: CALL(y, x, vapfrac, hv, hl) = pFlash(T, P, z);    // Flash at given T and P                    // Rev 001
EQ020e: CALL(y, x, vapfrac, hv, hl) = pFlash(T, P, I1.z);   // Flash at given T and P                    // Rev 001
End
```

PrecipComp Submodel

```
Model PrecipComp

// PARAMETERS & VARIABLES
SolidsSet as ComponentListName("Solids");      // Solids Component List
PrecipCompSet as StringSet                     (Description:"Set of components to precipitate");
PrecipCompName(SolidsSet) as StringParameter   (Description:"Components to precipitate");

// ASSIGNMENTS
PrecipCompName("NAALO(S)"): "NAALO2";
PrecipCompName("NACL(S)"): "NACL";
PrecipCompName("NACO3(S)"): "NA2CO3";
PrecipCompName("NAF(S)"): "NAF";
PrecipCompName("NANO2(S)"): "NANO2";
PrecipCompName("NANO3(S)"): "NANO3";
PrecipCompName("NAOH(S)"): "NAOH";
PrecipCompName("NAPO4(S)"): "NA3PO4";
PrecipCompName("NASO4(S)"): "NA2SO4";
PrecipCompSet:(["NAALO2", "NACL",
                "NA2CO3", "NAF",
                "NANO2" , "NANO3",
                "NAOH"  , "NA3PO4",
                "NA2SO4"]);

End
```

Feed Stream Model

```
Stream Feed
/*     WESTINGHOUSE SAVANNAH RIVER COMPANY
       INTEGRATED FLOWSHEET MODEL FOR THE HLW SYSTEM
       THE 2H & 3H EVAPORATOR FLOWSHEET MODELS

       Author:      T. Hang, SRTC
       Version Date:12 December 2001
       Description: This stream type handles feed stream

       Design Inputs/Outputs:
          Input Parameters
              FV_Feed      Nominal volumetric feed flow, m3/hr
              Feed_Signal  ON/OFF Feed signal
              T            Specified feed temperature, C
              P            Specified feed pressure, bar
              conc         Array of specified feed concentration, mol/L

          Input Stream/Signal
              DestTank_FillStatus_Signal
                           Destination tank fill status signal

          Output Stream/Signal
              O1           Liquid output stream of Port type Main

       Revision History:
             Rev No Date         Author              Description
             ------ ----------   --------------      ------------------------------------------------------------------
             001    02/26/2002   T. Hang             Calculate feed transfer volume
             002    03/04/2002   T. Hang             Add task counter TaskNum for the scheduled tasks

*/

// PARAMETERS & VARIABLES

       RadSet              AS  ComponentListName("Rads");          // Rads Component List

       FV_Feed             AS  flow_vol        (Description:"Nominal volumetric flow", Units:"m3/hr", Fixed);
       FV                  AS  flow_vol        (Description:"Volumetric flow", Units:"m3/hr");
       FMass               AS  flow_mass       (Description:"Mass flow", Units:"kg/hr");
       FMol                AS  flow_mol        (Description:"Molar flow", Units:"kmol/hr");
```

```
      T                   AS  temperature     (Description:"Temperature", Units:"C", Fixed);
      P                   AS  pressure        (Description:"Pressure", Units:"bar", Fixed);
      Volume              AS  volume          (Description:"Holdup volume", Units:"m3", Initial, 0);      // Rev 001
      Liq_DensMass        AS  dens_mass       (Description:"Density", Units:"kg/m3");

      h                   AS  enth_mol        (Description:"Molar enthalpy", Units:"GJ/kmol");

      MW                  AS  molweight       (Description:"Molecular weight", Units:"kg/kmol");

      Feed_Signal         AS  notype          (Description:"ON/OFF Feed signal", Fixed, 0);
      TaskNum             AS  pos_small       (Description:"Counter for feed task number", Fixed, 1);     // Rev 002

      conc(ComponentList) AS  conc_mole       (Description:"Component concentration", Units:"mol/L", Fixed);
      w(ComponentList)    AS  massfraction    (Description:"Mass fraction", Units:"kg/kg");
      z(ComponentList)    AS  molefraction    (Description:"Mole fraction", Units:"kmol/kmol");
      MWs(ComponentList)  AS  HIDDEN molweight(Description:"Component MW", Units:"kg/kmol");

      FRad(RadSet)        AS  flow_mol        (Description:"Component rad molar flow", Units:"kmol/hr");
      radconc(RadSet)     AS  conc_mole       (Description:"Rad concentration", Units:"mol/L", Fixed);

// PORTS

      O1                  AS  OUTPUT Main     (Description:"Liquid stream");

      DestTank_FillStatus_Signal AS  INPUT control_signal (Description:"Destination tank fill status signal", Fixed);

// ASSIGNMENTS

      // Free DestTank_FillStatus_Signal if it is connected
      If DestTank_FillStatus_Signal.IsConnected Then
         DestTank_FillStatus_Signal.Spec : Free;
      Endif

// EQUATIONS
EQ01:  z * SIGMA(conc) = conc;
EQ02:  w * SIGMA(z*MWs) = z*MWs;
EQ03:  FV = DestTank_FillStatus_Signal * Feed_Signal * FV_Feed;
EQ04:  FMass = FV * Liq_DensMass;
EQ05:  FMol * MW = FMass;
EQ06:  FRad = FV * radconc;
EQ07:  O1.F = FMass;
EQ08:  O1.FMol = FMol;
EQ09:  O1.FV = FV;
EQ10:  O1.w = w;
```

```
EQ11:  O1.z = z;
EQ12:  O1.T = T;
EQ13:  O1.P = P;
EQ14:  O1.h = h;
EQ15:  O1.FRad = FRad;
EQ16:  O1.radconc = radconc;

       // Transfer volume
EQ17:  $Volume = FV;                                                    // Rev 001

// PROCEDURES
EQ20:  CALL(MWs) = pMolWeights();
EQ20a: CALL(MW) = pMolWeight(z);
EQ20b: CALL(Liq_DensMass) = pDens_Mass_Liq(T, P, z);
EQ20c: CALL(h) = pEnth_Mol_Liq(T, P, z);

End
```

Steam Stream Model

```
Stream Steam
/*     WESTINGHOUSE SAVANNAH RIVER COMPANY
       INTEGRATED FLOWSHEET MODEL FOR THE HLW SYSTEM
       THE 2H & 3H EVAPORATOR FLOWSHEET MODELS

       Author:       T. Hang, SRTC
       Version Date: 12 December 2001
       Description:  This stream type handles steam streams

       Design Inputs/Outputs:
          Fixed Parameters/Variables
              FMass         Mass flow, kg/hr
              P             Specified steam pressure, bar
              Steam_Signal  ON/OFF steam signal

          Output Streams
              O1            Output of Port type Vapor

       Revision History:
              Rev No Date         Author              Description
              ------ ----------   --------------      ------------------------------------------------------------------
              001    02/22/2002   T. Hang              Add Steam_Signal to turn steam ON/OFF

*/

// PARAMETERS & VARIABLES

       WaterSet            AS  ComponentListName("Water");         // Water Component List

       FV                  AS  flow_vol        (Description:"Volumetric flow", Units:"m3/hr");
       FMass               AS  flow_mass       (Description:"Mass flow", Units:"kg/hr", Fixed);
       FMol                AS  flow_mol        (Description:"Molar flow", Units:"kmol/hr");

       T                   AS  temperature     (Description:"Temperature", Units:"C");
       P                   AS  pressure        (Description:"Pressure", Units:"bar", Fixed);
       Vap_DensMass        AS  dens_mass       (Description:"Density", Units:"kg/m3");

       h                   AS  enth_mol        (Description:"Molar enthalpy", Units:"GJ/kmol");

       MW                  AS  molweight       (Description:"Molecular weight", Units:"kg/kmol");
```

```
      Steam_Signal        AS  notype          (Description:"ON/OFF steam signal", Fixed, 1);              // Rev 001

      w(WaterSet)         AS  massfraction    (Description:"Mass fraction", Units:"kg/kg", Fixed, 1);
      y(WaterSet)         AS  molefraction    (Description:"Mole fraction", Units:"kmol/kmol");
      MWs(WaterSet)       AS  HIDDEN molweight (Description:"Component MW", Units:"kg/kmol");

// PORTS

      O1                  AS  OUTPUT Vapor    (Description:"Steam", ComponentList: WaterSet);

// EQUATIONS
EQ01:  y * SIGMA(w/MWs) = w/MWs;
EQ02:  FV * Vap_DensMass = FMass;
EQ03:  FMol * MW = FMass;
//EQ04:      O1.F = FMass;
EQ04:  O1.F = Steam_Signal * FMass;                                                                        // Rev 001
//EQ05:      O1.FV = FV;
EQ05:  O1.FV = Steam_Signal * FV;                                                                          // Rev 001
EQ06:  O1.w = w;
//EQ07:      O1.FMol = FMol;
EQ07:  O1.FMol = Steam_Signal * FMol;                                                                      // Rev 001
EQ08:  O1.y = y;
EQ09:  O1.T = T;
EQ10:  O1.P = P;
EQ11:  O1.h = h;

// PROCEDURES
EQ20:  CALL(MWs) = pMolWeights() WaterSet;
EQ20a: CALL(MW) = pMolWeight(y);
EQ20b: CALL(T) = pDewt(P, y);
EQ20c: CALL(Vap_DensMass) = pDens_Mass_Vap(T, P, y);
EQ20d: CALL(h) = pEnth_Mol_Vap(T, P, y);

End
```