Project
February 2000

# TOTAL FLUX & SCALANT PROGRAM

## A Membrane System Design Assistant

By

Thomas D. Wolfe
PerLorica Inc.

Project 9C-R193-NTSX

Project Officer

Thomas F. Speth, Ph.D., P.E.
Environmental Engineer
Water Supply and Water Resources Division
26 W. Martin Luther King Drive
Cinncinati, OH 45268

Water Supply and Water Resources Division
U.S. Environmental Protection Agency
26 W. Martin Luther King Drive
Cinncinati, OH 45268

## Notice



## Abstract

This works covers the development of a program – the Total Flux and Scalant program (TFS) - for designing a generic membrane system using Reverse Osmosis or Nanofiltration membranes. In addition to design of a membrane system the program calculates solubility conditions for the feed and concentrate streams using a variety of methods. The User is provided a facility to input and balance a water analysis using a variety of input methods, calculate a carbonate equilibrium based on pH and temperature, and specify the use of acid or a softening resin to "pretreat" the water before the membrane system.

The membrane system design allows the user to utilize "generic" membrane elements or create new types of elements and membranes specific to their own application from either data sheets or empirical data. The design function calculates pressures, pressure drops, and the ion by ion salt passage through the membranes. The program returns the ionic concentration of each stream and calculates the scaling index for many potentially scaling species using either ASTM methods or methods based on ion complex calculations closely resembling the traditional programs used in the geological sciences.

The program requires a PC with Excel 97 or higher to run. It is designed so that the reasonably experienced Excel user can understand the calculations taking place, particularly as regards scaling and equilibrium conditions.

# Table of Contents

## Total Flux and Scalant Program

Table of Figures

# Total Flux and Scalant Program

Project 9C-R193-NTSX

# Final Report

## Executive Summary & Introduction

This project covers the development of computer program written in Microsoft Excel to assist the user in the evaluation of reverse osmosis and nanofiltration process applications. This program is capable of receiving a water analysis and evaluating the analysis as to its self consistency – i.e. charge balance and carbonate/pH equilibrium. Scaling potential of many common compounds of interest in membrane systems are calculated, using either ASTM standard methods or a similar to that used by WaterEQ or Phreeqc based on ion complex formations.

With a validated water analysis, the user may proceed to design a membrane system using “generic” membrane types – i.e. Thin File Composite (TFC), Cellulose Acetate (CA), or Nanofiltration (NF). Physical element configurations are included for most standard sizes and a facility for customizing these sizes and membrane performance parameters is included for skilled and knowledgeable users. A typical membrane system comprising multiple banks with or without concentrate recycle can be specified. Once flow rates and recovery values are specified, a detailed calculation on a bank by bank basis is performed. This returns analytical, pressure, and flow information at each bank in the system. The program can then optionally or automatically calculate the scaling conditions in the feed, feed plus recycle, and concentrate streams. Thus the user is provided a generic facility for designing membrane systems and calculating scale potential throughout the specified system. The program also contains a simple database facility which allows the user to store analyses and project information within the spreadsheet itself.

This program, written in Excel with use of the Visual Basic programming language provides a skilled user an insight into the actual calculations used for the design of membrane systems and provides an easy to transport framework for exporting scaling calculations to other venues. This program also provides a spreadsheet ready subsystem which translate the mostly graphical methods provided by ASTM into Excel based formulas.

## Acknowledgments

Dr. William Bourcier of Lawrence Livermore National Laboratory assisted greatly in the development of the solubility methodology used in the program. His contributions were essential to the development of the robust solubility calculations.

Special thanks to Dr. Thomas Speth of the US EPA for his continued support during this project and many insightful comments.

# Calculation Methodology

## General Comments

The Program Flow is designed as follows:

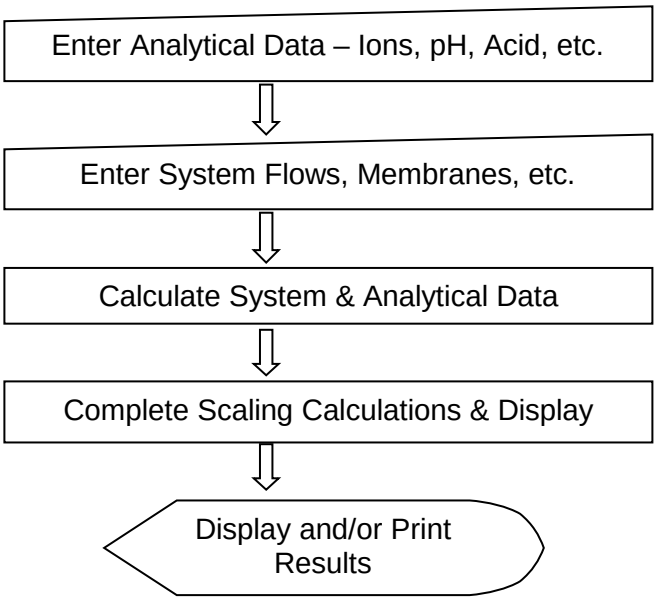



Figure 1 – General Program Flow

Since this is a spreadsheet, the data and calculations are organized by type and function onto different "sheets" or tabs. Buttons on the sheets provide an alternate means of navigation between sections.

Where it was reasonably possible the spreadsheet itself was used to perform the calculations. For example, the carbonate equilibria are handled in the spreadsheet itself for the feedwater entry. In addition, Langelier Saturation Index (LSI) and ASTM methods for CaSO4, SrSO4, BaSO4, and SiO2 are also handled in the spreadsheet.

***Organization of the Spreadsheet – by Sheet/Tab***

Only the first two sheets (Analysis & System Design) plus the DataBase sheet require any user interaction. The other sheets are either display sheets for results or intermediate calculation sheets.

1. **Analysis** – Sheet for entering analytical and project data. Ions, pH, acid, chlorine content. Units available are mg/l, meq/l, mg/l as $CaCO_3$, deg F, deg C, deg K. Already selected flow and recovery data also shows on this sheet if available.

   This sheet also alerts the user to a badly balanced analysis and provides a provision for the input of the free chlorine content. Free chlorine is reacted stoichiometrically with any Ferrous iron ($Fe^{2+}$) to oxidize it to Ferric iron ($Fe^{3+}$)

2. **System Design** – This sheet allows the user to select an element type and then complete the design of the membrane system. As data is entered, flow and recovery data, as well as total membrane area and number of elements and vessels are continually updated. At the bottom of the tab, an approximate TDS for concentrate and permeate is displayed. This becomes an exact value after calculation is complete. The user may specify:

   - The number of banks in the system where a bank is defined as an array of pressure vessels with common parallel feed. Figure 2 at right illustrates a 2 bank system arranged in a 4:2 array of pressure vessels.
   - Number of elements per vessel
   - Feed, recycle, concentrate, and/or permeate flow rates
   - Flow Units of measure – gpm, gpd, $m^3$/hr, liter/min, liter/hr
   - Percent Recovery, for both the total system and the internal recovery (with recycle included).

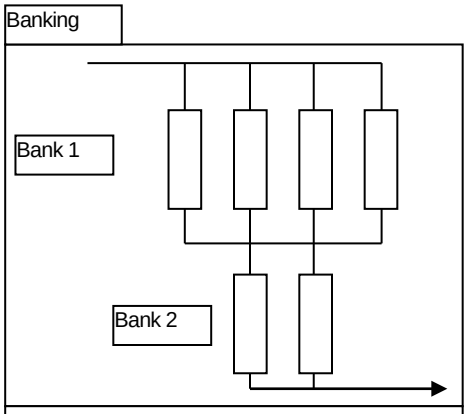


Figure 2 – Banking Description

   When data entry/system design is complete the user may "Calculate" the system to launch the Visual Basic RO/NF calculation engine and the scaling calculations via the Excel solver.

   The user may also add or edit membrane data directly from this sheet.

3. **Summary & Streams Sheets** – These sheets are "Display Only" sheets and are not meant to be modified by the user. They collect the data from the calculations, organize it, and display it in a convenient, readily printable format. All system data is available in these two sheets.

4. **DataBase** – This sheet stores project information for later recall. The user enters this sheet to locate an old project and load it into the system

5. **SoluCalcs** – This sheet handles the calculations for LSI, $CaSO_4$, $SrSO_4$, $BaSO_4$, $SiO_2$, and $CaF_2$. With the exception of $CaF_2$ all of these calculation are per the referenced ASTM method. $CaF_2$ is calculated by using the modified Debye Huckel calculation with the stream Ionic Strength. The Ksp for $CaF_2$ is corrected for temperature using the Van't Hoff method.

6. **Membrane Data** – This sheet keeps the membrane information added or edited by the user. It also keeps the data for relative salt transport numbers of the different ions relative to Cl or Na ions. Experienced users may modify the ion transport data directly on this sheet or create a new membrane type altogether. Cells AA15 to AK63 (light blue background) contain this data. The value for each ion is the salt transport ratio relative to sodium for cations and relative to chloride for anions.

7. **InfoLic** – This sheet stores the version numbers of the release and contains the general license disclaimer.

8. **Calc** – This sheet sets up and solves the complexation equations for computing the Saturation Index (SI) for the species of interest. Treated Feed (Feed after acid has been added and any free Cl2 reacted with Ferrous Iron), Net Feed (defined as Treated Feed + Recycle) and Concentrate streams are addressed. The user may interact directly on this sheet only by clicking the colored buttons.

9. **LogKs** – This large sheet contains the base data for each of the complex species evaluated. Three sets of tables are provided with spaces between them. The spaces relate to the same spacing on the Calc and ActCof sheets so that cell name references are consistent. Log Ksp's for the species are provided along with a temperature dependence polynomial. If no temperature dependence polynomial was available in the database used then the Van't Hoff method is applied. Since most data is available for 25 deg C, this provides reasonable accuracy for membrane systems, which typically operate in the 5 to 60 deg C range.

10. **ActCof** – This sheet determines the correct activity coefficient to apply to each specie – both complex and mineral. The Davies equation is used with Ionic strength to calculate the appropriate activity depending on the net charge of the components or the overall complex.

Sheets SoluCalcs, Membrane Data, Calc, LogKs, and ActCof contain much documentation as plain text on the sheet with cited references and detail explanations of the how the individual calculations are effected.

## Solubility Methods

### *ASTM Methods*

$CaSO_4$, $BaSO_4$, and $SrSO_4$ are taken from ASTM D 4692, Figures 1-3 respectively. ASTM D4993 is the source for $SiO_2$ solubility. Langelier Saturation Index (LSI) is taken from the definition of LSI in the literature, but most particularly from Stumm and Morgan (see reference bibliography). ASTM data was used for these common scale forming minerals since this the accepted industry method. Calcium Fluoride ($CaF_2$) is computed in a similar manner to the ASTM data except that the Ksp was taken from the Phreeqc database. The Van't Hoff method is used to correct for different temperatures.

The ASTM methods are essentially polynomials derived by curve fitting techniques. These provide a Ksp value as a function of temperature and ionic strength (IS). The corrected Ksp is compared to the actual ion product for the stream of interest and a percent of solubility or percent of saturation is defined as follows. This method is also used for calcium fluoride.

$$Percent\ Solubility = \left( Ion\ Product \Big/ Ksp\ corrected \right) * 100 \qquad \text{Equation 1}$$

The Langelier Saturation Index (LSI) is computed as follows:

$$LSI = pH - pHs \quad where$$
$$pHs = (pK2 - pKSo + pCa + pAlk)$$

pH is the pH of the solution, "pHs" is Langelier's calculated saturation pH. Thus a negative LSI indicates that $CaCO_3$ should not precipitate from solution.

- The values of pK2 and pKSo are the carbonate solubility products and are corrected for temperature. PK2 is corrected for ionic strength as well.

- pCa equals -log([Ca]) where [Ca] = moles/liter calcium, corrected for ionic strength using the Debye Huckel method.

- pALK equals -log([Total Alkalinity]), and Total Alkalinity = $[HCO_3]$ + $[CO_3]$ + [OH] – [H]. $[HCO_3]$ and $[CO_3]$ are also corrected for ionic strength. Langelier originally did this with charts from empirical data.

- The value for carbonate ($CO_3$) is computed from the HCO3 and pH of the stream in question.

It should be noted that LSI is not considered useful above an ionic strength of about 0.3. There are many excellent texts on LSI calculation so it will not be covered in further detail here.

***Other Minerals***

Less common other minerals which are nevertheless of interest to membrane system designers are handled by the more rigorous aqueous complexation methods typical of the WaterEQ or Phreeqc programs (WATEQ4F, Plummer et al., 1976, PHREEQE Parkhurst et al., 1980). A given solution analysis is first numerically "speciated" into the various aqueous complexes that are present using thermodynamic stability constants. The saturation state of the solid is then calculated from the true solubility product, which is not a function of ionic strength, and the ionic concentrations determined from the speciation calculation.

After a careful consideration of many possible minerals, it was decided to limit the calculations and display to the minerals of most interest in the operating temperature and operating pH range of most membrane systems – i.e. 5 to 50 deg C, and pH 4 to 10.

The minerals of interest include:

- Analcime $NaAlSi_2O_6.H_2O$
- Gibbsite - aluminum hydroxide $Al(OH)_3$
- Amorphous Al(OH)3 - aluminum hydroxide $Al(OH)_3$
- Fe(OH)3 (amorphous) - Ferric hydroxide $Fe(OH)_3$
- Goethite - Ferric hydroxide $FeO(OH)$
- Siderite - Ferrous carbonate $FeCO_3$
- Vivianite - Ferrous phosphate $Fe_3(PO_4)_2:8H_2O$
- Hydroxyapatite - Calcium hydroxy phosphate $Ca_5(PO_4)_3OH$
- Gypsum (also by ASTM methods) - Calcium sulfate $CaSO_4.2H_2O$
- Sepiolite (High pH silica foulant) - Mag Silicate $Mg_4Si_6O_{15}(OH)_2:6H_2O$

In a very simplistic sense, the approach in this case is to create a list of all possible complex species which could exist in the solution, and then calculate a series of equations describing these complexes based on known Ksp data for each complex. The “left over” ions, that is the concentrations left after the complexes are accounted for, are the concentration of the ions available to form the minerals of interest. These ion products are then compared to known Ksp values for the minerals of interest and a Solubility Index (SI) is calculated:

$$SI = Log\left(\frac{Ion\,PRODUCT}{Ksp}\right) = Log(IP) - Log(Ksp) \qquad \text{Equation 3}$$

A SI negative value suggests under-saturation while a SI positive suggests over-saturation and possible precipitation. An SI of zero of course suggests the solution is exactly at saturation.

For example, the program calculates complex concentrations for the following primary ferric ion ($Fe^{+++}$) complexes:

- $Fe^{3+} + Cl^{-} = FeCl^{++}$
- $Fe^{3+} + SO_4^{--} = FeSO4^{+}$
- $Fe^{3+} + 2OH^{-} = Fe(OH)_2^{+}$
- $Fe^{3+} + 3OH^{-} = Fe(OH)_3(aq)$
- $Fe^{3+} + HCO_3^{-} = FeHCO_3^{+}$
- $Fe^{3+} + CO_3^{--} = FeCO_3^{+}$
- $Fe^{3+} + HPO_4^{--} = FeHPO_4^{+}$
- $Fe^{3+} + H_2PO_4^{-} = FeH_2PO_4^{++}$

For another simpler example, Na and SO4 form a complex with one negative charge - i.e. $NaSO_4$. The Log(Ksp) is calculated for this complex at the solution temperature on the sheet entitled LogKs. The formula for this complex is created as follows, where $\alpha$ represents the activity of the species under consideration. Activity values are calculated for each complex and each species on the sheet ActCof, and the "Available" concentration is determined on the sheet entitled "Calc".

$$[NaSO_4] = \frac{[Na]_{available} * \alpha_{Na} * [SO_4]_{available} * \alpha_{SO4} * 10^{-\log Ksp}}{\alpha_{NaSO4}} \qquad \text{Equation 4}$$

Of course many other complexes containing both Na and $SO_4$ are also produced, and the amount available of each ion is affected by all of the other complexes. The solution to the problem must be obtained iteratively. Using the Excel built in Solver provides a convenient method for solving this problem and the program implements a series of VBA commands which setups the Solver and then runs its automatically. The remaining "available" concentrations of each ionic species - that is the real solution concentration less the amount "complexed" with the various complexes as determined by the Solver - is the concentration available for mineral formation.

For example, assume we wish to calculate the actual ion product for $CaSO_4 - 2H_2O$ in the solution to determine the scaling potential. We calculated the actual and available concentrations as follows.

| Specie | Ca(molal) | $SO_4$ (molal) |
|---|---|---|
| Actual Conc | 0.00374 | 0.01041 |
| Available Conc | 0.00292 | 0.00885 |
| Activity coefficient ($\alpha$) | 0.447 | 0.447 |

We then calculated the Log(Ksp) for $CaSO_4$-$2H_2O$ in this solution and temperature to be -4.59. The logarithm of the ion product for $CaSO_4$ based on the concentration available is compared to the Log(Ksp) to provide an SI.

$$SI = Log(0.00292 * 0.447 * 0.00885 * 0.447) - \log(Ksp) \qquad \text{Equation 5}$$

$$SI = -5.29 - (-4.59) = -0.70$$

Thus at this condition $CaSO_4$ is significantly under saturation. Since the SI is log based, the percent of saturation is $10^{(-0.7)}$ = 0.1995 or about 20%. In this example, the ASTM method, which uses a Ksp corrected for ionic strength with typical analyses yielded a value of about 21% of saturation. Thus the ASTM method compares very well and slightly conservatively in this case.

Nevertheless, it should be noted that the calculations performed using the ion complex method are performed for minerals that may actually supersaturate to a very high degree, and thus not cause problems in many membrane system applications. The aluminum and iron compounds investigated are also almost always present to some degree in surface and ground waters. This program provides the user a means to effectively determine the optimum pH conditions to minimize potential fouling from these compounds. However, the user should note that iron and aluminum compounds are usually present in the amorphous condition and the fouling or non-fouling tendencies of the compounds may be greatly dependent on the presence or absence of organic or biological components in the feedwaters.

One further note, if the user does enters "0.0" or no value for an ion, the program assigns a default minimum value of 1 ppt (1 part per trillion). This prevents division by zero and the small values are neglible in the calculations. For HCO3 however, a minimum value of 0.1 ppm is assigned so that the carbonate distribution calculations avoid division by zero. For ferric and ferrous ions the program assumes the user input distribution is correct. Free chlorine (if specified) is reacted stoichiometrically with any $Fe^{2+}$ converting it to ferric ion, $Fe^{3+}$, to the extent available. Excess free chlorine after reaction with ferrous iron is next reacted with any $NH_4$ present, removing it from consideration as an assumed $NH_3Cl$ species. Dissolved oxygen conditions are not addressed.

## RO & NF Calculation Methods

### *General Approach*

The general approach used is the same as used by most commercial membrane design program. Basically the user needs to guess a design and then a detailed calculation is made. The TFS program assists this process by continually displaying and updating an estimated feed pressure and TDS values for the user at the bottom of the System Design sheet.

With a reasonable guess in hand, the convergence to a final design is usually quite rapid. The TFS program makes one simplification over most commercial programs which greatly helps the calculation rates in that each vessel is converted to the form of one element, which directly returns

the performance for each bank in the system. TFS calculates salt flux on and ion by ion basis during each pass through the system and balances carbonates in the concentrate and permeate analyses while accounting for the free passage of $CO_2$ through the membranes.

Water flux, or more properly the flow rate, is calculated as follows, where Pi is osmotic pressure. The effective pressure or PNet is the net trans-membrane differential pressure. All values are first converted to meters, seconds, and atmospheres to make calculations simpler.

Equation 6

$$Flow(m^3/\sec) = AValue_{m/(\sec * atm)} * Area_{m\wedge 2} * \left( FeedPress_{atm} - \frac{differential_{atm}}{2} - (Pi\ Membrane - Pi\ Permeate)_{atm} \right)$$

The concentration of each ion in the permeate is found as follows. A variable, "MegDiff" is defined for each ion first which is the *effective* difference in concentration of the ion from the membrane surface to the permeate.

$$MeqDiff = BulkLMAvg\,Conc * ConcPolarization - PermConc$$ Equation 7

$$PermConc = MeqDiff * \frac{BValue * BCorrectionByIon}{PNet * AValue + BValue}$$ Equation 8

PNet is as defined above, i.e. the Feed Pressure less differential pressure less the osmotic pressure difference between the average feed stream and permeate. BcorrectionByIon is stored in the spreadsheet as a function of the membrane type selected. This the ratio of salt flux for the ion in question relative to either chloride or sodium.

Since the permeate concentration obviously affects the water flux and required pressure and vice versa, an iterative approach is necessary to solve the system. The general steps are as follows:

a) Acquire a valid water analysis and target feed pH from the user supplied data. Read this data into the internal variables.

b) Acquire flow and recovery information from the graphic screen (System Design sheet).

c) Convert all flow rates to the internal flow scheme - we use $m^3$/sec as the base flow units.

d) Acquire from the user the suggested array, bank, and vessel layout and save in the internal variables. Typically, a system with higher recovery needs more banks. For example a 75% recovery system might specify 6 elements per vessel, 2 banks total in the array, and an array layout of 16:8. That is 16 vessels feeding 8 vessels in bank 2.

e) Acquire from the user the selected membrane element information. When the user selects the type of element, the program loads the specific data about the element such as A value, B value, B values relative to Na or Cl per each ion (dependent on membrane type), area in $m^2$, differential pressure coefficients and so on. Membrane data is stored on the “Membrane Data” sheet.

f) Setup up the selected membrane element in the vessel and calculate delta pressure and concentration polarization coefficients (called the "*BetaCoef*" in the code) for the selected elements. Delta pressure is calculated from a coefficient specific to the membrane configuration. The TFS program by default utilizes coefficients based on commercial membranes typical of those elements used for so called "pure water" applications which assumes water is the medium processed and assumes normal thickness feed size spacer materials. The user can adjust these coefficients for special elements as needed.

$$dp = dpCoef * \left( \frac{FeedFlow + ConcFlow}{2} \right)^{1.5}$$ Equation 9

$$ConcPolarization = CP = Exp(BetaCoef * Recovery)$$ Equation 10

g) At this point calculations are ready to begin. The first step is to "Guess" the applied pressure needed to produce the water flow rate required. To do this the total membrane area in the system is calculated and the feed osmotic pressure is calculated. Based on nominal recovery and a nominal rejection for the element, a concentrate concentration is estimated by mass balance. The concentrate osmotic pressure is then calculated. Next a differential pressure is calculated for the first and last elements in the system and the average is used to guess a differential pressure for the system. The "Guess" pressure needed is then calculated:

$$Guess(atm) = \frac{Flow}{"A" value * Area} + Total\_dp + \frac{FeedPi + ConcPi}{2}$$ Equation 11

N.B. - Flow is $m^3$/sec, "A" value is m/(sec*atm) or $m^3$/sec/$m^2$/atm, dp and Pi in atm.

h) Guess is the net pressure, which includes osmotic pressure and dp losses so the applied pressure is increased by average of the feed and concentrate osmotic pressures and the differential pressure. Osmotic pressure is calculated using the typical equation of state, summed for each ion in the analysis. Total organic carbon contributions (if any) are ignored since the contribution is highly dependent on the type of organic carbon present.

$$OsmoticPressure(atm) = Pi = RT * \sum_{AllIons} \alpha_i [c]_i$$ Equation 12

$$R = GasConst,\ T = \deg K, \alpha_i = activity,\ [c]_i = moles/l$$

i) The Calculations for RO and NF are arranged in a "Do While Loop" that starts with an assumed applied pressure. From this pressure, flow rates and concentrations are calculated on a vessel by vessel basis through the system. The total calculated flow is then compared to the target flow the user provided. If low, then the pressure is raised proportionally and if low then the pressure is lowered proportionally. Once the calculated flow is within the set percentage of the target flow - typically 0.2% to 1% - then the calculation program terminates and data is returned to the spreadsheet for display to the user.

j) RO and NF calculations differ in approach to the transport of ions. For reverse osmosis systems, the permeate anions and cations are balanced equally. That is, any difference in overall charge in the first pass through the permeate concentration calculation (Equation 8) is split equally and the permeate concentration adjusted proportionally for each ion. For an NF system, the anions are assumed to control and the cations are adjusted to balance the permeate analysis.

k) If a convergence cannot be obtained, the user is alerted. Failure to converge usually occurs when too few or too large a number of vessels are specified for the flow rates that are required. The applied pressure required to produce the needed flow cannot be reconciled with the differential pressures necessary to pass the flow through the system.

In the source code annotation section of this report, the detail calculations can be seen. Many of the above functions utilize sub routines or functions defined to make the program code more readable and easily maintained.

## Data Collection

Data collection in the TFS program from the User takes place on the Analysis and System Design sheets. Integral to the Analysis sheet are the calculations to balance the analysis for charge neutrality and to calculate the carbonate distribution. The user has the choice of entering either a measured bicarbonate value or a measured Total Inorganic Carbon value. With the pH entered by the user, a carbonate distribution is calculated.

The second sheet – "System Design" – allows the user to enter the flow rates, recovery, and system layout. Calculation of the flows occurs as the data is entered. One important note is that recycle flow affects the concentration of the feedwater actually going to the system, sometimes quite dramatically. The TFS program allows the user to evaluate the option of designing a "tree" array approach with many banks or opting for a single bank system with recycle. Use of the recycle option causes the main calculation program to loop through the system to iteratively calculate the concentrations of the "Net Feed" stream.

### *Carbonate Equilibrium*

The calculation of the appropriate carbonate and pH distribution is an important part of this program. Note that membrane systems readily pass carbon dioxide ($CO_2$) while carbonate ($CO_3$) is quite well rejected. Thus the permeate side of the membrane is enriched with carbon dioxide while the feed/concentrate side of the membrane is enriched with carbonate and bicarbonate. The net effect of this separation is to lower the pH of the permeate while the concentrate pH rises. The TFS program effectively models this behavior. For applications in which a membrane system precedes a DI system, the additional carbon dioxide can pose a load on the anion resins. The TFS program can help the user to evaluate the use of acid versus scale inhibitor or softening resins as the pretreatment step.

The steps necessary to calculate the carbonate equilibrium are as follows:

a) The solution ionic strength is calculated, defined as

$$IS = 0.5 * \sum_{\text{All Ions}} [\text{Ion}] * \text{Abs}(z) \qquad \text{Equation 13}$$

b) pK1 and pK2 are corrected for the solution temperature.

c) Ionic strength corrections are then applied to pK1 and pK2. The methods are described in the Stumm and Morgan reference in the Literature Citations section.

d) pK1 and pK2 are converted back to K1 and K2 values.

e) The [H+] concentration is calculated from solution pH

f) The Mole Fraction (fraction of the total inorganic carbon) for $HCO_3$ is calculated from K1, K2, and pH. If the $HCO_3$ concentration was specified then the Total Carbon (TC) is calculated from the mole fraction of $HCO_3$. If Total carbon was specified then the concentration of $HCO_3$ is calculated using the mole fraction of $HCO_3$.

$$MoleFraction\ HCO_3 = \left( \frac{(H^+)}{K1} + 1 + \frac{K2}{H^+} \right)^{-1} \qquad \text{Equation 14}$$

$$TotalCarbon = TC = \frac{[HCO_3]}{MoleFraction\, HCO_3}$$

g) Knowing the total carbon, we then calculate the CO2 and CO3 values from pH and K1 and K2 as follows. The values are in moles/liter.

$$[CO_3] = TC * \left( \frac{(H^+)^2}{K1 * K2} + 1 + \frac{H^+}{K2} \right)^{-1} \qquad \text{Equation 15}$$

$$[CO_2] = TC * \left( \frac{(K1)^2}{H^+} + 1 + \frac{K1 * K2}{(H^+)^2} \right)^{-1}$$

h) For the treated feed - i.e. the feed with the pH adjusted, we have already computed the Total Carbon (TC) and thus the new carbonate distribution can be readily calculated from the equations above. The amount of acid required is easily computed from the change in bicarbonate concentration from the User supplied feed water to the Treated Feed water composition.

# User Program Instructions

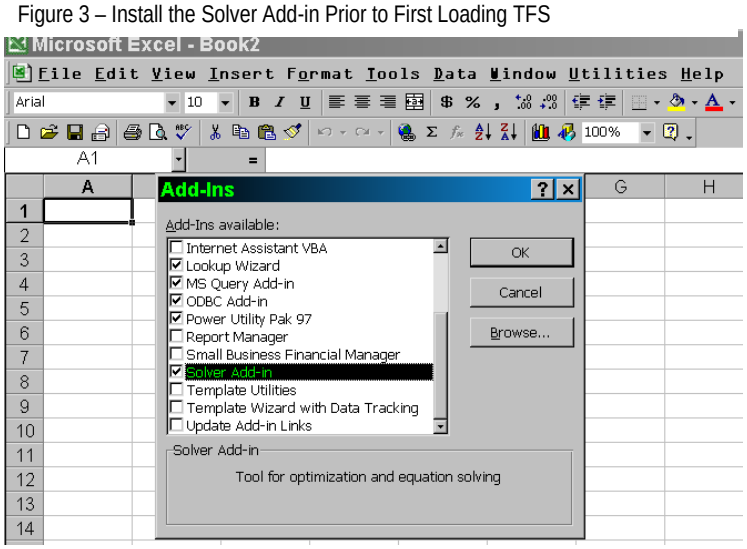
## System and Software Requirements

The TFS program requires a computer running Microsoft Office 97 or higher. Excel 97 or higher must be installed. The Solver add-in as well as Visual Basic for Applications must also be installed. Normally these two components are installed with Excel as defaults. A Pentium class computer is required for best performance and a screen resolution of at least 800 by 600 is needed.

## Basic Program Installation and Operation

1. Make a copy of your program diskette or download file.
2. Verify that Excel 97 or later in installed on your computer.
3. Launch Excel.
4. Set the calculations so that iteration is on. Tools:Options:Calculation. Be sure the iteration box is checked. If not, then check it now and click OK.
5. Check to see if the Solver Add-in is installed. Select Tools, Add-Ins. This will bring up the Add-Ins box as shown in Figure 3:
6. If the Solver box is present and checked, you are ready to run TFS. If the Solver add-in

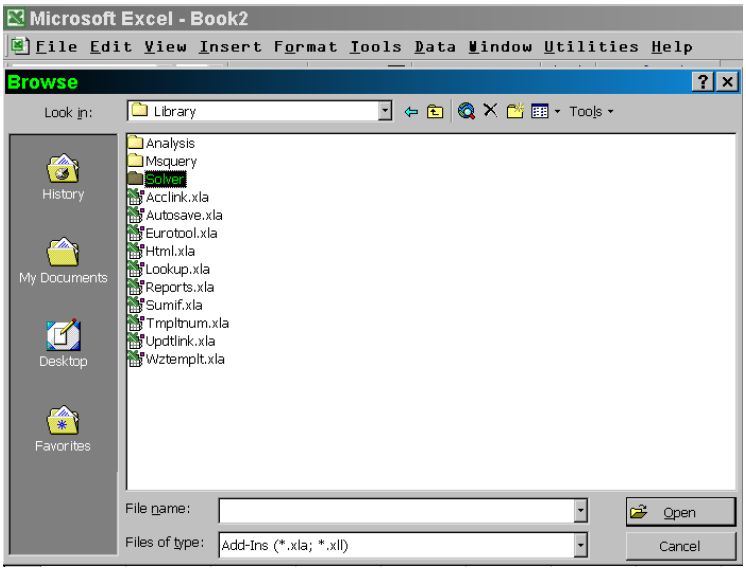
Figure 3 – Install the Solver Add-in Prior to First Loading TFS




is present but not checked, check and click OK. The system may ask you to insert your Excel or Office setup disk. In the very unlikely event the Solver add-in cannot be found, then you must install it first. Your Office CD's will have the files but for convenience, the "Solver" files have been included with the TFS release disk. They are in a directory called "Solver" which contains two files - "Solver.xla" and "Solver32.dll" On Windows 95 and 98 systems the Solver directory is usually stored in the C:\Program Files\Microsoft Office\Office\Library directory. Copy the entire Solver directory from either your Office setup disk or the TFS release to the \Library directory. Then use the Browse function to

find the program Solver.xla and install it - …\Library \Solver\ Solver.xla. See the next figure – figure 4.

Figure 4 – Manually finding (browsing) the Solver Add-in if necessary




7. Copy the program file – "TFS Ver x.x.xx.xls" from the release disk to a new hard drive directory of your choice.
8. Use the Excel File:Open command to find and open the file.
9. The program will load, go through some startup initialization and start at the Analysis Tab. Wait for the loading to finish then click the InfoLicense tab to read the conditions of use.
10. Return to the Analysis sheet. Enter analytical data in the cells with blue text and a white background. **206.0**

    The cells which contain formulas or links or static text often contain a pop up help balloon explaining their function. Most cells on this sheet which are not meant to accept user data are protected and a message will display if you attempt to enter data, or enter non-numeric data in a field expecting numeric data.
11. Complete the analytical data, specify a target pH or resin softening, and then move to the System Design sheet by selecting the System Design tab at the bottom of the screen.
12. In the System Design sheet you may enter flow and recovery information as well as the membrane type plus system design or layout. The smaller field near the bottom of this sheet – rows 37 through 42 – show an estimated pressure and permeate TDS conditions as flow rates are changed. You will need to adjust the layout and number of membranes (banks, number of vessels, and number of elements per vessel) so that the pressure is in a reasonable range – say 100 to 1200 psig.
13. Once complete, press the Calculate button and the calculations will be launched. On completion, the smaller field shows the calculated pressure and TDS. Buttons near this field labeled "See Summary Data" and "See Stream Data" take you to the two summary

data sheets which display all of the calculated data available. Both sheets can be printed.

14. If it is desired to save the system data, press on the tab labeled “DataBase” and find (and press) the button labeled “Save Current Data Set Now.” This saves the data into the first row of the database. Stored system may be sorted in numerous ways to find old data sets. To restore an old dataset, position the cursor anywhere in the row holding the data which is to be restored and then press the “Load Selected Data Set Now” button. This restores the only the feed flow and analytical data so you can test your current system design on a new water by simply recalculating.

### *List of Needed Add-In files and typical locations*

You will need the following files.

- The main program – TFSP ver xx.xls, located in your specified directory
- Solver.xla - Typically in C:\Program Files\Microsoft Office\Office\Library\Solver
- Solver32.dll - Typically in C:\Program Files\Microsoft Office\Office\Library\Solver
- TFS Program Documentation.doc (This file), located in your specified directory.
- WateQF.txt - the WaterQF database file in text format, located in your directory.
- Phreeqc.txt - the Phreeqc database file in text format, located in your directory.

## Special Functions

In addition to the basic operations above, the knowledgeable user can make several additions to the program functionality.

### *Add /Edit Membrane Element Types*

Before using this function, save the spreadsheet to prevent inadvertent loss to existing data. Go to the System Design sheet and select the button on the sheet entitled Add/Edit Element. This calls up a form which allows the user to create a new membrane from typical spec sheet data provided by the manufacturer. Enter data in ALL of the boxes on the form.

### *Modify/Create New Membrane Type*

This feature allows the user to change the characteristics of the membrane salt passage by ion. To modify the membrane data, select the Modify Membrane button or move to cell AB16. The user may then modify the temperature correction factor or any of the constants which relate individual ion passage to Cl or Na.

To create a new membrane, it is easiest to use the Add New Membrane button on the Membrane Data sheet. Press the button. Select the first/next empty row to create the new membrane data. A maximum of 10 membrane types are allowed. The button will take you to the first empty column.

First – enter the data. The “Type Name” (row 16) must be unique. Any of the other data may be copied from the prior membrane type. If the new type is NF be sure to set the value in the Calculate

as Nanofiltration row to TRUE. When done entering data in the column, press the NEXT button at the bottom of the screen. It will ask you for the name of the membrane type you are entering. This appears in the membrane type list on the System Design sheet so make it descriptive.

The program will show the current number of membrane types (the program came with 3) so you should accept the program value unless something is wrong.

The program will return to the System Design sheet and the new membrane should appear in the left hand drop down box. However, there are no elements yet with this type of membrane so now you must create at least one element use the Add/Edit Element button as described above. Once completed, you are ready to use the new element with the newly defined membrane type.

### *Printing*

The TFS program was designed with the bulk of the returned data displayed on two sheets, each suitable for printing on one page. The release version of the program is setup so that the Summary and Streams sheets print on one page each. Of course the user may change this arrangement at any time through the normal Excel printing mechanisms.

Source code may be printed as well, however, some intermediate export function is required. To print or view the code in a module, first launch the Visual Basic editor – Tools|Macros|Visual Basic Editor or use the Alt+F11 key. In the left hand pane, highlight the module you wish to export or print and right click or use the File function in the VB menu. Select Export or Print as desired.

# References


ASTM. (1993) Section VI - Water-Treatment Materials: Section D 3739 - 88 "Standard practice for calculation and Adjustment of the Langelier Saturation Index for Reverse Osmosis".

ASTM. (1993) Section VI - Water-Treatment Materials: Section D 4692 - 87 "Standard practice for calculation of sulfate scaling salts (CaSO4, SrSO4, BaSO4) for reverse osmosis". In *Annual Book of ASTM Standards*, Vol. 11.02, pp. 868-871.

ASTM. (1993) Section VI - Water-Treatment Materials: Section D 4993 - 89 "Standard practice for calculation and Adjustment of Silica ($SiO_2$) for Reverse Osmosis".

Bennett, A.C., and Adams, F., 1976, Solubility and solubility product of dicalcium phosphate dihydrite in aqueous solutions and soil solutions: *Soil. Sci. Soc. Amer. Jour*., 40, 39-42.

Bethke C. M. (1992) The Geochemist's Workbench*: A users guide to Rxn, Act2, Tact, React, and Gtplot*. U. of Illinois Press.

Byrne W., 1995, *Reverse Osmosis A practical guide for industrial users*., Tall Oaks Publishing

Cox, J.D., Wagman, D.D., and Medvedev, V.A., 1989, *Codata key values for thermodynamics*: Hemisphere Pub., New York, 271p.

Delany J. M., (1985) *Reaction of Topopah Spring Tuff with J-13 water: A geochemical modeling approach.*, Lawrence Livermore National Laboratory, Report Number UCRL-53631;

Eriksson P., (1988) Water and Salt Transport Through Two Types of Polyamide Composite Membranes, Journal of Membrane Science, 36, p 297-313

Hem J. D., (1992) *Study and interpretation of the chemical characteristics of natural waters*, United States Geological Survey, Report Number Water Supply Paper 2254; 263 p. p.;

Hogfeldt E. (1982) *Stability Constants of Metal-Ion Complexes. Part A Inorganic Ligands*. Pergamon Press. 310 pages p.

Holland H. D. (1978) *The Chemistry of the Atmosphere and Oceans*. John Wiley & Sons. 351 p. p.

Johnson J. W. and Lundeen S. R., (1994) *GEMBOCHS thermodynamic datafiles for use with the EQ3/6 software package*, Lawrence Livermore National Laboratory, Report Number LLNL--YMP milestone report MOL72, 99; 126 p.;

Langmuir, D., (1979), *Techniques of estimating thermodynamic properties for some aqueous complexes of geochemical interest*, in Jenne, E.A., ed., Chemical Modeling in Aqueous systems: Speciation, Sorption, Solubility, and Kinetics: ACS Symp. Ser. 93, Amer. Chem. Soc., Washington, 353-387.

Merten U (editor), (1966), *Transport Properties of Osmotic Membranes,* in Desalination by Reverse Osmosis, MIT Press.

Montgomery JM, (1985), Water Treatment Principles and Design, Wiley and Sons.

Naumov G. B., Ryzhenko B. N., and Khodakovsky I. L., (1974) *Handbook of Thermodynamic Data, United States Geological Survey*, Report Number USGS-WRD-74-001; 328 p. p.;

Nriagu, J.O., 1972, Stability of vivianite and ion-pair formation in the system Fe3(PO4)2-H3PO4-H2O: *Geochim. Cosmochim. Acta*, 36, 459-470.

Parkhurst D. L., Thorstenson D. C., and Plummer L. N., (1980) *PHREEQE- A computer program for geochemical calculations*, United States Geological Survey, Report Number

Plummer L. N., Jones B. F., and Truesdell A. H., (1976) *WATEQF - A Fortran IV version of WATEQ, a computer program for calculating chemical equilibrium of natural waters.,* United States Geological Survey, Report Number Wat. REs. Inv. 76-13.;

Robie, R.A., Hemingway, B.S., and Fisher, J.R., 1979, *Thermodynamic properties of minerals and related substances at 298.15K and 1 bar ($10^5$ Pascals) pressure and at higher temperatures*: U.S.G.S. Bull. 1452 (with corrections), 1452, 1456p.

Shock, E.L., and Helgeson, H.C., 1988, Calculation of the thermodynamic and transport properties of aqueous species at high pressures and temperatures: Correlation algorithms for ionic species and equation of state predictions to 5kb and 1000°C: Geochim. Cosmochim. Acta, 52, 2009-2036.

Shock, E.L., and Helgeson, H.C., 1989, Corrections to Shock and Helgeson (1988): *Geochim. Cosmochim. Acta*, 53, 215.

Shock, EL; Sassani, DC; Willis, M; Sverjensky, DA., 1997, Inorganic species in geologic fluids: Correlations among standard molal thermodynamic properties of aqueous ions and hydroxide complexes. *Geochim. Cosmochim. Acta,* 61:907-950.

Smith R. M. and Martell A. E. (1976) *Critical Stability Constants. Volume 4: Inorganic complexes*. Plenum Press. 257 p.

Spycher N. F. and Reed M. H., (1989) *SOLVEQ: A computer program for computing aqueous-mineral-gas equilibria.*, University of Oregon, Report Number unpublished;

Stumm W., Morgan J, 1976, Aquatic Chemistry, Wiley and Sons

Sverjensky D. A., Shock E. L., and Helgeson H. C. (1997) Prediction of the thermodynamic properties of aqueous metal complexes to 1000 C and 5 kb. *Geochim. Cosmochim. Acta* 61:1359-1412.

Truesdell, A.H., and Jones, B.F., 1974, WATEQ, a computer program for calculating chemical equilbria of natural waters: *U.S.G.S. J. Res*., 2, 233-248.

Wolery T. J., (1992) EQ3NR, *A computer program for geochemical aqueous speciation-solubility calculations: Theoretical manual, User's Guide, and Related Documentation*, Lawrence Livermore National Laboratory.

Woods, T.L., and Garrels, R.M., 1987, Thermodynamic values at low temperature for natural inorganic materials: An uncritical summary: Oxford Univ. Press, Oxford.

# Source Code and Annotations

## General Comments and Arrangement

The Visual Basic source code was written using the standard software provided in Microsoft Office 97. The source code is not protected in any way and can be viewed by the following procedure. Load the program, select Tools from the standard Excel Menu Bar, select Macros, select Visual Basic Editor. In the left hand pane, locate the program folder. Under this folder locate the folder icon labeled “Modules” and double click it. Select one of the modules – for example RO/NF calc. This opens the Visual Basic editor in the right hand side of the screen. The code should be visible.

The code has been broken down logically into several different code “modules” according to function. A Visual Basic or any type of Basic programmer should readily be able to understand and modify the code as they may desire. No additional compilers or software are needed to make changes to the code, so the user should keep a copy of the original installed program should the need arise for a complete code backup.

Most of the functions used are accessible in the Microsoft provided Visual Basic and Excel Help Files. If the Visual Basic help file cannot be found, it may be necessary to install it using the original MS Office installation disk or network installation pathname. The only poorly documented (from Microsoft) functions used in the TFS program are the calls to launch the Excel Solver directly from Visual Basic. These can be found by “recording” a macro which includes a manual call to the Solver.

The actual source code itself is displayed below. Comments are preceded with the single exclamation point (‘) and do not execute. Comments in *italics* are added in the final report notation. Note that some lines have wrapped themselves in the report format and do not indicate a new line of code.

Where reasonably convenient, references to cells in the worksheets is via the Range Name rather than a direct cell reference. All cell references are via the “A1” type format. The Application.Screenupdating function turns on or off the screen update. This prevents flicker while many cells are being changed.

## UtilsAnalysis Module

This module contains code to switch units

```
Attribute VB_Name = "UtilsAnalysis"
Option Explicit
Public strFlowFormat As String
```

*This sub responds to a change in the units for analysis – ie meq/l to ppm to ppm as CaCO3 etc.*

```
Sub AnalyisGroup_Click()
Attribute AnalyisGroup_Click.VB_Description = "Macro recorded 5/26/98 by Tom Wolfe"
Attribute AnalyisGroup_Click.VB_ProcData.VB_Invoke_Func = " \n14"
'
'   Modified to reflect changes made
'   Since we have now separated the User Input from the values used for calculation to avoid
problems with bad input.
'   Changed the cell references for following
'       Input_Anion, Input_Cation, Input_Silica, Input_Balance, Input_CO2, Input_CO3
'       Added update to Feed_CO3 and Feed_CO2, which were the Input_CO2, Input_CO3
'       Added new range - User_Sum_Metals and formula for this cell
```

```
'       NB be sure to change the cell references in VB if rows are added on the PL_IonBal Sheet

' AnalyisGroup_Click Macro

Dim myRange As Range
If BoolLoading = False Then
    Application.ScreenUpdating = False
End If
'   also need to turn off iteration
Application.Iteration = False
Application.Calculation = xlManual
```

*This section looks to see if we are using Total Inorganic Carbon or HCO3*

```
Dim boolTIC As Boolean      ' added 11/1/99 for Total Inorganic carbon
If Left(Range("TotalInorganicCarbonFlag").Value, 1) = "Y" Or
Left(Range("TotalInorganicCarbonFlag").Value, 1) = "y" Then
    boolTIC = True
Else
    boolTIC = False
End If
```

*The actual calculations for units are done are the spreadsheet – this routine moves the proper values to the input section and resets the pointers. All internal calculations are handled in meq/l units.*

```
Select Case Range("Units_Analytical")
    Case "ppm"
        Select Case Range("Units_OldAnalytical")
            Case "ppm"
                GoTo windup 'Exit Sub
            Case "ppm CaCO3"    '    was ppm now wants to be ppm as CaCO3

                Range("Input_cation").Value = Range("out_cations_caco3").Value
                Range("Input_anion").Value = Range("out_anions_caco3").Value
                Range("Input_balance").Value = Range("out_balance_caco3").Value
                Range("Input_silica").Value = Range("out_silica_caco3").Value
                '   now fix up the CO2 and CO3 so they re-iterate properly
                Range("Input_CO2").Formula = "=Calc_CO2_CaCO3"
                Range("Input_CO3").Formula = "=Calc_CO3_CaCO3"
                '   8 July 1999 change
                '   The Feed_CO2 range already existed
                Range("Feed_CO2").Formula = "=Calc_CO2_CaCO3"
                Range("Feed_CO3").Formula = "=Calc_CO3_CaCO3"

                '   11/1/99 add version AV total inorganic carbon
                '   just need to fix up the HCO3
                If boolTIC = True Then
                    Range("User_HCO3").Value = Range("HCO3_FromTC_CaCO3").Value
                End If
            Case "mEq/L"        '   was ppm and now wants to be meq/l
                Range("Input_cation").Value = Range("out_cations_meq").Value
                Range("Input_anion").Value = Range("out_anions_meq").Value
                Range("Input_balance").Value = Range("out_balance_meq").Value
                Range("Input_silica").Value = Range("out_silica_meq").Value
                '   now fix up the CO2 and CO3 so they re-iterate properly
                Range("Input_CO2").Formula = "=Calc_CO2_meq"
                Range("Input_CO3").Formula = "=Calc_CO3_meq"
                '   8 July 1999 change
                '   The Feed_CO2 range already existed
                Range("Feed_CO2").Formula = "=Calc_CO2_meq"
                Range("Feed_CO3").Formula = "=Calc_CO3_meq"
                '   11/1/99 add version AV total inorganic carbon
                '   just need to fix up the HCO3
                If boolTIC = True Then
                    Range("User_HCO3").Value = Range("HCO3_FromTC_meq").Value
                End If

        End Select
    Case "ppm CaCO3"
        Select Case Range("Units_OldAnalytical")
            Case "ppm"
```

```
            Range("Input_cation").Value = Range("out_cations_ppm").Value
            Range("Input_anion").Value = Range("out_anions_ppm").Value
            Range("Input_balance").Value = Range("out_balance_ppm").Value
            Range("Input_silica").Value = Range("out_silica_ppm").Value
            '   now fix up the CO2 and CO3 so they re-iterate properly
            Range("Input_CO2").Formula = "=Calc_CO2"
            Range("Input_CO3").Formula = "=Calc_CO3"
            '   8 July 1999 change
            '   The Feed_CO2 range already existed
            Range("Feed_CO2").Formula = "=Calc_CO2"
            Range("Feed_CO3").Formula = "=Calc_CO3"
            '   11/1/99 add version AV total inorganic carbon
            '   just need to fix up the HCO3
            If boolTIC = True Then
                Range("User_HCO3").Value = Range("HCO3_FromTC").Value
            End If


        Case "ppm CaCO3"    '    was ppm now wants to be ppm as CaCO3
            GoTo windup

        Case "mEq/L"        '   was ppm and now wants to be meq/l
            Range("Input_cation").Value = Range("out_cations_meq").Value
            Range("Input_anion").Value = Range("out_anions_meq").Value
            Range("Input_balance").Value = Range("out_balance_meq").Value
            Range("Input_silica").Value = Range("out_silica_meq").Value
             '   now fix up the CO2 and CO3 so they re-iterate properly
            Range("Input_CO2").Formula = "=Calc_CO2_meq"
            Range("Input_CO3").Formula = "=Calc_CO3_meq"
            '   8 July 1999 change
            '   The Feed_CO2 range already existed
            Range("Feed_CO2").Formula = "=Calc_CO2_meq"
            Range("Feed_CO3").Formula = "=Calc_CO3_meq"
            '   11/1/99 add version AV total inorganic carbon
            '   just need to fix up the HCO3
            If boolTIC = True Then
                Range("User_HCO3").Value = Range("HCO3_FromTC_meq").Value
            End If

    End Select

Case "mEq/L"
    Select Case Range("Units_OldAnalytical")
        Case "ppm"
            Range("Input_cation").Value = Range("out_cations_ppm").Value
            Range("Input_anion").Value = Range("out_anions_ppm").Value
            Range("Input_balance").Value = Range("out_balance_ppm").Value
            Range("Input_silica").Value = Range("out_silica_ppm").Value
            '   now fix up the CO2 and CO3 so they re-iterate properly
            Range("Input_CO2").Formula = "=Calc_CO2"
            Range("Input_CO3").Formula = "=Calc_CO3"
            '   8 July 1999 change
            '   The Feed_CO2 range already existed
            Range("Feed_CO2").Formula = "=Calc_CO2"
            Range("Feed_CO3").Formula = "=Calc_CO3"
            '   11/1/99 add version AV total inorganic carbon
            '   just need to fix up the HCO3
            If boolTIC = True Then
                Range("User_HCO3").Value = Range("HCO3_FromTC").Value
            End If

        Case "ppm CaCO3"    '    was meq now wants to be ppm as CaCO3
            Range("Input_cation").Value = Range("out_cations_caco3").Value
            Range("Input_anion").Value = Range("out_anions_caco3").Value
            Range("Input_balance").Value = Range("out_balance_caco3").Value
            Range("Input_silica").Value = Range("out_silica_caco3").Value
             '   now fix up the CO2 and CO3 so they re-iterate properly
            Range("Input_CO2").Formula = "=Calc_CO2_CaCO3"
            Range("Input_CO3").Formula = "=Calc_CO3_CaCO3"
            '   8 July 1999 change
            '   The Feed_CO2 range already existed
```

```
                Range("Feed_CO2").Formula = "=Calc_CO2_CaCO3"
                Range("Feed_CO3").Formula = "=Calc_CO3_CaCO3"
                '   11/1/99 add version AV total inorganic carbon
                '   just need to fix up the HCO3
                If boolTIC = True Then
                    Range("User_HCO3").Value = Range("HCO3_FromTC_CaCO3").Value
                End If

            Case "mEq/L"        '   was meq and now wants to be meq/l
                GoTo windup  'Exit Sub

        End Select

End Select
'   if we go directly to windup no need for this section to put back formula in heavy metals
summation
'   8 July 1999 changed row reference to fit move of Al and Fe to upper section
'   8 July 1999 add write back to user_Sum_Metals formula
Range("User_Sum_Metals").Formula = "=Sum_Metals"
windup:
Range("Units_Analytical").Value = Range("Units_OldAnalytical").Value
Application.Calculation = xlAutomatic
Application.Iteration = True

If BoolLoading = False Then
    Application.ScreenUpdating = True
End If
End Sub

Sub Flow_Select_Drop_Click()
Attribute Flow_Select_Drop_Click.VB_ProcData.VB_Invoke_Func = " \n14"

Dim oldFlowUnits As Integer
Dim newFlowUnits As Integer

Dim myRange As Range
Dim FlowGpm As Double
Dim NewFlow As Double
oldFlowUnits = Range("select_oldflow").Value
newFlowUnits = Range("select_flow").Value

'   first get the old flow rate in gpm then convert it to the new flow rate
'   The old gpm rate is in Flow_gpm
    FlowGpm = Range("FeedFlow_gpm").Value

    Select Case Range("FlowUnit")
        Case 1
            NewFlow = FlowGpm
        Case 2  'to GPD
            NewFlow = FlowGpm * 1440
        Case 3
            NewFlow = FlowGpm * 60 / 264.2
        Case 4
            NewFlow = FlowGpm * 1440 / 264.2
        Case 5
            NewFlow = FlowGpm * 3.784
        Case 6
            NewFlow = FlowGpm * 60 * 3.784
    End Select
    '   Now update the system design worksheet and reset the flows there
    tempStop = True
    Range("FeedFlow").Value = NewFlow
    Worksheets("System Design").UpdateFlows 7

    '   now format the result
    '   this gives an error durng dataLoad - go around it
    If BoolLoading = False Then
        'Range("input_flow").NumberFormat = SetFormat(NewFlow)
    Else '  save the format
        strFlowFormat = SetFormat(NewFlow)
    End If
```

```
    '   now put the current flow selection value in the select_OldFlow
    Range("old_FlowUnit").Value = Range("FlowUnit").Value
End Sub
```

*The temperature units – C, F, K. Internal units are deg C.*

```
Sub Temp_Select_click()
Attribute Temp_Select_click.VB_ProcData.VB_Invoke_Func = " \n14"
Dim myRange As Range
Dim Temp_F As Double
Dim NewTemp As Double
    Temp_F = Range("Temp_F").Value
    Select Case Range("select_temperature")
        Case 1
            NewTemp = Temp_F
        Case 2  'to C
            NewTemp = (Temp_F - 32) * 5 / 9
        Case 3  ' to deg K
            NewTemp = (Temp_F - 32) * 5 / 9 + 273.15
    End Select
    Range("input_Temp").Value = NewTemp
    '   now put the selected temp back
    Range("select_oldTemperature").Value = Range("select_temperature").Value
End Sub

Sub Acid_Select_Click()
Range("input_acid").Value = Range("select_acid_name").Value
If Range("input_acid").Value = "No Acid" Or Range("input_acid").Value = "Sodium Softener" Then
        Worksheets("Analysis").Range("Input_Feed_pH").Font.ColorIndex = 16
        Worksheets("Analysis").Range("Input_Feed_pH").Interior.ColorIndex = 15
        Worksheets("Analysis").Range("Input_Feed_pH").Interior.Pattern = xlSolid
    Else
        Worksheets("Analysis").Range("Input_Feed_pH").Font.ColorIndex = 5
        Worksheets("Analysis").Range("Input_Feed_pH").Interior.ColorIndex = xlNone
End If
End Sub
```

*Sets target pH – ignored if no acid is selected.*

```
Sub Spin_pH_Click()
Dim myRange As Range
Dim OldpH As Double
Dim NewpH As Double
Dim FeedpH As Double

'   Sets ph for system - used to calculate treated Feed conditions and acid dose
'   Spin_pH /10
OldpH = Range("Old_pH").Value
NewpH = Range("Spin_pH").Value / 10
    If Abs(OldpH - NewpH) > 0.2 Then
        '   we had a value entered by the user, reset the spinner to the prior user entered value
        Range("Spin_ph").Value = OldpH * 10
    End If
Range("input_feed_ph").Formula = "=Spin_ph / 10"
End Sub
```

## Database Module

*This code saves and retrieves data from a stored data set- Sheet is "DataBase"*

```
Option Explicit
'
'   This is the Save amd Retrieve Module
'   Puts or takes data from the workbooks.Sheets("Database") sheet
'   Data is stored as a row of data.
Public FeedIons(52) As Double
Public BoolLoading As Boolean
Public BoolOptimizing As Boolean
Public Const MainSheet = "Analysis"
```

```
Type ProjectData
    ProjectName As String
    Projectdate As Date
    FeedGPM As Double
    TempF As Double
    Recovery As Double
    Rejection As Double
    AcidType As Integer
    InputPH As Double
    FeedTDS As Double
    FeedUnits As String
    nUnits As Integer
    FlowUnits As String
    nFlowUnits As Integer
End Type
Dim MyProjectData As ProjectData

Public Const ION_NA = 0
Public Const ION_K = 1
Public Const ION_CA = 2
Public Const ION_MG = 3
Public Const ION_SR = 4
Public Const ION_BA = 5
Public Const ION_FE3 = 6
Public Const ION_AL = 7
Public Const ION_NH4 = 8
Public Const LAST_CATION = 8

Public Const Ion_H = 9

Public Const FIRST_ANION = 10
Public Const ION_HCO3 = 10
Public Const ION_CO2 = 11
Public Const ION_CO3 = 12
Public Const ION_BR = 13
Public Const ION_CL = 14
Public Const ION_F = 15
Public Const ION_SO4 = 16
Public Const ION_NO3 = 17
Public Const ION_PO4 = 18

Public Const LAST_ANION = 18

Public Const Ion_OH = 19
Public Const ION_SIO2 = 20
Public Const Ion_pH = 21
Public Const ION_ORGANICS = 22
Public Const Ion_AG = 23
Public Const Ion_AS = 24
Public Const Ion_AU = 25
Public Const Ion_CD = 26
Public Const Ion_CR = 27
Public Const Ion_CU = 28
Public Const Ion_FE2 = 29
Public Const Ion_HG = 30
Public Const Ion_MN = 31
Public Const Ion_NI = 32
Public Const Ion_PB = 33
Public Const Ion_SE = 34
Public Const Ion_SN = 35
Public Const Ion_TI = 36
Public Const Ion_V = 37
Public Const Ion_ZN = 38
Public Const ION_B4O7 = 39
Public Const ION_NO2 = 40
Public Const ION_SO3 = 41

Public Sub AssignIons()
    FeedIons(ION_NA) = Range("Feed_Na").Value
    FeedIons(ION_K) = Range("Feed_K").Value
    FeedIons(ION_CA) = Range("Feed_CA").Value
```

```
    FeedIons(ION_MG) = Range("Feed_MG").Value
    FeedIons(ION_SR) = Range("Feed_SR").Value
    FeedIons(ION_BA) = Range("Feed_BA").Value
    FeedIons(ION_NH4) = Range("Feed_NH4").Value
    FeedIons(Ion_H) = Range("Feed_H").Value
    FeedIons(ION_HCO3) = Range("Feed_HCO3").Value
    FeedIons(ION_CO2) = Range("Feed_CO2").Value
    FeedIons(ION_CO3) = Range("Feed_CO3").Value
    FeedIons(ION_BR) = Range("Feed_BR").Value
    FeedIons(ION_CL) = Range("Feed_CL").Value
    FeedIons(ION_F) = Range("Feed_F").Value
    FeedIons(ION_SO4) = Range("Feed_SO4").Value
    FeedIons(ION_NO3) = Range("Feed_NO3").Value
    FeedIons(ION_PO4) = Range("Feed_PO4").Value
    FeedIons(Ion_OH) = Range("Feed_OH").Value
    FeedIons(ION_SIO2) = Range("Feed_SIO2").Value
    FeedIons(Ion_pH) = Range("Feed_pH").Value
    FeedIons(ION_ORGANICS) = Range("Feed_ORGANICS").Value
    FeedIons(Ion_AG) = Range("Feed_AG").Value
    FeedIons(ION_AL) = Range("Feed_AL").Value
    FeedIons(Ion_AS) = Range("Feed_AS").Value
    FeedIons(Ion_AU) = Range("Feed_AU").Value
    FeedIons(Ion_CD) = Range("Feed_CD").Value
    FeedIons(Ion_CR) = Range("Feed_CR").Value
    FeedIons(Ion_CU) = Range("Feed_CU").Value
    FeedIons(Ion_FE2) = Range("Feed_FE2").Value
    FeedIons(ION_FE3) = Range("Feed_FE3").Value
    FeedIons(Ion_HG) = Range("Feed_HG").Value
    FeedIons(Ion_MN) = Range("Feed_MN").Value
    FeedIons(Ion_NI) = Range("Feed_NI").Value
    FeedIons(Ion_PB) = Range("Feed_PB").Value
    FeedIons(Ion_SE) = Range("Feed_SE").Value
    FeedIons(Ion_SN) = Range("Feed_SN").Value
    FeedIons(Ion_TI) = Range("Feed_TI").Value
    FeedIons(Ion_V) = Range("Feed_V").Value
    FeedIons(Ion_ZN) = Range("Feed_ZN").Value
    FeedIons(ION_B4O7) = Range("Feed_B4O7").Value
    FeedIons(ION_NO2) = Range("Feed_NO2").Value
    FeedIons(ION_SO3) = Range("Feed_SO3").Value
End Sub

Public Sub AssignProjectData()
With MyProjectData
    .ProjectName = Range("Input_projectname").Value
    .Projectdate = Now()
    .FeedGPM = Range("FeedFlow_gpm").Value
    .TempF = Range("Temp_F").Value
    .Recovery = Range("SystemRecovery").Value
    .Rejection = Range("Input_Rejection").Value
    .AcidType = Range("Select_acid").Value
    .InputPH = Range("Input_Feed_pH").Value
    .FeedTDS = Range("Feed_TDS").Value
    .FeedUnits = Range("Units_analytical").Value
    .nUnits = Range("Select_Analytical").Value
    .FlowUnits = Range("Units_Flow").Value
    .nFlowUnits = Range("FlowUnit").Value
End With
End Sub

Sub DataSave()
Dim i As Integer
Dim nRow As Integer, nCol As Integer
Dim mySheet As Worksheet
'   be sure we come back to where we started from
Set mySheet = ActiveWorkbook.ActiveSheet
'   On Error Resume Next
'   first make some room for the new data - add a new row
    Application.ScreenUpdating = False
    AssignIons
    AssignProjectData
    Worksheets("DataBase").Activate
```

```
    Range("Data_Start").Select
    ActiveCell.Offset(1, 0).Select
    Selection.EntireRow.Insert

'   we are now at the date column, so we can just start putting in the data
    With MyProjectData
        Selection.Value = .Projectdate
        ActiveCell.Offset(0, 1).Select
        Selection.Value = .ProjectName
        ActiveCell.Offset(0, 1).Select
        Selection.Value = .FeedGPM
        ActiveCell.Offset(0, 1).Select
        Selection.Value = .TempF
        ActiveCell.Offset(0, 1).Select
        Selection.Value = .Recovery
        ActiveCell.Offset(0, 1).Select
        Selection.Value = .Rejection
        ActiveCell.Offset(0, 1).Select
        Selection.Value = .AcidType
        ActiveCell.Offset(0, 1).Select
        Selection.Value = .InputPH
        ActiveCell.Offset(0, 1).Select
        Selection.Value = .FeedTDS
        ActiveCell.Offset(0, 1).Select
        '   meq/ppm/caco3
        Selection.Value = .FeedUnits
        ActiveCell.Offset(0, 1).Select
        Selection.Value = .nUnits
        '   gpm/gpd/etc.
        ActiveCell.Offset(0, 1).Select
        Selection.Value = .FlowUnits
        ActiveCell.Offset(0, 1).Select
        Selection.Value = .nFlowUnits
    End With

    '   now the ions
    For i = ION_NA To ION_SO3
        ActiveCell.Offset(0, 1).Select
        Selection.Value = FeedIons(i)
    Next i

    '   now include chlorine
        ActiveCell.Offset(0, 1).Select
        Selection.Value = Range("User_Chlorine").Value

    '   last the valid data checksum - this must be greater than 0
        ActiveCell.Offset(0, 1).Select
        'nRow = ActiveCell.Row
        nCol = ActiveCell.Column - 2    '   subtract 1 for current column, and 1 to miss the
first row date
        Selection.Formula = "=SUM(RC[-" & Format(nCol, "#") & "]:RC[-1])"

mySheet.Activate
Application.ScreenUpdating = True
MsgBox ("Data Saved to Sheet - Database")

End Sub

Sub ResetProject()
Dim strBlanks As String
strBlanks = " "

    Range("Input_projectname").Value = strBlanks
    Range("Input_projectDate").Formula = Now()

    '   flow to gpm
    Range("Select_flow").Value = 1
    Range("Input_Flow").Value = 100#
    Flow_Select_Drop_Click
```

```
    Range("Select_temperature").Value = 1
    Range("Input_Temp").Value = 77
    Temp_Select_click

    Range("Select_analytical").Value = 1
    AnalyisGroup_Click

    Range("Select_acid").Value = 1
    Acid_Select_Click


End Sub

Sub DataLoad()
Dim nRow As Integer
Dim nCol As Integer
Dim i As Integer
Dim float_Checksum As Double
Dim nErrCount As Integer
nErrCount = 0
Application.ScreenUpdating = False
BoolLoading = True
nRow = ActiveCell.Row
Range(Cells(nRow, 1), Cells(nRow, 1)).Select
nCol = Range("checksum").Column
If Not IsNumeric(Cells(nRow, nCol).Value) Then
    '   check to see if he just sorted and stuck at top of column. If so move down one row to
first data set
    If nRow = Range("checksum").Row Then
        nRow = nRow + 1
        ActiveCell.Offset(1, 0).Select
    Else
        MsgBox ("The data row selected does not appear to contain valid data. Please check the
selected row and try again.")
        Exit Sub
    End If
End If
float_Checksum = Cells(nRow, nCol).Value
If float_Checksum <= 0 Then
    If MsgBox("The data row selected does not appear to contain valid data. Continuing may cause
invalid results.  Continue Anyway?", vbOKCancel) _
                        = vbCancel Then Exit Sub
    '   Doesn't want to cancel
End If

'   now at column one in the data row we want to load
'   Put all of the data into the variables we need
With MyProjectData

        If Not IsNumeric(Selection.Value) Then
                nErrCount = nErrCount + 1
                Selection.Value = Now()
                'ActiveCell.Font.Color = vbRed
        End If
        .Projectdate = Selection.Value

        ActiveCell.Offset(0, 1).Select
            .ProjectName = Selection.Value

        ActiveCell.Offset(0, 1).Select
            If Not IsNumeric(Selection.Value) Then
                Selection.Value = 100
                nErrCount = nErrCount + 1
                ActiveCell.Font.Color = vbRed
            ElseIf Selection.Value < 0 Then
                Selection.Value = 100
                nErrCount = nErrCount + 1
                ActiveCell.Font.Color = vbRed
            End If
            .FeedGPM = Selection.Value
```

```
        ActiveCell.Offset(0, 1).Select
            If Not IsNumeric(Selection.Value) Then
                Selection.Value = 77
                nErrCount = nErrCount + 1
                ActiveCell.Font.Color = vbRed
            ElseIf Selection.Value < 32 Then
                Selection.Value = 77
                nErrCount = nErrCount + 1
                ActiveCell.Font.Color = vbRed
            End If
            .TempF = Selection.Value

        ActiveCell.Offset(0, 1).Select
            If Not IsNumeric(Selection.Value) Then
                If MsgBox("The recovery data appears invalid. Set to default of 75%?",
vbOKCancel) = vbCancel Then Exit Sub
                Selection.Value = 75
                nErrCount = nErrCount + 1
                ActiveCell.Font.Color = vbRed
            ElseIf Selection.Value > 95 Or Selection.Value < 10 Then
                Selection.Value = 75#
                nErrCount = nErrCount + 1
                ActiveCell.Font.Color = vbRed
            End If
            .Recovery = Selection.Value

        ActiveCell.Offset(0, 1).Select
            If Not IsNumeric(Selection.Value) Then
                Selection.Value = 98
                nErrCount = nErrCount + 1
                ActiveCell.Font.Color = vbRed
            ElseIf Selection.Value < 10 Then
                Selection.Value = 98
                nErrCount = nErrCount + 1
                ActiveCell.Font.Color = vbRed
            ElseIf Selection.Value > 100 Then
                Selection.Value = 98
                nErrCount = nErrCount + 1
                ActiveCell.Font.Color = vbRed
            End If
            .Rejection = Selection.Value

        '   acid type number
        ActiveCell.Offset(0, 1).Select
            If Not IsNumeric(Selection.Value) Then
                Selection.Value = 1
                nErrCount = nErrCount + 1
                ActiveCell.Font.Color = vbRed
            ElseIf Selection.Value < 1 Then
                Selection.Value = 1
                nErrCount = nErrCount + 1
                ActiveCell.Font.Color = vbRed
            ElseIf Selection.Value > 4 Then
                Selection.Value = 1
                nErrCount = nErrCount + 1
                ActiveCell.Font.Color = vbRed
            End If
            .AcidType = Selection.Value

         '  now pH units
        ActiveCell.Offset(0, 1).Select
            If Not IsNumeric(Selection.Value) Then
                Selection.Value = 7
                nErrCount = nErrCount + 1
                ActiveCell.Font.Color = vbRed
            ElseIf Selection.Value < 4 Or Selection.Value > 10 Then
                Selection.Value = 7
                nErrCount = nErrCount + 1
                ActiveCell.Font.Color = vbRed
            End If
            .InputPH = Selection.Value
```

```
        ActiveCell.Offset(0, 1).Select
         '   .FeedTDS = Selection.Value 'not used, bypass it

        ActiveCell.Offset(0, 1).Select
        '   these are string - don't matter
            .FeedUnits = Selection.Value
        '   feed units can range from 1 to 3 - ppm, caco3, meq
        ActiveCell.Offset(0, 1).Select
            If Not IsNumeric(Selection.Value) Then
                Selection.Value = 1
                nErrCount = nErrCount + 1
                ActiveCell.Font.Color = vbRed
            ElseIf Selection.Value < 1 Or Selection.Value > 3 Then
                Selection.Value = 1
                nErrCount = nErrCount + 1
                ActiveCell.Font.Color = vbRed
            End If
            .nUnits = Selection.Value

        '   now flow units
        ActiveCell.Offset(0, 1).Select
            .FlowUnits = Selection.Value

        ActiveCell.Offset(0, 1).Select
            If Not IsNumeric(Selection.Value) Then
                Selection.Value = 1
                nErrCount = nErrCount + 1
                ActiveCell.Font.Color = vbRed
            ElseIf Selection.Value < 1 Or Selection.Value > 6 Then
                Selection.Value = 1
                nErrCount = nErrCount + 1
                ActiveCell.Font.Color = vbRed
            End If
            .nFlowUnits = Selection.Value

            '   now the ions
        For i = ION_NA To ION_SO3
                ActiveCell.Offset(0, 1).Select
                If Not IsNumeric(Selection.Value) Then
                    Selection.Value = 0
                    ActiveCell.Font.Color = vbRed
                ElseIf Selection.Value < 0 Then
                    Selection.Value = 0
                    ActiveCell.Font.Color = vbRed
                End If
                FeedIons(i) = Selection.Value
        Next i

            '   now include chlorine
        ActiveCell.Offset(0, 1).Select
                If Not IsNumeric(Selection.Value) Then
                    Selection.Value = 0
                    ActiveCell.Font.Color = vbRed
                    nErrCount = nErrCount + 1
                ElseIf Selection.Value < 0 Then
                    Selection.Value = 0
                    ActiveCell.Font.Color = vbRed
                    nErrCount = nErrCount + 1
                End If
        Range("User_Chlorine").Value = Selection.Value

    '   error check and stop
    '   if the nerrcount > 1 then we need to ask if we want to continue
        If nErrCount > 1 Then
            Dim strMsg As String
            Dim nResponse As Integer
            strMsg = "There were > " & nErrCount & " < errors during the load and validation of
the data set you selected.  Errors include invalid units, and negative or missing or nonnumeric "
& _
```

```
                            "values where numeric values were expected. The program tried to
correct them. " & _
                            "It might be useful to recheck the data row you selected and try
again now that the errors were corrected."
            nResponse = MsgBox(strMsg, vbAbortRetryIgnore)
            If nResponse = vbAbort Or nResponse = vbRetry Then
                Range(Cells(nRow, 1), Cells(nRow, 1)).Select
                Exit Sub
            End If
        End If

    '   now put all of the data in tha variable back onto the spreadsheet
        Range("Input_projectname").Value = .ProjectName
        Range("Input_projectDate").Value = .Projectdate

'       9 Feb 2000 - last minute update
'   need to move the flow rate after the recovery, since recovery maintains constant perm flow
        '   now must first change to gpm units
        Range("FlowUnit").Value = 1
        Flow_Select_Drop_Click
'        Range("FeedFlow").Value = .FeedGPM
'        Range("FlowUnit").Value = .nFlowUnits
'        Flow_Select_Drop_Click

        '   Change to Deg F first
        Range("Select_temperature").Value = 1
        Temp_Select_click
        Range("Input_Temp").Value = .TempF

        '   Recovery and Rejection are simple changes
        Range("Input_Recovery").Value = .Recovery
        Range("SystemRecovery").Value = .Recovery
        Range("Input_Rejection").Value = .Rejection
'       9 feb 2000 - moved the flow update to after the recovery update
        Range("FeedFlow").Value = .FeedGPM
'       now change flow units
        Range("FlowUnit").Value = .nFlowUnits
        Flow_Select_Drop_Click


        '   Change acid to new type from file
        Range("Select_acid").Value = .AcidType
        Acid_Select_Click

        '   change the set point for the pH - only useful if acid is specified
        Range("Input_Feed_pH").Value = .InputPH

        '   need to handle the units section a little differently
        Range("select_analytical").Value = .nUnits
        AnalyisGroup_Click
End With

'   Now put in the ions
'   calculation must be off
Application.Iteration = False
Application.Calculation = xlCalculationManual

'   8 July 1999 Changes
'       Added new names to spreadsheet called User_NA, User_MG etc.
'       Before 8 July 1999 copied from FeedIons() to Range("Feed_NA")
'       Now to Range("User_NA") etc.

Range("User_Na").Value = FeedIons(ION_NA)
Range("User_K").Value = FeedIons(ION_K)
Range("User_CA").Value = FeedIons(ION_CA)
Range("User_MG").Value = FeedIons(ION_MG)
Range("User_SR").Value = FeedIons(ION_SR)
Range("User_BA").Value = FeedIons(ION_BA)
Range("User_NH4").Value = FeedIons(ION_NH4)
'calculated - Range("User_H").Value = FeedIons(Ion_H)
'   11/1/99 - reset the total inorganic entry flag to NO
```

```
'   2/9/00 use the changecarbon routine here
Dim strflag As String
    strflag = Range("TotalInorganicCarbonFlag").Value
    strflag = Left(strflag, 1)
    If strflag = "y" Or strflag = "Y" Then
        ChangeCarbon
    End If
'Range("TotalInorganicCarbonFlag").Value = "N"

Range("User_HCO3").Value = FeedIons(ION_HCO3)
'calculated - Range("User_CO2").Value = FeedIons(Ion_CO2)
'calculated - Range("User_CO3").Value = FeedIons(Ion_CO3)
Range("User_BR").Value = FeedIons(ION_BR)
Range("User_CL").Value = FeedIons(ION_CL)
Range("User_F").Value = FeedIons(ION_F)
Range("User_SO4").Value = FeedIons(ION_SO4)
Range("User_NO3").Value = FeedIons(ION_NO3)
Range("User_PO4").Value = FeedIons(ION_PO4)
'calculated - Range("User_OH").Value = FeedIons(Ion_OH)
Range("User_SIO2").Value = FeedIons(ION_SIO2)
Range("User_pH").Value = FeedIons(Ion_pH)
Range("User_ORGANICS").Value = FeedIons(ION_ORGANICS)
Range("User_AG").Value = FeedIons(Ion_AG)
Range("User_AL").Value = FeedIons(ION_AL)
Range("User_AS").Value = FeedIons(Ion_AS)
Range("User_AU").Value = FeedIons(Ion_AU)
Range("User_CD").Value = FeedIons(Ion_CD)
Range("User_CR").Value = FeedIons(Ion_CR)
Range("User_CU").Value = FeedIons(Ion_CU)
Range("User_FE2").Value = FeedIons(Ion_FE2)
Range("User_FE3").Value = FeedIons(ION_FE3)
Range("User_HG").Value = FeedIons(Ion_HG)
Range("User_MN").Value = FeedIons(Ion_MN)
Range("User_NI").Value = FeedIons(Ion_NI)
Range("User_PB").Value = FeedIons(Ion_PB)
Range("User_SE").Value = FeedIons(Ion_SE)
Range("User_SN").Value = FeedIons(Ion_SN)
Range("User_TI").Value = FeedIons(Ion_TI)
Range("User_V").Value = FeedIons(Ion_V)
Range("User_ZN").Value = FeedIons(Ion_ZN)
Range("User_B4O7").Value = FeedIons(ION_B4O7)
Range("User_NO2").Value = FeedIons(ION_NO2)
Range("User_SO3").Value = FeedIons(ION_SO3)

'   End 8 July 1999 Changes

'   turn on iteration & Calculation
Application.Iteration = True
Application.Calculation = xlCalculationAutomatic
Application.ScreenUpdating = True
BoolLoading = False
'   set the flag for needs recalc
    Worksheets("Summary").Range("SystemCalculated").Value = "Needs Recalc"
    Worksheets("Streams").Range("SystemCalculated1").Value = "Needs Recalc"

Worksheets(MainSheet).Activate
Range("A1").Select
Range("D4").Select
End Sub

Sub SortbyDate()
Range("A11").Select
    Selection.Sort key1:=Range("A12"), order1:=xlDescending, key2:=Range("B12" _
        ), order2:=xlAscending, Header:=xlGuess, ordercustom:=1, MatchCase:= _
        False, Orientation:=xlTopToBottom
End Sub

Sub SortbyProject()
Range("B11").Select
    Selection.Sort key1:=Range("B12"), order1:=xlAscending, key2:=Range("A12" _
        ), order2:=xlDescending, Header:=xlGuess, ordercustom:=1, MatchCase:= _
```

```
        False, Orientation:=xlTopToBottom
End Sub

Sub SortByFlow()
Range("C11").Select
    Selection.Sort key1:=Range("C12"), order1:=xlDescending, key2:=Range( _
        "B12"), order2:=xlAscending, Header:=xlGuess, ordercustom:=1, MatchCase _
        :=False, Orientation:=xlTopToBottom
End Sub

Sub SortbyFeedTDS()
Range("I11").Select
    Selection.Sort key1:=Range("I12"), order1:=xlDescending, key2:=Range("A12" _
        ), order2:=xlAscending, Header:=xlGuess, ordercustom:=1, MatchCase:= _
        False, Orientation:=xlTopToBottom
End Sub

Sub SortbyRecovery()
Range("E11").Select
    Selection.Sort key1:=Range("E12"), order1:=xlDescending, key2:=Range("A12" _
        ), order2:=xlAscending, Header:=xlGuess, ordercustom:=1, MatchCase:= _
        False, Orientation:=xlTopToBottom
End Sub
```

## RO/NF Ions

```
Attribute VB_Name = "RONFIons"
Option Explicit
```

*These constants allow access to ions by using a meaningful value. For example, suppose the array Ions(1 to 25) contains the values of the different ions. To access the Ca concentration, we can use Ions(SP_Ca) rather than a call to Ions(3).*

```
Public Const SP_Na = 1
Public Const SP_K = 2
Public Const SP_Ca = 3
Public Const SP_Mg = 4
Public Const SP_Sr = 5
Public Const SP_Ba = 6
Public Const SP_Fe3 = 7
Public Const SP_Al = 8
Public Const SP_NH4 = 9
Public Const SP_HCO3 = 10
Public Const SP_CO2 = 11
Public Const SP_CO3 = 12
Public Const SP_Br = 13
Public Const SP_Cl = 14
Public Const SP_F = 15
Public Const SP_SO4 = 16
Public Const SP_NO3 = 17
Public Const SP_PO4 = 18
Public Const SP_SiO2 = 19
Public Const SP_Organic = 20
Public Const SP_Ag = 21
Public Const SP_As = 22
Public Const SP_Au = 23
Public Const SP_Cd = 24
Public Const SP_Cr = 25
Public Const SP_Cu = 26
Public Const SP_Fe2 = 27
Public Const SP_Hg = 28
Public Const SP_Mn = 29
Public Const SP_Ni = 30
Public Const SP_Pb = 31
Public Const SP_Se = 32
Public Const SP_Sn = 33
Public Const SP_Ti = 34
Public Const SP_V = 35
Public Const SP_Zn = 36
```

```
Public Const SP_B4O7 = 37
Public Const SP_NO2 = 38
Public Const SP_SO3 = 39
Public Const LAST_ION = 39

Type IonType
    Charge As Double
    MolWt As Double
    Name As String
End Type

Public Ions(LAST_ION) As IonType
```

*Similar approach to units*

```
'Public Const FLOW_MSEC = 1 / 60 / 264.2
Public Const FLOW_M3SEC = 0
Public Const FLOW_GPM = 1
Public Const FLOW_GPD = 2
Public Const FLOW_M3HR = 3
Public Const FLOW_M3DAY = 4
' change to L/Min from L/sec
'Public Const FLOW_LSEC = 5
Public Const FLOW_LMIN = 5
Public Const FLOW_LHR = 6
Public Const PRESS_ATM = 0
Public Const PRESS_PSI = 1
Public Const PRESS_KPA = 2
```

*A function to convert to and from ATM to Psi or back, depending on what flow unit the user selected.*

```
Public Function ATMtoPRESS(ATM, UserUnits As Integer) As Double
Dim Multiplier As Double
'   PSI if GPD or GPM, ATM else
Select Case UserUnits
    Case FLOW_GPM
        Multiplier = 14.69
    Case FLOW_GPD
        Multiplier = 14.69
    Case Else
        Multiplier = 1
End Select
ATMtoPRESS = Multiplier * ATM
End Function
```

*Converts flow to $m^3$/sec from the user flow rates on the sheet*

```
Public Function FlowM3SEC(InputFlow As Double, UserUnits As Integer) As Double
'   going from userunits to m^3/sec
Dim Multiplier As Double
Select Case UserUnits
    Case FLOW_GPM
        Multiplier = 1# / 15852# '60 * 264.2
    Case FLOW_GPD
        Multiplier = 1# / 22826880# '86400 * 264.2
    Case FLOW_M3HR
        Multiplier = 1# / 3600#
    Case FLOW_M3DAY
        Multiplier = 1# / 86400#
    'Case FLOW_LSEC
    '    Multiplier = 0.001
    Case FLOW_LMIN
        Multiplier = 1# / 60000#
    Case FLOW_LHR
        Multiplier = 1# / 3600000#
    Case Else
        Multiplier = 1
End Select
FlowM3SEC = Multiplier * InputFlow
End Function
```

*Goes back to User Flow units from $m^3$/sec – the internal units*

```
Public Function FlowUser(InputFlow As Double, UserUnits As Integer) As Double
'   going from M3/sec to the User units
Dim Multiplier As Double
Select Case UserUnits
    Case FLOW_GPM
        Multiplier = 15852# '60 * 264.2
    Case FLOW_GPD
        Multiplier = 22826880# '86400 * 264.2
    Case FLOW_M3HR
        Multiplier = 3600#
    Case FLOW_M3DAY
        Multiplier = 86400#
    'Case FLOW_LSEC
    '    Multiplier = 1000#
    Case FLOW_LMIN
        Multiplier = 60000#
    Case FLOW_LHR
        Multiplier = 3600000#
    Case Else
        Multiplier = 1
End Select
FlowUser = Multiplier * InputFlow
End Function
```

*Used to calculate osmotic pressure from a water analysis. The values are passed in the form of meq/l in an array ordered as shown in the list of variables above.*

```
'=========================================================
' OsmoticPressureMEQ (IonValues, nCount, Temperature)
'   Calculates the osmotic pressure for a stream of the
'   specified ion composition as meq/l.  Uses Van't Hoff method with
'   modified Debye Huckel activity.
'
'   Input variables:
'   meqValues()         An Array with the concentration data as meq/l
'   ncount              The highest value of the Array
'   Temperature         The temperature of the stream in deg C
'
'   Pi = cRT * Activity, where
'           R = 0.082054 liter-atm/mol-DegK
'           c = concentration, mol/liter (not equiv/liter)
'           Activity = (10^(-0.5Z^2*IonStr^0.5/(1+IonStr^0.5)))^0.14

'=========================================================
Public Function OsmoticPressureMEQ(meqValues() As Double, ByVal Temperature As Double) As Double
Dim i As Integer, z As Integer
Dim PITemp As Double, RT As Double
Dim ActCoef As Double, floatDummy As Double

    PITemp = 0#
    RT = 0.082054 * (Temperature + 273#)        ' 14.69 converts answer to psi
    RT = RT / 1000                              ' convert to moles/l from mm/l
    floatDummy = Sqr(IonicStrengthfromMeq(meqValues()))

    For i = 0 To LAST_ION
        ' skip CO2
        If i <> SP_CO2 Then
            ' note when Z=0, ActCoef = 1.0 (for SiO2)
            z = Ions(i).Charge
            If z = 0 Then z = 1
            ActCoef = (10# ^ (-0.5 * z * z * floatDummy / (1# + floatDummy))) ^ 0.14
            PITemp = PITemp + ActCoef * RT * meqValues(i) / Abs(z)                  ' divide by z
to get to moles/l, eq 2 meq/l Ca = 1 mm/l Ca
            End If
    Next i
    OsmoticPressureMEQ = PITemp
End Function
```

*The Ionic strength calculation.*

```
'=======================================================
' IonicStrengthfromMeq (floatMeq,nCount)
'
'   Calculates the ionic strength for the stream ions
'   passed to the function.
'
'   IONIC STRENGTH = 1/2 * SUMMATION [CONC * CHARGE^2]
'   Concentration expressed as moles/l (NB - not meq/l)
'   If already in meq then 1 mole/l = (meq/l)/abs(Charge)
'   so IS = 1/2 * summation[conc * abs(charge)]
'   Since data is in meq and not eq/l
'   divide by 1000 at end
'
'   Input Variables
'   floatMeq - Array of concentrations in meq/l
'=======================================================
Public Function IonicStrengthfromMeq(floatMeq() As Double) As Double
Dim floatIonStr As Double
Dim i As Integer
    floatIonStr = 0#
    For i = 0 To LAST_ION
        If i <> SP_CO2 Then
        ' ignores CO2 and organics
            If i <> SP_Organic Then
                floatIonStr = floatIonStr + floatMeq(i) * Abs(Ions(i).Charge)
            End If
        End If
    Next i
    ' Set some nonzero value just in case to avoid division by zero somewhere.
    If floatIonStr < 0.0000001 Then floatIonStr = 0.0000001
    IonicStrengthfromMeq = floatIonStr / (2# * 1000#)
End Function
```

*Ionic strength from an analysis by ion. Included for convenience*

```
'=======================================================
' IonicStrength (floatIons,nCount)
'
'   Calculates the ionic strength for the stream number
'   passed to the function.
'
'   IONIC STRENGTH = 1/2 * SUMMATION [CONC * CHARGE^2]
'   Concentration expressed as moles/l (NB - not meq/l)
'   Since data is in ppm then converted to meq/l need to
'   divide by 1000 at end
'
'   Input Variables
'   floatIons - Array of concentrations in ppmw
'=======================================================
Public Function IonicStrength(floatIONS() As Double) As Double
Dim floatIonStr As Double
Dim i As Integer
    floatIonStr = 0#
    For i = 0 To LAST_ION
        ' first check to make sure we have non-zero values
        If Ions(i).MolWt <> 0# And Ions(i).Charge <> 0# Then
            ' Convert IonValues() to moles/l on the fly
            floatIonStr = floatIonStr + floatIONS(i) / Ions(i).MolWt * Ions(i).Charge ^ 2
            End If
        Next i
    ' Set some nonzero value just in case to avoid division by zero somewhere.
    If floatIonStr < 0.0000001 Then floatIonStr = 0.0000001
    IonicStrength = floatIonStr / (2# * 1000#)
End Function
```

*Used for permeate and concentrate reconciliation after CO2 passes through membrane*

```
'=============================================================================
'   CO2toPH(floatIonStr,Temperature,Hplus,HCO3,CO2,CO3)
'
```

```
'   Returns new distrbution of carbonates and new pH based TotalCarbon coming in, CO2,
'   HCO3 and CO3 Concentrations
'   See also Carbonates, and CarbonateEquilibria
'
'   Carbonate Equilibria data using Stumm and Morgan, Aquatic Chemistry
'           pages 120-121 and 148-150
'   Linear Regression analysis of Stumm and Morgan data for K1 and K2 versus temperature
'   Estimation of Activity Change as Ionic strength from Stumm and Morgan, p 149.
'
'   Inputs are CO2, HCO3, and CO3
'   N.B.   Input data is in meq/l !!
'   Outputs are pH and new distribution of HCO3 and CO3 to fit new pH
'
'   NB  - HPlus in moles/liter is returned, not pH
'============================================================
Public Sub CO2toPH(ByVal floatIonStr As Double, ByVal Temperature As Double, _
    ByRef HPlus As Double, ByRef HCO3 As Double, ByRef CO2 As Double, ByRef CO3 As Double)
'   NB HPlus, CO3, CO2, HCO3 are changed by this subroutine - using ByRef
Dim K1 As Double, K2 As Double
Dim pK1 As Double, pK2 As Double
Dim MF_CO2 As Double, MF_CO3 As Double, MF_HCO3 As Double   'MF = Mole Fraction
Dim TotalCarbon As Double

'DEAL WITH ZERO VALUES
If HCO3 = 0 Then
    CO3 = 0
    CO2 = 0
    HPlus = 10 ^ (-7)   '   PH SET AT 7
    Exit Sub
    End If

    '   first correct K1 and K2 for temperature, using Excel derived equation
    '   pK1 = -0.007*(Temp+273)+8.4454
    '   pK2 = -0.0101*(273+Temp)+13.368

    pK1 = -0.007 * (Temperature + 273#) + 8.4454
    pK2 = -0.0101 * (Temperature + 273#) + 13.368

    '   now correct for ionic strength
    pK1 = pK1 - 0.5 * Sqr(floatIonStr) / (1# + 1.4 * Sqr(floatIonStr))
    pK2 = pK2 - 2# * Sqr(floatIonStr) / (1# + 1.4 * Sqr(floatIonStr))

    K1 = 10# ^ (-pK1)
    K2 = 10# ^ (-pK2)

    '   First get the total carbon figured out
    TotalCarbon = CO3 / 2 + HCO3 + CO2 / 2        ' 2 meqs/mol for CO3, others 1 to 1
    If HCO3 <= 0 Then HCO3 = 0.001639            '   minimum 0.1 ppm = .1/61

    'Very high pH case
    If CO3 / 2 > HCO3 Then                      ' very high ph ~10.33
        HPlus = K2 * HCO3 / (CO3 / 2)
    Else                                        ' Do it normally with CO2/HCO3 ratio
        HPlus = K1 * (CO2 / 2) / HCO3
    End If

    '   Now have the pH, figure out the new distribution of carbonates
    '   keeping CO2 constant and changing the CO3/HCO3 ratio as needed.
    MF_HCO3 = 1# / (HPlus / K1 + 1# + K2 / HPlus)             ' Stumm and Morgan, alpha1, p 120
    MF_CO3 = 1# / (HPlus ^ 2 / (K1 * K2) + HPlus / K2 + 1)
    MF_CO2 = 1# / (1# + K1 / HPlus + (K1 * K2) / HPlus ^ 2)

    '   now re-assign CO2, HCO3, and CO3 based on calculated distrbutions
    HCO3 = TotalCarbon * MF_HCO3        ' 1 meq = 1 mmole so no change needed
    CO3 = TotalCarbon * MF_CO3 * 2#     ' 1 meq = 0.5 mmoles or 1 mmole = 2 meq!
    CO2 = TotalCarbon * MF_CO2 * 2
End Sub
```

*Returns the differential pressure for an element in atm.  Needs the dpCoef and flows.*

```
'==========DELTA PRESSURE FUNCTION==========
'           DP = c * FLOW_AVERAGE ^ 1.5
'           DP in atm
'           FLOW_average in m^3/sec
'
'           since FeedFlow and ConcFlow passed as m^3/sec
'           dpCoef is taken from the Membrane Data sheet at calc time
'           and is passed to this sub as a real value
'           dpCoef is already computed in terms of m^3/sec and atm
'           Since local units are atm and m^3/sec, function expects m^3/sec as flows
'               To use other flows adjust as needed.
'
Public Function DeltaPressure(FeedFlow As Double, ConcFlow As Double, dpCoef As Double) As Double
    Dim c As Double
    Dim Flow_Average As Double
    Dim dp As Double
    c = dpCoef
    If FeedFlow < 0 Then FeedFlow = 0
    If ConcFlow <= 0 Then ConcFlow = FeedFlow
    Flow_Average = (FeedFlow + ConcFlow) / 2
    DeltaPressure = c * Flow_Average ^ 1.5
    '   done
End Function
```

*Used to set up the charge and molecular weight data for each ion*

```
' InitializeIonData
'
'   Sets initial values for all ionic species.  Called by
'   InitializeData
'
'=========================================================
Public Sub InitializeIonData()
    ' Initialize ion charge values
    Ions(SP_Ca).Charge = 2
    Ions(SP_Mg).Charge = 2
    Ions(SP_Na).Charge = 1
    Ions(SP_K).Charge = 1
    Ions(SP_NH4).Charge = 1
    Ions(SP_Sr).Charge = 2
    Ions(SP_Ba).Charge = 2
    Ions(SP_Fe3).Charge = 3
    Ions(SP_Al).Charge = 3
    Ions(SP_CO3).Charge = -2
    Ions(SP_HCO3).Charge = -1
    Ions(SP_CO2).Charge = -2
    Ions(SP_SO4).Charge = -2
    Ions(SP_Cl).Charge = -1
    Ions(SP_Br).Charge = -1
    Ions(SP_NO3).Charge = -1
    Ions(SP_F).Charge = -1
    Ions(SP_PO4).Charge = -3
    Ions(SP_SiO2).Charge = 2

    Ions(SP_Organic).Charge = 1
    Ions(SP_Ag).Charge = 1
    Ions(SP_As).Charge = 3
    Ions(SP_Au).Charge = 3
    Ions(SP_Cd).Charge = 2
    Ions(SP_Cr).Charge = 3
    Ions(SP_Cu).Charge = 2
    Ions(SP_Fe2).Charge = 2
    Ions(SP_Hg).Charge = 2
    Ions(SP_Mn).Charge = 2
    Ions(SP_Ni).Charge = 2
    Ions(SP_Pb).Charge = 2
    Ions(SP_Se).Charge = 6
    Ions(SP_Sn).Charge = 2
    Ions(SP_Ti).Charge = 2
    Ions(SP_V).Charge = 2
```

```
    Ions(SP_Zn).Charge = 2
    Ions(SP_B4O7).Charge = -2
    Ions(SP_NO2).Charge = -1
    Ions(SP_SO3).Charge = -2

    ' Initialize ion molecular wieghts
    Ions(SP_Ca).MolWt = 40.08
    Ions(SP_Mg).MolWt = 24.305
    Ions(SP_Na).MolWt = 22.99
    Ions(SP_K).MolWt = 39.098
    Ions(SP_NH4).MolWt = 18.04
    Ions(SP_Sr).MolWt = 87.62
    Ions(SP_Ba).MolWt = 137.34
    Ions(SP_Fe3).MolWt = 55.847
    Ions(SP_Al).MolWt = 26.98154
    Ions(SP_CO3).MolWt = 60.01
    Ions(SP_HCO3).MolWt = 61.01
    Ions(SP_CO2).MolWt = 44.04
    Ions(SP_SO4).MolWt = 96.06
    Ions(SP_Br).MolWt = 79.904
    Ions(SP_Cl).MolWt = 35.45
    Ions(SP_NO3).MolWt = 62.005
    Ions(SP_F).MolWt = 19#
    Ions(SP_PO4).MolWt = 94.9738
    Ions(SP_SiO2).MolWt = 60.08

    Ions(SP_Organic).MolWt = 1
    Ions(SP_Ag).MolWt = 107.868
    Ions(SP_As).MolWt = 74.9216
    Ions(SP_Au).MolWt = 196.9665
    Ions(SP_Cd).MolWt = 112.4
    Ions(SP_Cr).MolWt = 51.996
    Ions(SP_Cu).MolWt = 63.546
    Ions(SP_Fe2).MolWt = 55.847
    Ions(SP_Hg).MolWt = 200.59
    Ions(SP_Mn).MolWt = 54.938
    Ions(SP_Ni).MolWt = 58.7
    Ions(SP_Pb).MolWt = 207.2
    Ions(SP_Se).MolWt = 78.96
    Ions(SP_Sn).MolWt = 118.69
    Ions(SP_Ti).MolWt = 47.9
    Ions(SP_V).MolWt = 50.9419
    Ions(SP_Zn).MolWt = 65.38
    Ions(SP_B4O7).MolWt = 155.2358
    Ions(SP_NO2).MolWt = 46.0055
    Ions(SP_SO3).MolWt = 80.058

'   names
    Ions(SP_Na).Name = "Na = 1"
    Ions(SP_K).Name = "K = 2"
    Ions(SP_Ca).Name = "Ca = 3"
    Ions(SP_Mg).Name = "Mg = 4"
    Ions(SP_Sr).Name = "Sr = 5"
    Ions(SP_Ba).Name = "Ba = 6"
    Ions(SP_Fe3).Name = "Fe3 = 7"
    Ions(SP_Al).Name = "Al = 8"
    Ions(SP_NH4).Name = "NH4 = 9"
    Ions(SP_HCO3).Name = "HCO3 = 10"
    Ions(SP_CO2).Name = "CO2 = 11"
    Ions(SP_CO3).Name = "CO3 = 12"
    Ions(SP_Br).Name = "Br = 13"
    Ions(SP_Cl).Name = "Cl = 14"
    Ions(SP_F).Name = "F = 15"
    Ions(SP_SO4).Name = "SO4 = 16"
    Ions(SP_NO3).Name = "NO3 = 17"
    Ions(SP_PO4).Name = "PO4 = 18"
    Ions(SP_SiO2).Name = "SiO2 = 19"
    Ions(SP_Organic).Name = "Organic = 20"
    Ions(SP_Ag).Name = "Ag = 21"
    Ions(SP_As).Name = "As = 22"
    Ions(SP_Au).Name = "Au = 23"
```

```
    Ions(SP_Cd).Name = "Cd = 24"
    Ions(SP_Cr).Name = "Cr = 25"
    Ions(SP_Cu).Name = "Cu = 26"
    Ions(SP_Fe2).Name = "Fe2 = 27"
    Ions(SP_Hg).Name = "Hg = 28"
    Ions(SP_Mn).Name = "Mn = 29"
    Ions(SP_Ni).Name = "Ni = 30"
    Ions(SP_Pb).Name = "Pb = 31"
    Ions(SP_Se).Name = "Se = 32"
    Ions(SP_Sn).Name = "Sn = 33"
    Ions(SP_Ti).Name = "Ti = 34"
    Ions(SP_V).Name = "V = 35"
    Ions(SP_Zn).Name = "Zn = 36"
    Ions(SP_B4O7).Name = "B4O7 = 37"
    Ions(SP_NO2).Name = "NO2 = 38"
    Ions(SP_SO3).Name = "SO3 = 39"

End Sub
```

*Used to read in the water analysis from the Analysis sheet. Range names refer to the Treated column*

```
Public Sub GetTreatedIons(ByRef floatIONS() As Double)
    floatIONS(SP_Na) = Range("Treated_Na").Value
    floatIONS(SP_K) = Range("Treated_K").Value
    floatIONS(SP_Ca) = Range("Treated_CA").Value
    floatIONS(SP_Mg) = Range("Treated_MG").Value
    floatIONS(SP_Sr) = Range("Treated_SR").Value
    floatIONS(SP_Ba) = Range("Treated_BA").Value
    floatIONS(SP_NH4) = Range("Treated_NH4").Value
    floatIONS(SP_HCO3) = Range("Treated_HCO3").Value
    floatIONS(SP_CO2) = Range("Treated_CO2").Value
    floatIONS(SP_CO3) = Range("Treated_CO3").Value
    floatIONS(SP_Br) = Range("Treated_BR").Value
    floatIONS(SP_Cl) = Range("Treated_CL").Value
    floatIONS(SP_F) = Range("Treated_F").Value
    floatIONS(SP_SO4) = Range("Treated_SO4").Value
    floatIONS(SP_NO3) = Range("Treated_NO3").Value
    floatIONS(SP_PO4) = Range("Treated_PO4").Value
    floatIONS(SP_SiO2) = Range("Treated_SIO2").Value
    floatIONS(SP_Organic) = Range("Treated_ORGANICS").Value
    floatIONS(SP_Ag) = Range("Treated_AG").Value
    floatIONS(SP_Al) = Range("Treated_AL").Value
    floatIONS(SP_As) = Range("Treated_AS").Value
    floatIONS(SP_Au) = Range("Treated_AU").Value
    floatIONS(SP_Cd) = Range("Treated_CD").Value
    floatIONS(SP_Cr) = Range("Treated_CR").Value
    floatIONS(SP_Cu) = Range("Treated_CU").Value
    floatIONS(SP_Fe2) = Range("Treated_FE2").Value
    floatIONS(SP_Fe3) = Range("Treated_FE3").Value
    floatIONS(SP_Hg) = Range("Treated_HG").Value
    floatIONS(SP_Mn) = Range("Treated_MN").Value
    floatIONS(SP_Ni) = Range("Treated_NI").Value
    floatIONS(SP_Pb) = Range("Treated_PB").Value
    floatIONS(SP_Se) = Range("Treated_SE").Value
    floatIONS(SP_Sn) = Range("Treated_SN").Value
    floatIONS(SP_Ti) = Range("Treated_TI").Value
    floatIONS(SP_V) = Range("Treated_V").Value
    floatIONS(SP_Zn) = Range("Treated_ZN").Value
    floatIONS(SP_B4O7) = Range("Treated_B4O7").Value
    floatIONS(SP_NO2) = Range("Treated_NO2").Value
    floatIONS(SP_SO3) = Range("Treated_SO3").Value
End Sub


'   ConvertMEQtoPPM(floatMEQ() As Double, ByRef floatIons() As Double)
'   floatMeq() is the data coming in with the analysis as meq/l,
'   the floatIons() variable gets the new data put in it as ppm
'
'   eg  Ca as 2 meq/l = 2 * 20 = 40 ppm
'       SiO2 as 1 meq/l = 1 * 60 = 60 ppm
'=======================================================
```

```
Public Sub ConvertMEQtoPPM(MEQ() As Double, ByRef floatIONS() As Double)
Dim i As Integer

    For i = 1 To LAST_ION
        If Ions(i).Charge <> 0 Then
            floatIONS(i) = MEQ(i) * Ions(i).MolWt / Abs(Ions(i).Charge)
        Else
            floatIONS(i) = MEQ(i)
        End If
    Next i
End Sub


'=======================================================
'   ConvertPPMtoMEQ(byref MEQ() As Double, Ions() As Double)
'   Ions() is the data coming in with the analysis as ppm
'   the MEQ() variable gets the new data put in it as meq/l
'
'   eg  Ca as 40 ppm, 40 /20 = 2
'       SiO2 as 90 ppm, 90 / 60 = 1.5
'=======================================================
Public Sub ConvertPPMtoMEQ(ByRef MEQ() As Double, floatIONS() As Double)
Dim i As Integer

    For i = 1 To LAST_ION
        If Ions(i).Charge <> 0 Then
            MEQ(i) = floatIONS(i) / (Ions(i).MolWt / Abs(Ions(i).Charge))
            'Debug.Print Ions(i).Name, floatIONS(i), MEQ(i)
        Else
            MEQ(i) = floatIONS(i)
        End If
    'If boolDebug Then Debug.Print i, floatIons(i), MEQ(i), Ions(i).MolWt, Ions(i).Charge
    Next i
End Sub

Public Function TDSfromMEQ(floatMeq() As Double) As Double
Dim i As Integer
Dim floatTemp As Double
Dim floatDummy As Double
    floatTemp = 0#
    For i = 1 To LAST_ION
        If Ions(i).Charge <> 0 Then
            floatDummy = floatMeq(i) * Ions(i).MolWt / Abs(Ions(i).Charge)
        End If
        floatTemp = floatTemp + floatDummy
    Next i
    '   now remove CO2
    TDSfromMEQ = floatTemp - (floatMeq(SP_CO2) * Ions(SP_CO2).MolWt / Abs(Ions(SP_CO2).Charge))
End Function

Public Function TDS(floatIONS() As Double) As Double
Dim i As Integer
Dim floatTemp As Double
    floatTemp = 0#
    For i = 1 To LAST_ION
        floatTemp = floatTemp + floatIONS(i)
    Next i
TDS = floatTemp
End Function
```

*This function is the blending function. It takes the recycle stream and blends it with the feed stream to get the "Net Feed" going to the membranes.*

```
Public Sub GetNetFeed(ByRef NetIons() As Double, ByRef NewFlow As Double, _
    FeedIons() As Double, FlowFeed As Double, _
    RecycleIons() As Double, FlowRecycle As Double)
Dim n As Integer
Dim Istr As Double
If FlowRecycle < 0 Then FlowRecycle = 0
NewFlow = FlowFeed + FlowRecycle
```

```
If FlowRecycle <= 0 Then
    For n = 1 To LAST_ION
        NetIons(n) = FeedIons(n)
    Next n
Else
    For n = 1 To LAST_ION
        NetIons(n) = (FlowFeed * FeedIons(n) + FlowRecycle * RecycleIons(n)) / NewFlow
    Next n
    '   may need to deal with the change in pH and carbonates as the concentration changes
    '   but for calc purposes pH is really ignored as it is not in the list of ions
    '    iStr = IonicStrengthfromMeq(NetIons())
    '    CO2toPH iStr, Temperature, FeedpH, NetIons(ION_HCO3), NetIons(ION_CO2), NetIons(ION_CO3)
End If
End Sub
```

*Balances the permeate analysis based on both anions and cations (for RO) or for anions as the controlling ions (NF)*

```
Public Sub BalanceAnalysis(ByRef floatIONS() As Double, Optional boolNF As Boolean)
'   used to balance the PermMeq() ions during the calculation phase
'   argument is expected in the form of meq(s)
'   the floations() is directly modified
'   SiO2, CO2, and organics are ignored.
Dim n As Integer
Dim Mult As Double
Dim sumCations As Double
Dim sumAnions As Double
sumAnions = 0
sumCations = 0
'   cations
For n = 1 To SP_NH4
    sumCations = sumCations + floatIONS(n)
Next n
For n = SP_Ag To SP_Zn
    sumCations = sumCations + floatIONS(n)
Next n

'   anions (skip CO2)
For n = SP_CO3 To SP_PO4
    sumAnions = sumAnions + floatIONS(n)
Next n

'   go back for HCO3 itself
sumAnions = sumAnions + floatIONS(SP_HCO3)

For n = SP_B4O7 To SP_SO3
    sumAnions = sumAnions + floatIONS(n)
Next n

'   now have the sums, if zero we are in trouble somewhere - get out to avoid an error message
If sumCations = 0 Or sumAnions = 0 Then Exit Sub

'   how will we process the information
If boolNF Then
'   balance ONLY on the anions. Anions control passage so the cations are raised or lowered to
match
    ' example sum cations = 9, sumanions = 6, mult = 6/9 to give equality
    Mult = sumAnions / sumCations
    For n = 1 To SP_NH4
        floatIONS(n) = floatIONS(n) * Mult
    Next n

    For n = SP_Ag To SP_Zn
        floatIONS(n) = floatIONS(n) * Mult
    Next n

Else
    '   standard RO
    ' example sum cations = 9, sum anions = 6. 15/2 = 7.5, 7.5/9 is mult
    Mult = ((sumCations + sumAnions) / 2) / sumCations
```

```
    For n = 1 To SP_NH4
        floatIONS(n) = floatIONS(n) * Mult
    Next n

    For n = SP_Ag To SP_Zn
        floatIONS(n) = floatIONS(n) * Mult
    Next n

    '   now anions
    Mult = ((sumCations + sumAnions) / 2) / sumAnions
    For n = SP_CO3 To SP_PO4
        floatIONS(n) = floatIONS(n) * Mult
    Next n
    floatIONS(SP_HCO3) = floatIONS(SP_HCO3) * Mult
    For n = SP_B4O7 To SP_SO3
        floatIONS(n) = floatIONS(n) * Mult
    Next n
End If

End Sub
```

*Used to print intermediate values to the immediate window during debugging*

```
Public Sub DebugIons(FIons() As Double, PIons() As Double, CIons() As Double)
Debug.Print "Descr", "Feed", "Perm", "Conc"
Dim n As Integer
For n = 1 To LAST_ION
    Debug.Print Ions(n).Name, Format(FIons(n), "#.##"), Format(PIons(n), "#.####"),
Format(CIons(n), "#.##")
Next n
End Sub
```

## RONFCalc

This is the main engine for calculating the system values. Calculates the RO or NF design, calculates the solubility values by calling the Solver, and puts the data on the spreadsheet.

```
Attribute VB_Name = "RONFCalc"
Option Explicit
Public Const boolDebug = False
Public Const boolDebug2 = False
Public boolAbortScaleCalcs As Boolean
Public Const TYPE_CA = 1
Public Const TYPE_TFC = 2
Public Const TYPE_NF = 3
Public ATempCorr As Double
Public BTempCorr As Double
Public Temperature As Double
Public FeedpH As Double

Public boolBadDesign As Boolean
Public boolFullyCalcd As Boolean
Dim FeedMeq(LAST_ION) As Double
Dim PermMeq(LAST_ION) As Double
Dim MembMeq(LAST_ION) As Double
Dim MeqDiff(LAST_ION) As Double
Dim ConcMeq(LAST_ION) As Double
Dim NetFeedMeq(LAST_ION) As Double  ' this is the blend of feed and recycle

Dim FeedFlow As Double
Dim PermFlow As Double
Dim ConcFlow As Double
Dim RecycleFlow As Double
Dim NetFlow As Double
Public B_Ions(LAST_ION) As Double

'   variables used for getting the three carbonate balances
```

```
'   1 = netfeed, 2 = concentrate, 3 = permeate
Dim Istr(3) As Double
Dim HPlus(3) As Double
Dim PPI(3) As Double
```

*The variable which corresponds to one (1) vessel in a bank*

```
Type BigElementType
    Area_M2 As Double
    Avalue As Double
    BValue As Double
    B_Ion(LAST_ION) As Double
    dpCoef As Double
    BetaCoef As Double
    ' in the form of exp(BetaCoef * Recovery), typically dependent on element length
                            ' 40 inch = 0.7, eg at 15% recovery, exp(0.7 *.15) = exp(.105) = 1.11
                            '  betaCoef = -0.3492Ln(x) + 2.0152 where x is length in inches
End Type

Dim MyElement As BigElementType
```

*A whole bank variable*

```
Type BankType
    nVessels As Integer
    FeedFlow As Double
    PermFlow As Double
    ConcFlow As Double

    '   pressure information
    FeedPress As Double
    ConcPress As Double

    '   now as ions
    FeedMeq(LAST_ION) As Double
    ConcMeq(LAST_ION) As Double
    PermMeq(LAST_ION) As Double

    ConcPolarization As Double
    PTransMemb As Double
End Type
```

*The complete system is contained in this variable*

```
Type ArrayType
    nBanks As Integer
    Bank(4) As BankType
    Element As BigElementType

    '   Flows in m^3/sec
    FeedFlow As Double
    PermFlow As Double
    ConcFlow As Double
    RecycleFlow As Double

    '   Press in atm
    FeedPress As Double
    ConcPress As Double

    FeedIons(LAST_ION) As Double
    NetFeedIons(LAST_ION) As Double
    PermIons(LAST_ION) As Double
    ConcIons(LAST_ION) As Double
End Type
```

*Carray is the actual name as used. Can't use the word "Array" alone as it is a reserved word.*

```
Dim CArray As ArrayType
```

*Pressing the calculation button launches this routine*

```
Sub MainCalc()
Dim Pressure As Double
Dim NewPressure As Double
Dim StartPressure As Double
Dim nIters As Long

Dim boolCalcAlways As Boolean
boolAbortScaleCalcs = False
nIters = 0
```

*Get all of the data into the system*

```
    InitializeIonData
    GetElementData
    '  get element needs to run first!
    GetSystemData
```

*Guess a pressure in the "guess" function*

```
    Pressure = FirstGuess
    StartPressure = Pressure
```

*Call the main routine*

```
    NewPressure = CalcSystem(StartPressure)
```

*Display the data for flows and pressure, etc.*

```
    PutDataOnSheets
    boolFullyCalcd = False
    If boolBadDesign = False Then
```

*Run the solubility calcs – ask first if the user wants to proceed if not TRUE*

```
        boolCalcAlways = Range("CalcAlways").Value
            If boolCalcAlways = True Then
                RunSolver (0)
                boolFullyCalcd = True
            Else
                If MsgBox("Calculate all saturation data at this time? This takes a few seconds.
To suppress this message set the F2 Cell on the Calc sheet to true.", vbOKCancel) = vbOK Then
                    RunSolver 0
                    boolFullyCalcd = True
                End If
            End If
    End If
    Worksheets("System Design").Activate
    Range("FeedPressure").Select
    ShowNotes
End Sub
```

*Show status – all OK if calc'd OK, etc. Labels appear on the spreadsheet*

```
Sub ShowNotes()
    If boolFullyCalcd = True Then
            Worksheets("Summary").Range("SystemCalculated").Value = "YES"
            Worksheets("Streams").Range("SystemCalculated1").Value = "YES"
            Worksheets("System Design").lblRecalc.Visible = False
            Worksheets("System Design").msgRecalc.Visible = False
    ElseIf boolBadDesign = True Then
            Worksheets("Summary").Range("SystemCalculated").Value = "Not Valid"
            Worksheets("Streams").Range("SystemCalculated1").Value = "Not Valid"
            Worksheets("System Design").lblRecalc.Visible = True
            Worksheets("System Design").msgRecalc.Visible = True
    Else
            Worksheets("Summary").Range("SystemCalculated").Value = "SI Not valid"
            Worksheets("Streams").Range("SystemCalculated1").Value = "SI Not valid"
            Worksheets("System Design").lblRecalc.Visible = False
            Worksheets("System Design").msgRecalc.Visible = False
    End If
End Sub
```

*This is the main calculation engine*

```
'   set up the counters
'   Calc method - converge to a constant permeate Quality and balanced analysis
'   Flow to bank must be divided by number of vessels to do one vessel
'   1.  Set Loop variable - PermeateBal
'   2.  calculate a Beta (concentration polarization) from BetaCoef and Recovery for this bank
'   3.  Calculate a net membrane surface concentration by ion from
'           log mean avg conc = (Cf - Cb) / ln(Cf/Cb)
'           net membrane conc = membMeq() = Beta * LMAvgConc
'           Difference between membrane surface and permeate = meqDiff()
'           MeqDiff() = membMeq() - permMeq()
'   4.  calculate each ion in the permeat using Avalue, Bvalue, B_Ion,
'           PermMeq() = meqDiff() * B_Ion() * B / (P*A + B)
'   5.  Since CO2 is unchanged from Feed to permeate, we need to exclude the CO2 from the Balance
'           and then recalc the carbonate equilibirum based on the new distribution of HCO3 and
CO2
'   6.  Adjust the permeate balance by adding some cations and subtracting and equal amount of
anions (or vice versa)
'           '   for NF, only the anions are adjusted since cations are not really rejected only
dragged along for charge neutrality
'   7.  With the new PermMeq() compute a new concMeq() using Perm and Feed flows
'   8.  Compute a new ConcPi and PermPi to get a new Net PI (FeedPI - ConcPI)/log(FeedPi/ConcPI)
- PermPi
'           Pnet = Pressure - NetPI - DP/2
'   9.  Compute new PermFlow using DP from old FeedFlow and ConcFlow
'           PermFlow = PNet * AValue * Area
'   10. Compute new ConcFlow and New Recovery froim PermFlow
'   11. Recompute DP with New flows
'   12. Check PermeateTDS - OldPermeateTDS, if less than 1% we are done with this bank
'   13. Compute Total bank PermFlow as PermFlow * NVessels, FeedFlow
'   Bank 2-4 in same way
'   14. Add up all permFlows.
'   15. Compare desired TargetFlow with calculated PermFlow
'           '   NewPressure = TargetFlow/PermFlow * OldPressure
'   16  If targetFlow - PermFlow < 1% then we are done, otherwise try again.
Function CalcSystem(LastPress As Double) As Double
Dim n As Integer
Dim nBank As Integer
Dim i As Integer
Dim nNumbanks As Integer
'   convergence variables
Dim DeltaFlow As Double
Dim DeltaPercent As Double
Dim TargetPermFlow As Double
Dim CalcPermFlow As Double
Dim OldPressure As Double
Dim NewPressure As Double
Dim nIters As Integer

Dim Area As Double
Dim A As Double
Dim B As Double

Dim Beta As Double
Dim BetaCoef As Double
Dim dp As Double
Dim dpCoef As Double
Dim PI As Double
Dim CF As Double
Dim PermeateBal As Double

Dim TDS As Double
Dim SaltPass As Double
Dim Pressure As Double
Dim PNet As Double
Dim Recovery As Double


    Pressure = LastPress
    NewPressure = Pressure
```

```
    boolBadDesign = False

'   set up values used throughout the system
    nNumbanks = CArray.nBanks
    Area = MyElement.Area_M2            '  Area in One Vessel, always the same in any system
    A = MyElement.Avalue                '   always the same
    B = MyElement.BValue                '   always the same
    dpCoef = MyElement.dpCoef           '   always the same
    BetaCoef = MyElement.BetaCoef       '   always the same


    '   start with known flow data, which is already stored in Bank(1) from FirstGuess
    '   convert back to single vessel flow
    FeedFlow = CArray.Bank(1).FeedFlow / CArray.Bank(1).nVessels
    ConcFlow = CArray.Bank(1).ConcFlow / CArray.Bank(1).nVessels
    PermFlow = FeedFlow - ConcFlow

    TargetPermFlow = CArray.PermFlow
    CalcPermFlow = 0

    '   get first dp
    dp = DeltaPressure(FeedFlow, ConcFlow, dpCoef)

    '   ions in first bank
    For i = 1 To LAST_ION
        FeedMeq(i) = CArray.Bank(1).FeedMeq(i)
        PermMeq(i) = CArray.Bank(1).PermMeq(i)
        ConcMeq(i) = CArray.Bank(1).ConcMeq(i)
    Next i

If boolDebug Then
    Debug.Print "area", Area, Area * 10.76
    Debug.Print "dp", dp, dp * 14.69
    Debug.Print "feedflow", FeedFlow, FeedFlow * 60 * 264.2
    Debug.Print "permflow", PermFlow, PermFlow * 60 * 264.2
    Debug.Print "Pi", PI, PI * 14.69
    Debug.Print "A", A
    Debug.Print "TempCorr", ATempCorr
    Debug.Print "temperature", Temperature
    Debug.Print "pressure", Pressure, Pressure * 14.69
End If

nIters = 0
```

*Set the convergence test right here. 0.005 is the default as released – ie pressures agree within 0.5%*

```
Do While Abs(TargetPermFlow - CalcPermFlow) / TargetPermFlow > 0.005     ' 1 percent convergence
    nIters = nIters + 1
    If nIters > 100 Then
        MsgBox ("The program failed too converge - the results may be inaccurate.")
        Exit Do
    End If

    Pressure = NewPressure

    '   Bank 1 = must always be done
    For nBank = 1 To nNumbanks
            '   banks are calculated for one (1) vessel only, so need to adjust flows at
beginning and end
            '   Area is area of one (1) vessel
            '   first get the data into the local variables
            FeedFlow = CArray.Bank(nBank).FeedFlow / CArray.Bank(nBank).nVessels
'       N.B if the following two lines are included, then no convergence is possible
'       Don't try it. It will be done the next pass through
'            ConcFlow = CArray.Bank(nBank).ConcFlow / CArray.Bank(nBank).nVessels
'            PermFlow = FeedFlow - ConcFlow

            '   now calculate the neq values for the bank
            Recovery = PermFlow / FeedFlow
            Beta = Exp(BetaCoef * Recovery)
            For i = 1 To LAST_ION
                MeqDiff(i) = (FeedMeq(i) + ConcMeq(i)) / 2 - PermMeq(i)
            Next i
```

```
            '   Pressure will change each time
            '   get dp for this bank
            dp = DeltaPressure(FeedFlow, ConcFlow, dpCoef)
            PNet = Pressure - OsmoticPressureMEQ(MeqDiff(), Temperature) - dp / 2
```

*The main flow rate equation*

```
            PermFlow = A * Area * PNet
            'If boolDebug2 Then Debug.Print
            '   if permflow is going negative (we assume this is impossible)

            If PermFlow <= 0 Then
                PermFlow = 0
                Recovery = 0
                Beta = 1
            Else
                Recovery = PermFlow / FeedFlow
                Beta = Exp(BetaCoef * Recovery)
            End If
            '   now get a salt passage rate
            '   if this is less than 0 then this means the dp and osmotic pressure is higher than
the
            '   applied pressure.  In reality this can happen so we need to deal with the problem
            '   because in low recovery but multiple bank situations this is frequently a result.
            '   Physically the situation is that at the applied pressure needed to make the water
flow
            '   there is not enough pressure to force the desired flow through the array of
banks.
            '   Need to give the user the message and ask him/her to try again.
            SaltPass = Beta * B / (PNet * A + B)
            If boolDebug2 Then Debug.Print nBank, "DP = "; dp * 14.69, SaltPass, PNet, PermFlow *
60 * 264.2
            For i = 1 To LAST_ION
                '   first get ion data for the bank
                FeedMeq(i) = CArray.Bank(nBank).FeedMeq(i)
                'If ConcMeq(i) < 0 Then MsgBox "Less than 0"
                ConcMeq(i) = CArray.Bank(nBank).ConcMeq(i)
                PermMeq(i) = CArray.Bank(nBank).PermMeq(i)
                '   deal with 0 values
                If i = SP_CO2 Then
                    '   first fix up CO
                    PermMeq(SP_CO2) = FeedMeq(SP_CO2)
                Else
                    '   skip 0 values
                    If FeedMeq(i) = 0 Or ConcMeq(i) = 0 Then
                        MeqDiff(i) = 0
                        PermMeq(i) = 0
                    Else
```

*The main salt flux/rejection equations*

```
                        MembMeq(i) = Beta * (FeedMeq(i) - ConcMeq(i)) / Log(FeedMeq(i) /
ConcMeq(i))
                        MeqDiff(i) = MembMeq(i) - PermMeq(i)
                        '   now need to check for reverse perm flow and deal with it
                        If MeqDiff(i) > 0 And SaltPass > 0 Then
                            PermMeq(i) = MeqDiff(i) * B_Ions(i) * B / (PNet * A + B)
                        Else
                            '   can't be negative so there is no change
                            PermMeq(i) = FeedMeq(i)
                        End If

                    End If
                End If
            Next i



            '   now make sure there is an ionic balance, the True/false is for NF, True = NF,
false = RO
            BalanceAnalysis PermMeq(), False
            '   now have all of the ions into the PermMeq() variable
```

```
            '   get ConcMeq() by mass balance
            ConcFlow = FeedFlow - PermFlow
            '   IF CONCFLOW LESS THAN ZERO THEN RESET
            If ConcFlow < 0 Then ConcFlow = 0.1 * FeedFlow
            For i = 1 To LAST_ION
                ConcMeq(i) = (FeedFlow * FeedMeq(i) - PermFlow * PermMeq(i)) / ConcFlow
            Next i
            If boolDebug Then DebugIons FeedMeq(), PermMeq(), ConcMeq()

            '   we are done with the bank, need to pass data to next bank and update the present
bank
            '   Assign data to present Bank
            With CArray.Bank(nBank)
                .ConcFlow = ConcFlow * .nVessels
                .PermFlow = PermFlow * .nVessels
                .FeedFlow = FeedFlow * .nVessels
                .FeedPress = Pressure
                .ConcPress = Pressure - dp
                .ConcPolarization = Beta
                .PTransMemb = PNet
                For i = 1 To LAST_ION
                    .ConcMeq(i) = ConcMeq(i)
                    .PermMeq(i) = PermMeq(i)
                    .FeedMeq(i) = FeedMeq(i)
                Next i
                If boolDebug Then DebugIons .FeedMeq(), .PermMeq(), .ConcMeq()
            End With
            '   now assign values to next bank
            If nNumbanks >= nBank + 1 Then
                With CArray.Bank(nBank + 1)
                    .FeedFlow = CArray.Bank(nBank).ConcFlow
                    .FeedPress = CArray.Bank(nBank).ConcPress
                    For i = 1 To LAST_ION
                        .FeedMeq(i) = ConcMeq(i)
                        FeedMeq(i) = .FeedMeq(i)
                        '   these are still there from the last iteration
                        PermMeq(i) = CArray.Bank(nBank + 1).PermMeq(i)
                        ConcMeq(i) = CArray.Bank(nBank + 1).ConcMeq(i)
                    Next i
                    Pressure = .FeedPress
                    FeedFlow = .FeedFlow / .nVessels
                    PermFlow = .PermFlow / .nVessels
                    ConcFlow = .ConcFlow / .nVessels
                End With
                '   now recalculate the differential pressure loss for this NEXT bank
                dp = DeltaPressure(FeedFlow, ConcFlow, dpCoef)
            End If
    Next nBank

    '   now redo the recycle if there is any
    If CArray.RecycleFlow > 0 Then
        '   GetNetFeed(ByRef NetIons() As Double, ByRef NewFlow As Double, _
            FeedIons() As Double, FlowFeed As Double, _
            RecycleIons() As Double, FlowRecycle As Double)
            GetNetFeed NetFeedMeq(), FeedFlow, FeedIons(), CArray.FeedFlow, ConcMeq(),
RecycleFlow
            For i = 1 To LAST_ION
                FeedMeq(i) = NetFeedMeq(i)
            Next i
    End If

'   set up convergence test variables here and move pressure up or down as needed
    CalcPermFlow = 0
    For n = 1 To CArray.nBanks
        CalcPermFlow = CalcPermFlow + CArray.Bank(n).PermFlow
    Next n

    If CalcPermFlow <= 0 Then
        boolBadDesign = True
        Exit Do
    End If
```

```
    '   calcpermflow is the perm flow we actually got
    '   targetpermflow is what we actually want
    DeltaFlow = TargetPermFlow - CalcPermFlow
    DeltaPercent = Abs(TargetPermFlow - CalcPermFlow) / TargetPermFlow
    '   use a brute force convergence - actually faster and more sure.
    OldPressure = NewPressure
    Select Case DeltaFlow
        Case Is > 0                 '   the pressure needs to go up
            If DeltaPercent > 1 Then
                 NewPressure = OldPressure + OldPressure * (TargetPermFlow / CalcPermFlow)
            ElseIf DeltaPercent > 0.1 Then
                NewPressure = OldPressure + OldPressure * (TargetPermFlow / CalcPermFlow) / 10
            ElseIf DeltaPercent > 0.01 Then
                NewPressure = OldPressure + OldPressure * (TargetPermFlow / CalcPermFlow) / 200
            Else
                NewPressure = OldPressure + OldPressure * (TargetPermFlow / CalcPermFlow) / 500
            End If
        Case Is < 0                 '   the pressure needs to go down
            If DeltaPercent > 1 Then
                 NewPressure = OldPressure - OldPressure * (TargetPermFlow / CalcPermFlow)
            ElseIf DeltaPercent > 0.1 Then
                NewPressure = OldPressure - OldPressure * (TargetPermFlow / CalcPermFlow) / 10
            ElseIf DeltaPercent > 0.01 Then
                NewPressure = OldPressure - OldPressure * (TargetPermFlow / CalcPermFlow) / 200
            Else
                NewPressure = OldPressure - OldPressure * (TargetPermFlow / CalcPermFlow) / 500
            End If
    End Select
    If boolDebug2 Then
        Debug.Print "CalcPermFlow = ", CalcPermFlow, "Targetpermflow = ", TargetPermFlow
        Debug.Print "Old pressure was = ", OldPressure, "New Pressure = ", NewPressure
    End If
```

*The comparison test takes place at this point. If the pressure this pass is close enough to the last pass we are done.*

```
Loop

'   save the pressure
CArray.FeedPress = NewPressure
CArray.Bank(1).FeedPress = NewPressure
'   now put the analytical back into the main variable as ppm - since all calcs above are meq()
'   first the net feed stream
ConvertMEQtoPPM CArray.Bank(1).FeedMeq(), CArray.NetFeedIons()

'   the last bank is the same as the concentrate
ConvertMEQtoPPM CArray.Bank(nNumbanks).ConcMeq(), CArray.ConcIons()
'   need to get the permeate by going thru all of the banks
    For i = 1 To LAST_ION
        If PermMeq(i) > 0 Then
            PermMeq(i) = 0
            For n = 1 To nNumbanks
                PermMeq(i) = PermMeq(i) + CArray.Bank(n).PermFlow * CArray.Bank(n).PermMeq(i)
            Next n
            PermMeq(i) = PermMeq(i) / CArray.PermFlow
        Else
            '   get rid of any inadvertent negative values
            PermMeq(i) = 0
        End If
    Next i
    '   Now check if the Permeate TDS is 0 - this happens when there are too many membranes in a
system and
    '   there is not enough pressure to overcome the osmotic pressure
    '   Regular occurence with seawater systems and too many membranes
    TDS = TDSfromMEQ(PermMeq())
    If TDS <= 0 Then
        MsgBox "The permeate TDS cannot be converged. Try reducing the number of elements or
reducing the number of vessesls. This situation often results from high TDS feed and too much
membrane area. The calculations are not valid at this time."
        boolBadDesign = True
```

```
    End If
    '   now redo once again the carbonate balances for netfeed, perm, and concentrate
    '   CO2toPH(ByVal floatIonStr As Double, ByVal Temperature As Double, _
        ByRef HPlus As Double, ByRef HCO3 As Double, ByRef CO2 As Double, ByRef CO3 As Double)
    '   NB HPlus, CO3, CO2, HCO3 are changed by this subroutine - using ByRef, HPlus as [Hplus],
not pH
    Istr(1) = IonicStrengthfromMeq(CArray.Bank(1).FeedMeq())
    Istr(2) = IonicStrengthfromMeq(CArray.Bank(nNumbanks).ConcMeq())
    Istr(3) = IonicStrengthfromMeq(PermMeq())
    CO2toPH Istr(1), Temperature, HPlus(1), CArray.Bank(1).FeedMeq(SP_HCO3),
CArray.Bank(1).FeedMeq(SP_CO2), CArray.Bank(1).FeedMeq(SP_CO3)
    CO2toPH Istr(2), Temperature, HPlus(2), CArray.Bank(nNumbanks).ConcMeq(SP_HCO3),
CArray.Bank(nNumbanks).ConcMeq(SP_CO2), CArray.Bank(nNumbanks).ConcMeq(SP_CO3)
    CO2toPH Istr(3), Temperature, HPlus(3), PermMeq(SP_HCO3), PermMeq(SP_CO2), PermMeq(SP_CO3)
    ConvertMEQtoPPM PermMeq(), CArray.PermIons()
    PPI(1) = 14.69 * OsmoticPressureMEQ(CArray.Bank(1).FeedMeq(), Temperature)
    PPI(2) = 14.69 * OsmoticPressureMEQ(CArray.Bank(nNumbanks).ConcMeq(), Temperature)
    PPI(3) = 14.69 * OsmoticPressureMEQ(PermMeq(), Temperature)

CalcSystem = CArray.Bank(1).FeedPress
End Function
```

*Subs to display the data on the Summary and Streams Sheets*

```
Sub PutDataOnSheets()
'   Puts the calculated data back onto the spreadsheet
Dim nCol As Integer, Pcol As Integer, Ccol As Integer
Dim CRow As Integer
Dim n As Integer
Dim c As Integer
Dim dummy As Double
Dim S As Worksheet

'   turn off updating to avoid slowing down the update
Application.ScreenUpdating = False
Set S = Worksheets("Streams")

nCol = Range("StartNetFeed").Column
Ccol = Range("StartConcentrate").Column
Pcol = Range("StarrtPermeate").Column
CRow = Range("StartNetFeed").Row

With CArray
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Na)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Na)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Na)
'   Potassium -k
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_K)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_K)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_K)
'   Calcium -Ca
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Ca)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Ca)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Ca)
'   Magnesium -Mg
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Mg)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Mg)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Mg)
'   Strontium -Sr
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Sr)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Sr)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Sr)
'   Barium -Ba
```

```
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Ba)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Ba)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Ba)
'   Aluminum - Al+++
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Al)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Al)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Al)
'   Ferric Iron - Fe+++ (as Fe)
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Fe3)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Fe3)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Fe3)
'   Ammonia - NH4 (as NH4)
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_NH4)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_NH4)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_NH4)
'   Hydrogen Ion - H
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = HPlus(1)
        S.Cells(CRow, Ccol).Value = HPlus(2)
        S.Cells(CRow, Pcol).Value = HPlus(3)
'   Sum all cations (incl metals)
'       done on the sheet

'   Bicarbonate -HC03
        CRow = CRow + 3
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_HCO3)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_HCO3)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_HCO3)
'   Carbon Dioxide - CO2 (calc'd)
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_CO2)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_CO2)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_CO2)
'   Carbonate - CO3 (calc'd)
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_CO3)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_CO3)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_CO3)
'   Bromide -Br
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Br)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Br)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Br)
'   Chloride -Cl
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Cl)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Cl)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Cl)
'   Fluoride -F
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_F)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_F)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_F)
'   Sulfate -SO4
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_SO4)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_SO4)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_SO4)
'   Nitrate -NO3
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_NO3)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_NO3)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_NO3)
'   Phosphate - PO4 (as PO4)
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_PO4)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_PO4)
```

```
        S.Cells(CRow, Pcol).Value = .PermIons(SP_PO4)
'   Hydroxide -OH
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = 10 ^ -14 / HPlus(1)
        S.Cells(CRow, Ccol).Value = 10 ^ -14 / HPlus(2)
        S.Cells(CRow, Pcol).Value = 10 ^ -14 / HPlus(3)

'   Sum Total Anions
'       done on the sheet
'   Silica (as SiO2)
        CRow = CRow + 3
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_SiO2)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_SiO2)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_SiO2)
'   PH
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = -Log(HPlus(1)) / Log(10#)
        S.Cells(CRow, Ccol).Value = -Log(HPlus(2)) / Log(10#)
        S.Cells(CRow, Pcol).Value = -Log(HPlus(3)) / Log(10#)
'   TDS (in ppm)
        CRow = CRow + 1
'   Osmotic Pressure (estimated psi)
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = PPI(1)
        S.Cells(CRow, Ccol).Value = PPI(2)
        S.Cells(CRow, Pcol).Value = PPI(3)
'   Ionic Strength
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = Istr(1)
        S.Cells(CRow, Ccol).Value = Istr(2)
        S.Cells(CRow, Pcol).Value = Istr(3)
'   SI = Solubility Indices, expressed as percent of calculated solubility.
'   Langelier Saturation Index (LSI)
'   SI -Silica(SiO2)
'   SI - Calcium Sulfate (CaSO4)
'   SI - Strontium Sulfate (SrSO4)
'   SI - Barium Sulfate (BaSO4)
'   SI - Calcium Fluoride(CaF2)
'   SI - Analcime - NaAlSi2O6.H2O
'   SI - Gibbsite - Al(OH)3
'   SI - Am Al(OH)3 - Al(OH)3
'   SI - Fe(OH)3(am) - Fe(OH)3
'   SI - Goethite - FeO(OH)
'   SI - Siderite - FeCO3
'   SI - Vivianite - Fe3(PO4)2:8H2O
'   SI - Hydroxyapatite - Ca5(PO4)3OH

'   Metals/Others
'   Organics As TOC, row 55
        CRow = CRow + 17
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Organic)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Organic)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Organic)
'   Silver -Ag
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Ag)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Ag)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Ag)
'   Arsenic - As (as Arsenic)
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_As)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_As)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_As)
'   Gold -Au
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Au)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Au)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Au)
'   Cadmium -Cd
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Cd)
```

```
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Cd)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Cd)
'   Chromium - Cr (as Chromium)
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Cr)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Cr)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Cr)
'   Copper -Cu
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Cu)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Cu)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Cu)
'   Ferrous Iron - Fe++ (as Fe)
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Fe2)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Fe2)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Fe2)
'   Mercury -Hg
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Hg)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Hg)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Hg)
'   Manganese -Mn
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Mn)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Mn)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Mn)
'   Nickel -Ni
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Ni)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Ni)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Ni)
'   Lead -Pb
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Pb)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Pb)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Pb)
'   Selenium - Se (as Selenium)
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Se)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Se)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Se)
'   Tin -Sn
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Sn)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Sn)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Sn)
'   Titanium -Ti
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Ti)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Ti)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Ti)
'   Vanadium -V
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_V)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_V)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_V)
'   Zinc -Zn
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_Zn)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_Zn)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_Zn)
'   Sum heavy metals:
'   Borate - B4O7= row 71
        CRow = CRow + 2
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_B4O7)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_B4O7)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_B4O7)
'   Nitrite - NO2-
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_NO2)
```

```
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_NO2)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_NO2)
'   Sulfite - SO3=
        CRow = CRow + 1
        S.Cells(CRow, nCol).Value = .NetFeedIons(SP_SO3)
        S.Cells(CRow, Ccol).Value = .ConcIons(SP_SO3)
        S.Cells(CRow, Pcol).Value = .PermIons(SP_SO3)
'   Sum Other Anions:

    '   now do the total carbon for Conc and netfeed
    dummy = .NetFeedIons(SP_CO2) / 44.0099
    dummy = dummy + .NetFeedIons(SP_CO3) / 60.0092
    dummy = dummy + .NetFeedIons(SP_HCO3) / 61.0171
    Range("TC_NetFeed").Value = dummy

    dummy = .ConcIons(SP_CO2) / 44.0099
    dummy = dummy + .ConcIons(SP_CO3) / 60.0092
    dummy = dummy + .ConcIons(SP_HCO3) / 61.0171
    Range("TC_Conc").Value = dummy

    '   pressure values
    dummy = .FeedPress * 14.69
    Range("FeedPressure").Value = dummy



'       Done on the sheet
End With

PutDataOnSummary
Worksheets("System Design").Activate
Range("A1").Select
Range("FeedPressure").Select
Application.ScreenUpdating = True
End Sub
```

*Continue display*

```
Sub PutDataOnSummary()
Dim dummy As Double
Dim Dummy2 As Double
Dim n As Integer, i As Integer

Dim FlowUnit As Integer
Dim S As Worksheet
Dim CRow As Integer
Dim ColFeed As Integer, ColNet As Integer, ColConc As Integer, ColPerm As Integer
Dim ColBank(4) As Integer
'   NOW DO THE SUMMARY WORKSHEET

    '   these are the place holder names from the spreadsheet
    Set S = Worksheets("Summary")
    S.Activate

    ColFeed = S.Range("ColFeed").Column
    ColNet = S.Range("ColNet").Column
    ColConc = S.Range("ColConc").Column
    ColPerm = S.Range("ColPerm").Column

    ColBank(1) = S.Range("ColBank1").Column
    ColBank(2) = S.Range("ColBank2").Column
    ColBank(3) = S.Range("ColBank3").Column
    ColBank(4) = S.Range("ColBank4").Column

    '   the first row
    CRow = S.Range("ColFeed").Row
With S
    '   now the labels are shown commented out and 1 row is added for each
    '   this is mostly to maintain readability
    '   Chemistry Information
    CRow = CRow + 1
```

```
    'Flow rate - Units are gpm
        FlowUnit = Range("FlowUnit").Value
        dummy = FlowUser(CArray.FeedFlow, FlowUnit)
        S.Cells(CRow, ColFeed).Value = dummy
        dummy = FlowUser(CArray.RecycleFlow, FlowUnit) + dummy
        S.Cells(CRow, ColNet).Value = dummy
        dummy = FlowUser(CArray.ConcFlow, FlowUnit)
        S.Cells(CRow, ColConc).Value = dummy
        dummy = FlowUser(CArray.PermFlow, FlowUnit)
        S.Cells(CRow, ColPerm).Value = dummy

    '   TDS
        CRow = CRow + 1
'        Dummy = TDS(CArray.FeedIons())
'        S.Cells(Crow, ColFeed).Value = Dummy
'        Dummy = TDS(CArray.NetFeedIons())
'        S.Cells(Crow, ColNet).Value = Dummy
'        Dummy = TDS(CArray.ConcIons())
'        S.Cells(Crow, ColConc).Value = Dummy
'        Dummy = TDS(CArray.PermIons())
'        S.Cells(Crow, ColPerm).Value = Dummy
    '   PH
        CRow = CRow + 1

    '   Acid Required - ppm
        CRow = CRow + 1

    '   Acid Required - lb / Day
        CRow = CRow + 1

    '   Langelier Saturation Index (LSI)
        CRow = CRow + 1

    '   Silica (SiO2)
        CRow = CRow + 1

    '   Calcium Sulfate(CaSO4)
        CRow = CRow + 1

    '   Strontium Sulfate(SrSO4)
        CRow = CRow + 1

    '   Barium Sulfate(BaSO4)
        CRow = CRow + 1

    '   Percent - Calcium Fluoride (CaF2)
        CRow = CRow + 1

    '   SI -Analcime - NaAlSi2O6.H2O
        CRow = CRow + 1

    '   SI - Gibbsite - Al(OH)3
        CRow = CRow + 1

    '   SI - Am Al(OH)3 - Al(OH)3
        CRow = CRow + 1

    '   SI - Fe(OH)3(am) - Fe(OH)3
        CRow = CRow + 1

    '   SI -Goethite - FeO(OH)
        CRow = CRow + 1

    '   SI -Siderite - FeCO3
        CRow = CRow + 1

    '   SI - Vivianite - Fe3(PO4)2:8H2O
        CRow = CRow + 1

    '   SI - Hydroxyapatite - Ca5(PO4)3OH
        CRow = CRow + 1
```

```
    '   Blank Line
        CRow = CRow + 1
'   ==========================
    '   Estimated System Information
        CRow = CRow + 1

    '   Element/Membrane Type
        CRow = CRow + 1

    '   Number of Elements
        CRow = CRow + 1

    '   Number of Pressure Vessels
        CRow = CRow + 1

    '   Water Flux Rate (GFD - average)
        CRow = CRow + 1
        dummy = 0
        For n = 1 To 4
            If n <= CArray.nBanks Then
                dummy = dummy + (10.76 * CArray.Element.Area_M2 * CArray.Bank(n).nVessels)
            End If
        Next n
        dummy = FlowUser(CArray.PermFlow, FLOW_GPD) / dummy
        S.Cells(CRow, ColFeed).Value = dummy


    '   Water Recovery (%)
        CRow = CRow + 1

    '   Overall Salt Rejection
        CRow = CRow + 1

    '   Blank Line
        CRow = CRow + 1
'   ============================
    '   Estimated Bank by Bank Data
        CRow = CRow + 1

    '   Number of Elements
        CRow = CRow + 1

    '   Number of Pressure Vessels
        CRow = CRow + 1

    '   Water Flux Rate (GFD - average)
        CRow = CRow + 1
        '   gfd is in M^3/sec/(m^2)
        For n = 1 To 4
            If n <= CArray.nBanks Then
                dummy = FlowUser(CArray.Bank(n).PermFlow, FLOW_GPD) / (10.76 *
CArray.Element.Area_M2 * CArray.Bank(n).nVessels)
                S.Cells(CRow, ColBank(n)).Value = dummy
            Else
                S.Cells(CRow, ColBank(n)).Value = "N/A"
            End If
        Next n

    '   Concentration Polarization(Average)
        CRow = CRow + 1
        For n = 1 To 4
            If n <= CArray.nBanks Then
                dummy = CArray.Bank(n).ConcPolarization
                S.Cells(CRow, ColBank(n)).Value = dummy
            Else
                S.Cells(CRow, ColBank(n)).Value = "N/A"
            End If
        Next n

    '   Pressure at bank inlet - psi
```

```
        CRow = CRow + 1
        For n = 1 To 4
            If n <= CArray.nBanks Then
                dummy = ATMtoPRESS(CArray.Bank(n).FeedPress, FlowUnit)
                S.Cells(CRow, ColBank(n)).Value = dummy
            Else
                S.Cells(CRow, ColBank(n)).Value = "N/A"
            End If
        Next n

    '   Transmembrane (Net) Pressure - psi
        CRow = CRow + 1
        For n = 1 To 4
            If n <= CArray.nBanks Then
                dummy = ATMtoPRESS(CArray.Bank(n).PTransMemb, FlowUnit)
                S.Cells(CRow, ColBank(n)).Value = dummy
            Else
                S.Cells(CRow, ColBank(n)).Value = "N/A"
            End If
        Next n

    '   Bank Differential pressure Loss - psi
        CRow = CRow + 1
        For n = 1 To 4
            If n <= CArray.nBanks Then
                dummy = ATMtoPRESS((CArray.Bank(n).FeedPress - CArray.Bank(n).ConcPress),
FlowUnit)
                S.Cells(CRow, ColBank(n)).Value = dummy
            Else
                S.Cells(CRow, ColBank(n)).Value = "N/A"
            End If
        Next n

    '   Flow Rate to First Element in Bank
        CRow = CRow + 1
        For n = 1 To 4
            If n <= CArray.nBanks Then
                dummy = FlowUser(CArray.Bank(n).FeedFlow, FlowUnit) / CArray.Bank(n).nVessels
                S.Cells(CRow, ColBank(n)).Value = dummy
            Else
                S.Cells(CRow, ColBank(n)).Value = "N/A"
            End If
        Next n

    '   Flow Rate to Last Element in Bank
        CRow = CRow + 1
        For n = 1 To 4
            If n <= CArray.nBanks Then
                dummy = FlowUser(CArray.Bank(n).ConcFlow, FlowUnit) / CArray.Bank(n).nVessels
                S.Cells(CRow, ColBank(n)).Value = dummy
            Else
                S.Cells(CRow, ColBank(n)).Value = "N/A"
            End If
        Next n

    '   Permeate Flow
        CRow = CRow + 1
        For n = 1 To 4
            If n <= CArray.nBanks Then
                dummy = FlowUser(CArray.Bank(n).PermFlow, FlowUnit)
                S.Cells(CRow, ColBank(n)).Value = dummy
            Else
                S.Cells(CRow, ColBank(n)).Value = "N/A"
            End If
        Next n

    '   Water Recovery (%)
        CRow = CRow + 1
        For n = 1 To 4
            If n <= CArray.nBanks Then
                dummy = CArray.Bank(n).PermFlow / CArray.Bank(n).FeedFlow
```

```
                S.Cells(CRow, ColBank(n)).Value = dummy
            Else
                S.Cells(CRow, ColBank(n)).Value = "N/A"
            End If
        Next n


    '   Feed TDS
        CRow = CRow + 1
        For n = 1 To 4
            If n <= CArray.nBanks Then
                dummy = TDSfromMEQ(CArray.Bank(n).FeedMeq())
                S.Cells(CRow, ColBank(n)).Value = dummy
            Else
                S.Cells(CRow, ColBank(n)).Value = "N/A"
            End If
        Next n

    '   Concentrate TDS
        CRow = CRow + 1
        For n = 1 To 4
            If n <= CArray.nBanks Then
                dummy = TDSfromMEQ(CArray.Bank(n).ConcMeq())
                S.Cells(CRow, ColBank(n)).Value = dummy
            Else
                S.Cells(CRow, ColBank(n)).Value = "N/A"
            End If
        Next n

    '   Permeate TDS
        CRow = CRow + 1
        For n = 1 To 4
            If n <= CArray.nBanks Then
                dummy = TDSfromMEQ(CArray.Bank(n).PermMeq())
                S.Cells(CRow, ColBank(n)).Value = dummy
            Else
                S.Cells(CRow, ColBank(n)).Value = "N/A"
            End If
        Next n

    '   Blank Line
        CRow = CRow + 1

    '   Nominal System Parameters*
        CRow = CRow + 1

    '   Water Flux Rate (GFD - average)
        CRow = CRow + 1

    '   Concentration Polarization(Average)
        CRow = CRow + 1

    '   Applied Pressure (average psi)
        CRow = CRow + 1

    '   Flow Rate to First Element in Bank
        CRow = CRow + 1

    '   Flow Rate to Last Element in Bank
        CRow = CRow + 1

    '   Blank Line
        CRow = CRow + 1

End With
```

*The Guess routine.*

```
End Sub
'   ======================First Guess ======================
'   Area As Double      -   Total System Area
```

```
'   A As Double         -   Effective (calculated A value)
'   B As Double         -
'   FeedFlow as double  -   The inlet flow in m^3/sec
'   ConcFlow as double  -   The conc flow in m^3/sec
'   AvgFlow As Double   -
'   Beta As Double      -   CP Guess as 1.1
'
Function FirstGuess() As Double
Dim n As Integer, i As Integer
Dim Area As Double
Dim A As Double
Dim B As Double
Dim Beta As Double
Dim dp As Double
Dim dpBank(4) As Double
Dim dpCoef As Double
Dim PI As Double
Dim CF As Double
Dim floatDummy As Double
'   FeedMeq(last_ION) is a publis so that next calc routine can use it as well
'   Dim FeedMeq(LAST_ION) As Double
Dim TDS As Double
Dim SaltPass As Double
Dim PermTDS As Double
Dim ConcTDS As Double
Dim Pressure As Double
Dim nTotalVessels As Integer

'   first get the whole system area in m^2
Area = 0
For n = 1 To CArray.nBanks
    Area = Area + CArray.Bank(n).nVessels * MyElement.Area_M2
    nTotalVessels = nTotalVessels + CArray.Bank(n).nVessels
Next n


'   Analysis, this also gets the analysis into the FeedMeq() variable
ConvertPPMtoMEQ FeedMeq(), CArray.FeedIons()

TDS = TDSfromMEQ(FeedMeq())
PI = OsmoticPressureMEQ(FeedMeq(), Temperature)                    ' 1 meq/l NaCl is 58.5 mg/l,
about 0.5 psi Pi = .5/14.69 = 0.035


'   dp, dpcoef is in form of atm = dpcoef * elementLen in inches * (flow in M^3/sec) ^ 1.5
'   MyElement has already been scaled for element length * number of elements in a vessel.
'   eg. eight inch elements, ElementLen = 40 inch, 6 elements/vessel, then dpcoef is 6 * 40 *
range("dp80_Inch").value
dpCoef = MyElement.dpCoef

'   do a full system dp and flow calc
'   see if we have recycle
RecycleFlow = CArray.RecycleFlow
'   need the feed flow for the system divided by number of vessels in fisrt bank, flow to 1
vessel!
FeedFlow = (CArray.FeedFlow + RecycleFlow) / CArray.Bank(1).nVessels
'   assume the perm flow is evenly divided by number of vessels (not entirely correct but OK for
the first cut)
PermFlow = CArray.PermFlow / nTotalVessels
dp = 0
For n = 1 To CArray.nBanks
    '   the concflow includes the recycle flows
    '   overall concflow is less the recycleflow
    ConcFlow = FeedFlow - PermFlow
    dpBank(n) = DeltaPressure(FeedFlow, ConcFlow, dpCoef)
    dp = dp + dpBank(n)
    '   use this loop to assign flows to banks
    '   assign current bank flows, this will include the recycle if any
    CArray.Bank(n).FeedFlow = FeedFlow * CArray.Bank(n).nVessels
    CArray.Bank(n).PermFlow = PermFlow * CArray.Bank(n).nVessels
    CArray.Bank(n).ConcFlow = ConcFlow * CArray.Bank(n).nVessels
```

```
    If CArray.nBanks >= n + 1 Then
        '   assign next bank flows
        CArray.Bank(n + 1).FeedFlow = ConcFlow * CArray.Bank(n + 1).nVessels
        CArray.Bank(n + 1).PermFlow = PermFlow * CArray.Bank(n + 1).nVessels
        CArray.Bank(n + 1).ConcFlow = CArray.Bank(n + 1).FeedFlow - CArray.Bank(n + 1).PermFlow
        '   Now setup for the next iteration
        '   the new feedflow is the old concflow * number of vessels in current bank divided by
number in next bank
        FeedFlow = ConcFlow * CArray.Bank(n).nVessels / CArray.Bank(n + 1).nVessels
        '   The new permflow is already done, it is the same for each bank
    End If
Next n

'   now find the first pressure by using total area, permflow, and dp
B = MyElement.BValue
A = MyElement.Avalue
Pressure = CArray.PermFlow / (Area * A)
Pressure = Pressure + dp / 2 + PI

'   now assign/partition pressure and flow to the first bank so we have some reasonable starting
values
CArray.Bank(1).FeedPress = Pressure
For n = 1 To CArray.nBanks
    CArray.Bank(n).ConcPress = CArray.Bank(n).FeedPress - dpBank(n)
    If CArray.nBanks >= n + 1 Then
        CArray.Bank(n + 1).FeedPress = CArray.Bank(n).ConcPress
    End If
Next n

'   get first guess salt values -
'   the recycle concentration and flow is ignored here
'   first redo ourlocal variables
FeedFlow = CArray.FeedFlow
PermFlow = CArray.PermFlow
ConcFlow = CArray.ConcFlow
SaltPass = B / (Pressure * A + B)
For i = 1 To LAST_ION
    PermMeq(i) = SaltPass * B_Ions(i) * FeedMeq(i)
    ConcMeq(i) = (FeedMeq(i) * FeedFlow - PermMeq(i) * PermFlow) / ConcFlow
Next i
'   now fix up CO2
    PermMeq(SP_CO2) = FeedMeq(SP_CO2)
    ConcMeq(SP_CO2) = FeedMeq(SP_CO2)

'   now redo but include the recycle concentration = the concentrate concentration
'   GetNetFeed(ByRef NetIons() As Double, ByRef NewFlow As Double, _
    FeedIons() As Double, FlowFeed As Double, _
    RecycleIons() As Double, FlowRecycle As Double)
'   NB NewFlow is now Recycle + Feed but not used in FirstGuess
GetNetFeed NetFeedMeq(), floatDummy, FeedMeq(), CArray.FeedFlow, ConcMeq(), CArray.RecycleFlow
'   redo just for the first bank
For i = 1 To LAST_ION
    PermMeq(i) = SaltPass * B_Ions(i) * NetFeedMeq(i)
    CArray.Bank(1).ConcMeq(i) = (NetFeedMeq(i) * FeedFlow - PermMeq(i) * CArray.Bank(1).PermFlow)
/ CArray.Bank(1).ConcFlow
    '   get the values into the first bank FeedMeq()
    CArray.Bank(1).FeedMeq(i) = NetFeedMeq(i)
    CArray.Bank(1).PermMeq(i) = PermMeq(i)
Next i

'   now we data in bank 1 as needed
'   Next - need to estimate the data for the remaining banks
If CArray.nBanks = 1 Then
    FirstGuess = Pressure
    Exit Function
End If

For n = 2 To CArray.nBanks
    With CArray.Bank(n)
'        .FeedFlow = CArray.Bank(n - 1).ConcFlow
```

```
'        .PermFlow = CArray.PermFlow * .nVessels / nTotalVessels
'        .ConcFlow = .FeedFlow - PermFlow
        For i = 1 To LAST_ION
            .FeedMeq(i) = CArray.Bank(n - 1).ConcMeq(i)
            .PermMeq(i) = CArray.Bank(n - 1).PermMeq(i)
            .ConcMeq(i) = (.FeedFlow * .FeedMeq(i) - .PermFlow * .PermMeq(i)) / .ConcFlow
        Next i
    End With
Next n

If boolDebug Then
    PermTDS = TDSfromMEQ(PermMeq())
    ConcTDS = TDSfromMEQ(ConcMeq())
    Debug.Print "area", Area, Area * 10.76
    Debug.Print "dp", dp, dp * 14.69
    Debug.Print "feedflow", FeedFlow, FeedFlow * 60 * 264.2
    Debug.Print "permflow", PermFlow, PermFlow * 60 * 264.2
    Debug.Print "Pi", PI, PI * 14.69
    Debug.Print "A", A
    Debug.Print "TempCorr", ATempCorr
    Debug.Print "temperature", Temperature
    Debug.Print "pressure", Pressure, Pressure * 14.69
    Debug.Print "TDS - Perm, Conc", PermTDS, ConcTDS
End If
FirstGuess = Pressure

End Function

Sub GetSystemData()
Dim n As Integer

'   This sub collects the input data about the system to start the calculations
'   It needs to run AFTER the GetElementData to fill in all of the variables
    With CArray
        .Element = MyElement
        .nBanks = Range("NumberBanks").Value
        .Bank(1).nVessels = Range("Bank1").Value
        .Bank(2).nVessels = Range("Bank2").Value
        .Bank(3).nVessels = Range("Bank3").Value
        .Bank(4).nVessels = Range("Bank4").Value

    '   Flows in system Units
        .FeedFlow = Range("FeedFlow").Value
        .PermFlow = Range("ProductFlow").Value
        .ConcFlow = Range("ConcentrateFlow").Value
        .RecycleFlow = Range("RecycleFlow").Value
    'convert to m^/sec
        n = Range("FlowUnit").Value
        .FeedFlow = FlowM3SEC(.FeedFlow, n)
        .PermFlow = FlowM3SEC(.PermFlow, n)
        .ConcFlow = FlowM3SEC(.ConcFlow, n)
        .RecycleFlow = FlowM3SEC(.RecycleFlow, n)

        '    pressures not yet known, set to 10 for starting point
        '    Press in atm
        .FeedPress = 10
        .ConcPress = 5

    '   feed water
       GetTreatedIons .FeedIons()
    End With


End Sub
```

*A data collection sub*

```
Sub GetElementData()
'   This sub takes the generic element data and bank data and creates a new master element
'   The master element is in equivalent length of the whole vessl. IE if the vessel contains 6
```

```
'   each, 40 inch elements, then the effective length is 240 inches , and the total area is 6 *
400 = 2400 ft^2.
'   240 inches is used for the dp calculation.  The relevant data is contained in the named
ranges
'       DP25Inch (2.5 inch elements)
'       DP40Inch (4 inch elements)
'       DP80Inch (8 inch elements)
'   The constant is multiplied by the effective length to calculate a DP value.
'   See Function DeltaPressure(Flow, dpCoef) as double
'
Dim n As Integer
Dim nRow As Long
Dim nCol As Long
Dim typElem As BigElementType
Dim typMemb As Membrane
Dim nElem As Integer '(elements per vessel)
Dim M3Sec As Double  ' flow in m^3/sec
Dim ATM As Double ' press in atm
Dim saltFlow As Double  'kg/sec of salt
Dim BValue As Double
Dim strType As String

    Temperature = Range("Temp_C").Value
    n = Worksheets("System Design").lstSelectMembrane.ListIndex + 1
    ATempCorr = Range("tempcorr").Offset(0, n).Value
    If ATempCorr = 0 Then ATempCorr = 2900
    ATempCorr = Exp(ATempCorr * (1 / (273 + Temperature) - 1 / 298))
    '   Ph - this can change if there is recycle in place
    FeedpH = Range("Input_ph_Treated").Value


'   get number of elements/vessel
nElem = Range("elementspervessel").Value


With typMemb
    .Description = Range("Description").Offset(-1, 0).Value
    .Diameter = Range("diameter").Offset(-1, 0).Value
    .Length = Range("Length").Offset(-1, 0).Value
    .Type = Range("Type").Offset(-1, 0).Value
    .NaCl_rejection = Range("NaCl_Rejection").Offset(-1, 0).Value
    .MgSO4_rejection = Range("MgSO4_Rejection").Offset(-1, 0).Value
    .TestPressure = Range("Test_Pressure").Offset(-1, 0).Value
    .TestTDS = Range("Test_Tds").Offset(-1, 0).Value
    .TestFlow = Range("Test_Flow_Rate").Offset(-1, 0).Value
    .TestTemperature = Range("Test_Temperature").Offset(-1, 0).Value
    .Area = Range("Elem_Area").Offset(-1, 0).Value

    '   now the calculated values
    ATM = .TestPressure - .TestTDS * 0.01 'estimate osmotic pressure of test solution = 2000 tds
= 20 psi osmotic pressure
    ATM = ATM / 14.69   ' net pressure in atm
    M3Sec = .TestFlow / 22826880# '86400 * 264.2 GPD to M^3/sec

    If .NaCl_rejection > 0 Then
        saltFlow = .TestTDS * (1 - .NaCl_rejection / 100) '   perm TDS in mg/l
        saltFlow = saltFlow / 1000 * M3Sec ' flow in kg/sec - mg/l * 1g/1000mg * 1 kg/1000g *
1000L/1M^3 * m^3/sec = 1kg/sec
    ElseIf .MgSO4_rejection > 0 Then   'using MgSO4
        saltFlow = .TestTDS * (1 - .MgSO4_rejection / 100) '   perm TDS in mg/l
        saltFlow = saltFlow / 1000 * M3Sec ' flow in kg/sec - mg/l * 1g/1000mg * 1 kg/1000g *
1000L/1M^3 * m^3/sec = 1kg/sec
    Else
        '   use default B value
        saltFlow = 0.0000052569
            'eg 6000 gpd = 4.38^10-8 * 6000 = 2.628*10^-4 M^3/sec
            'rejection = 99%, TestTDS = 2000
            '2000 * (1-99/100) / 1000 * 2.628e-4 = 5.25696 e-6
    End If
End With
```

```
'now create the big element
With typElem
    .Area_M2 = nElem * typMemb.Area / 10.76
    .Avalue = nElem * M3Sec / .Area_M2 / ATM '(m^3/sec)/(m^2*atm) - NB must divide by atm actual
value to get PER atm
    '   now correct for temperature
    '   Atempcorr is less than one for temp below 25 C so Avalue must be divided
    .Avalue = .Avalue / ATempCorr

    .BValue = nElem * saltFlow / .Area_M2   '   (kg/sec)/m^2 (actually the units are
M^3/Sec/(m^2)) from original data
                                            '   Units cancel to m/sec for B, and m/(sec*atm) for
A
                                            '   so that salt pass = dC * B/(A*Pnet + B) where dC
= Bulk Conc * Beta - Perm Conc
                                            '   all done by ion
    '.BValue = .BValue * BTempCorr

    '   betacoeff is used for concentration polarization over the element
    '   assumption is ConcPolarization ~= exp(BetaCoef * Recovery),
    '   eg, 50 % recovery in a 6 element system, CP about 1.1 max, Length = 40 *6 =240
    '   -.3492 * log(240) + 2.0152 = -1.54
    '   CP = exp(-1.54 * .5) = .46
    .BetaCoef = -0.3492 * Log(typMemb.Length * nElem) + 2.0152

    Select Case typMemb.Diameter
        Case 2.5
            .dpCoef = Range("DP25_Inch").Value
        Case 4
            .dpCoef = Range("DP40_Inch").Value
        Case 8
            .dpCoef = Range("DP80_Inch").Value
        Case Else
            .dpCoef = 5266.1 * typMemb.Diameter ^ (-2.6791)  ' see DPCoef vs Diameter.xls for
curve fit
    End Select
    '   now multiply by number of elements per vessel and length of each element
    .dpCoef = .dpCoef * nElem * typMemb.Length

    ' now get the B_Ion data from Membrane Data Sheet
    '   first find where the data is sitting
        strType = "Type" & Format(typMemb.Type, "#")

        nCol = Worksheets("Membrane Data").Range(strType).Column
        nRow = Worksheets("Membrane Data").Range("Beta_Na").Row
        For n = 1 To LAST_ION
            .B_Ion(n) = Worksheets("Membrane Data").Cells(nRow + n - 1, nCol).Value
            '   the global variable - set up for the first pass
            B_Ions(n) = .B_Ion(n)
        Next n
End With
MyElement = typElem
MyMembrane = typMemb

End Sub
```

*The solubility runs using the Solver*

```
Sub RunSolver(Optional nSections As Integer)
Dim n As Integer
Dim nLoops As Integer
Dim boolTreatedDone As Boolean
Dim nbtnWidth As Double


'   boolean to bypass the NetFeed calc if TreatedFeed is calculated AND RecycleFlow = 0
'   NetFeed is then the same as the treated Feed so no need to calculate twice
boolTreatedDone = False

If IsMissing(nSections) Then
    nSections = 0
```

```
End If
    DoEvents
    If boolAbortScaleCalcs = True Then
        boolAbortScaleCalcs = False
        If MsgBox("Abort Scaling Calcs?", vbOKCancel) = vbOK Then
            GoTo OuttaHere
        End If
    End If
```

*We have found that if the analysis changes dramatically – ie from 0 to some value, two passes of theSolver get a better convergence.*

```
nLoops = 1
'   nLoops needs to be set at 2 if any of the multipliers are set at 0

If Range("$H$44").Value = 0 And Range("Treated_PO4").Value > 0 Then   ' check PO4 first
    nLoops = 2
ElseIf Range("$N$38").Value = 0 And Range("Treated_Al").Value > 0 Then   '   Al
    nLoops = 2
ElseIf Range("$N$39").Value = 0 And Range("Treated_FE2").Value > 0 Then   '   Fe2
    nLoops = 2
ElseIf Range("$N$40").Value = 0 And Range("Treated_FE3").Value > 0 Then    '   Fe3
    nLoops = 2
ElseIf Range("$N$34").Value = 0 And Range("Treated_Na").Value > 0 Then    '   Na
    nLoops = 2
ElseIf Range("$N$35").Value = 0 And Range("Treated_K").Value > 0 Then
    nLoops = 2
ElseIf Range("$N$36").Value = 0 And Range("Ca_Treated").Value > 0 Then
    nLoops = 2
ElseIf Range("$N$37").Value = 0 And Range("Treated_Mg").Value > 0 Then
    nLoops = 2
ElseIf Range("$C$44").Value = 0 And Range("Treated_Cl").Value > 0 Then
    nLoops = 2
ElseIf Range("$D$44").Value = 0 And Range("Treated_SO4").Value > 0 Then
    nLoops = 2
ElseIf Range("$F$44").Value = 0 And Range("Treated_HCO3").Value > 0 Then
    nLoops = 2
End If


'On Error Resume Next
Worksheets("Calc").Activate
```

*Setup the progress button*

```
    nbtnWidth = Worksheets("Calc").btnRunSolver.Width
    Worksheets("Calc").btnRunSolver.Caption = "Setting Solver Config"
    Worksheets("Calc").btnRunSolver.BackColor = &HFF& 'This is Red
    Worksheets("Calc").btnRunSolver.Width = nbtnWidth / 2
    SolverOk SetCell:="$M$34", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$34"
    SolverOptions MaxTime:=100, Iterations:=10000, Precision:=0.00000001, _
        AssumeLinear:=True, StepThru:=False, Estimates:=1, Derivatives:=1, SearchOption _
        :=1, IntTolerance:=0.001, Scaling:=True, Convergence:=0.001, AssumeNonNeg:=True

    '   Need to make sure old criteria are deleted
        SolverDelete CellRef:="$M$34", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M$35", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M$36", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M$37", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M$38", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M$39", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M$40", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$D$43", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$F$43", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$H$43", Relation:=3, FormulaText:="0"
        DoEvents
        '   $N$38 keeps appearing so get rid of it just in case.
        SolverDelete CellRef:="$N#68", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M#64", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M#65", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M#66", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M#67", Relation:=3, FormulaText:="0"
```

```
        SolverDelete CellRef:="$M#68", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M#69", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M$70", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$D$73", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$F$73", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$H$73", Relation:=3, FormulaText:="0"
        DoEvents
        SolverDelete CellRef:="$M$94", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M$95", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M$96", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M$97", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M$98", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M$99", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M$100", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$D$103", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$F$103", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$H$103", Relation:=3, FormulaText:="0"
        DoEvents

Worksheets("Calc").btnRunSolver.Caption = "In Solution Phase"
Select Case nSections
    Case 0, 1
        GoTo SectionTreatedFeed
    Case 2
        boolTreatedDone = False
        GoTo SectionNetFeed
    Case 3
        GoTo SectionConcentrate
End Select


SectionTreatedFeed:

Worksheets("Calc").btnRunSolver.Width = nbtnWidth / 1.8
Worksheets("Calc").btnRunSolver.Caption = "Solving Feed"
For n = 1 To nLoops  ' loops if necessary
    DoEvents
    If boolAbortScaleCalcs = True Then
        boolAbortScaleCalcs = False
        If MsgBox("Abort Scaling Calcs?", vbOKCancel) = vbOK Then
            GoTo OuttaHere
        End If
    End If

    '   Na
    Worksheets("Calc").btnSolverTreated.Caption = "Solving Na - Treated Feed"


    If Range("$L$34").Value <= 0 Then
        Range("$N$34").Value = 0
    Else
```

*This is the prototype Solver setup call*

```
        SolverOk SetCell:="$M$34", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$34"
        SolverAdd CellRef:="$M$34", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$M$34", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$34"
        '   This UserFinish:=True causes the last section to be bypassed
        '   This is an undocumented feature of the Excel Solver
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$M$34", Relation:=3, FormulaText:="0"
    End If

    '   K
    If Range("$L$35").Value <= 0 Then
        Range("$N$35").Value = 0
    Else
        Worksheets("Calc").btnSolverTreated.Caption = "Solving K - Treated Feed"
        SolverOk SetCell:="$M$35", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$35"
        SolverAdd CellRef:="$M$35", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$M$35", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$35"
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$M$35", Relation:=3, FormulaText:="0"
```

```
    End If

    DoEvents

        '   Ca
    If Range("$L$36").Value <= 0 Then
        Range("$N$36").Value = 0
    Else
        Worksheets("Calc").btnSolverTreated.Caption = "Solving Ca - Treated Feed"
        SolverOk SetCell:="$M$36", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$36"
        SolverAdd CellRef:="$M$36", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$M$36", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$36"
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$M$36", Relation:=3, FormulaText:="0"
    End If

        '   Mg
    If Range("$L$37").Value <= 0 Then
        Range("$N$37").Value = 0
    Else
        Worksheets("Calc").btnSolverTreated.Caption = "Solving Mg - Treated Feed"
        SolverOk SetCell:="$M$37", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$37"
        SolverAdd CellRef:="$M$37", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$M$37", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$37"
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$M$37", Relation:=3, FormulaText:="0"
    End If

        '   Al
    If Range("$L$38").Value <= 0 Then
        Range("$N$38").Value = 0
    Else
        Worksheets("Calc").btnSolverTreated.Caption = "Solving Al - Treated Feed"
        SolverOk SetCell:="$M$38", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$38"
        'SolverDelete CellRef:="$M$37", Relation:=3, FormulaText:="0"
        SolverAdd CellRef:="$M$38", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$M$38", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$38"
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$M$38", Relation:=3, FormulaText:="0"
    End If

        '   Fe2
    If Range("$L$39").Value <= 0 Then
        Range("$N$39").Value = 0
    Else
        Worksheets("Calc").btnSolverTreated.Caption = "Solving Fe2 - Treated Feed"
        SolverOk SetCell:="$M$39", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$39"
        SolverAdd CellRef:="$M$39", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$M$39", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$39"
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$M$39", Relation:=3, FormulaText:="0"
    End If

        '   Fe3
    If Range("$L$40").Value <= 0 Then
        Range("$N$40").Value = 0
    Else
        Worksheets("Calc").btnSolverTreated.Caption = "Solving Fe3 - Treated Feed"
        SolverOk SetCell:="$M$40", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$40"
        SolverAdd CellRef:="$M$40", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$M$40", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$40"
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$M$40", Relation:=3, FormulaText:="0"
    End If

    DoEvents

'   Now Anions
    '   Cl
    If Range("$C$42").Value <= 0 Then
        Range("$C$44").Value = 0
```

```
    Else
        Worksheets("Calc").btnSolverTreated.Caption = "Solving Cl - Treated Feed"
        SolverOk SetCell:="$C$43", MaxMinVal:=2, ValueOf:="0", ByChange:="$C$44"
        SolverAdd CellRef:="$C$43", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$C$43", MaxMinVal:=2, ValueOf:="0", ByChange:="$C$44"
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$C$43", Relation:=3, FormulaText:="0"
    End If

    '   SO4
    If Range("$D$42").Value <= 0 Then
        Range("$D$44").Value = 0
    Else
        Worksheets("Calc").btnSolverTreated.Caption = "Solving SO4 - Treated Feed"
        SolverOk SetCell:="$D$43", MaxMinVal:=2, ValueOf:="0", ByChange:="$D$44"
        SolverAdd CellRef:="$D$43", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$D$43", MaxMinVal:=2, ValueOf:="0", ByChange:="$D$44"
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$D$43", Relation:=3, FormulaText:="0"
    End If

    '   HCO3
    If Range("$F$42").Value <= 0 Then
        Range("$F$44").Value = 0
    Else
        Worksheets("Calc").btnSolverTreated.Caption = "Solving HCO3 - Treated Feed"
        SolverOk SetCell:="$F$43", MaxMinVal:=2, ValueOf:="0", ByChange:="$F$44"
        SolverAdd CellRef:="$F$43", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$F$43", MaxMinVal:=2, ValueOf:="0", ByChange:="$F$44"
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$F$43", Relation:=3, FormulaText:="0"
    End If

    DoEvents

    '   HPO4
    If Range("$H$42").Value <= 0 Then
        Range("$H$44").Value = 0
    Else
        Worksheets("Calc").btnSolverTreated.Caption = "Solving HPO4 - Treated Feed"
        SolverOk SetCell:="$H$43", MaxMinVal:=2, ValueOf:="0", ByChange:="$H$44"
        SolverAdd CellRef:="$H$43", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$H$43", MaxMinVal:=2, ValueOf:="0", ByChange:="$H$44"
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$H$43", Relation:=3, FormulaText:="0"
    End If

    '   H2PO4
    If Range("$I$42").Value <= 0 Then
        Range("$I$44").Value = 0
    Else
        Worksheets("Calc").btnSolverTreated.Caption = "Solving H2PO4 - Treated Feed"
        SolverOk SetCell:="$I$43", MaxMinVal:=2, ValueOf:="0", ByChange:="$I$44"
        SolverAdd CellRef:="$I$43", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$I$43", MaxMinVal:=2, ValueOf:="0", ByChange:="$I$44"
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$I$43", Relation:=3, FormulaText:="0"
    End If
    Worksheets("Calc").btnSolverTreated.Caption = "Treated Feed - Complete Pass " & Format(n,
"#") & " of " & Format(nLoops, "#")
    boolTreatedDone = True
Next n
If nSections <> 0 Then GoTo OuttaHere

SectionNetFeed:


'   Now do the NET FEED
Worksheets("Calc").btnRunSolver.Width = nbtnWidth * 0.75
Worksheets("Calc").btnRunSolver.Caption = "Solving Net Feed"
```

```
If boolTreatedDone = True And Range("RecycleFlow").Value <= 0 Then
    '   Treated Feed is the same as the recycle. just copy over the mulitpliers
    Worksheets("Calc").Range("$N$64").Value = Worksheets("calc").Range("$N$34").Value
    Worksheets("Calc").Range("$N$65").Value = Worksheets("calc").Range("$N$35").Value
    Worksheets("Calc").Range("$N$66").Value = Worksheets("calc").Range("$N$36").Value
    Worksheets("Calc").Range("$N$67").Value = Worksheets("calc").Range("$N$37").Value
    Worksheets("Calc").Range("$N$68").Value = Worksheets("calc").Range("$N$38").Value
    Worksheets("Calc").Range("$N$69").Value = Worksheets("calc").Range("$N$39").Value
    Worksheets("Calc").Range("$N$70").Value = Worksheets("calc").Range("$N$40").Value
    '   Anions
    Worksheets("Calc").Range("$C$74").Value = Worksheets("calc").Range("$C$44").Value
    Worksheets("Calc").Range("$D$74").Value = Worksheets("calc").Range("$D$44").Value
    Worksheets("Calc").Range("$F$74").Value = Worksheets("calc").Range("$F$44").Value
    Worksheets("Calc").Range("$H$74").Value = Worksheets("calc").Range("$H$44").Value
    Worksheets("Calc").Range("$I$74").Value = Worksheets("calc").Range("$I$44").Value

    Worksheets("Calc").btnSolverNet.Caption = "Net Feed - Complete"
Else
    For n = 1 To nLoops  ' loops if necessary
        DoEvents
        If boolAbortScaleCalcs = True Then
            boolAbortScaleCalcs = False
            If MsgBox("Abort Scaling Calcs?", vbOKCancel) = vbOK Then
                GoTo OuttaHere
            End If
        End If

        Worksheets("Calc").btnSolverNet.Caption = "Solving Na - Net Feed"
        SolverOk SetCell:="$M$64", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$64"
        SolverOptions MaxTime:=100, Iterations:=10000, Precision:=0.00000001, _
            AssumeLinear:=True, StepThru:=False, Estimates:=1, Derivatives:=1, SearchOption _
            :=1, IntTolerance:=0.001, Scaling:=True, Convergence:=0.001, AssumeNonNeg:=True

        '   Need to make sure old criteria are deleted
            SolverDelete CellRef:="$H$43", Relation:=3, FormulaText:="0"
            SolverDelete CellRef:="$M$34", Relation:=3, FormulaText:="0"
            SolverDelete CellRef:="$M$35", Relation:=3, FormulaText:="0"
            SolverDelete CellRef:="$M$36", Relation:=3, FormulaText:="0"
            '   $N$68 keeps appearing so get rid of it just in case.
            SolverDelete CellRef:="$N$68", Relation:=3, FormulaText:="0"
            SolverDelete CellRef:="$H$73", Relation:=3, FormulaText:="0"
            SolverDelete CellRef:="$M$64", Relation:=3, FormulaText:="0"
            SolverDelete CellRef:="$M$65", Relation:=3, FormulaText:="0"
            SolverDelete CellRef:="$M$66", Relation:=3, FormulaText:="0"
            SolverDelete CellRef:="$N$68", Relation:=3, FormulaText:="0"
            SolverDelete CellRef:="$H$103", Relation:=3, FormulaText:="0"
            SolverDelete CellRef:="$M$94", Relation:=3, FormulaText:="0"
            SolverDelete CellRef:="$M$95", Relation:=3, FormulaText:="0"
            SolverDelete CellRef:="$M$96", Relation:=3, FormulaText:="0"
            SolverDelete CellRef:="$N$98", Relation:=3, FormulaText:="0"

            DoEvents


        If Range("$L$64").Value <= 0 Then
            Range("$N$64").Value = 0
        Else
            SolverAdd CellRef:="$M$64", Relation:=3, FormulaText:="0"
            SolverOk SetCell:="$M$64", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$64"
            '   This UserFinish:=True causes the last section to be bypassed
            '   This is an undocumented feature of the Excel Solver
            Solversolve UserFinish:=True
            SolverDelete CellRef:="$M$64", Relation:=3, FormulaText:="0"
        End If

        '   K
        If Range("$L$65").Value <= 0 Then
            Range("$N$65").Value = 0
        Else
            Worksheets("Calc").btnSolverNet.Caption = "Solving K - Net Feed"
            SolverOk SetCell:="$M$65", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$65"
```

```
    SolverAdd CellRef:="$M$65", Relation:=3, FormulaText:="0"
    SolverOk SetCell:="$M$65", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$65"
    Solversolve UserFinish:=True
    SolverDelete CellRef:="$M$65", Relation:=3, FormulaText:="0"
End If

    '   Ca
If Range("$L$66").Value <= 0 Then
    Range("$N$66").Value = 0
Else
    Worksheets("Calc").btnSolverNet.Caption = "Solving Ca - Net Feed"
    SolverOk SetCell:="$M$66", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$66"
    SolverAdd CellRef:="$M$66", Relation:=3, FormulaText:="0"
    SolverOk SetCell:="$M$66", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$66"
    Solversolve UserFinish:=True
    SolverDelete CellRef:="$M$66", Relation:=3, FormulaText:="0"
End If

DoEvents

    '   Mg
 If Range("$L$67").Value <= 0 Then
    Range("$N$67").Value = 0
Else
    Worksheets("Calc").btnSolverNet.Caption = "Solving Mg - Net Feed"
    SolverOk SetCell:="$M$67", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$67"
    SolverAdd CellRef:="$M$67", Relation:=3, FormulaText:="0"
    SolverOk SetCell:="$M$67", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$67"
    Solversolve UserFinish:=True
    SolverDelete CellRef:="$M$67", Relation:=3, FormulaText:="0"
End If

    '   Al
If Range("$L$68").Value <= 0 Then
    Range("$N$68").Value = 0
Else
    Worksheets("Calc").btnSolverNet.Caption = "Solving Al - Net Feed"
    SolverOk SetCell:="$M$68", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$68"
    'SolverDelete CellRef:="$M$67", Relation:=3, FormulaText:="0"
    SolverAdd CellRef:="$M$68", Relation:=3, FormulaText:="0"
    SolverOk SetCell:="$M$68", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$68"
    Solversolve UserFinish:=True
    SolverDelete CellRef:="$M$68", Relation:=3, FormulaText:="0"
End If

    '   Fe2
If Range("$L$69").Value <= 0 Then
    Range("$N$69").Value = 0
Else
    Worksheets("Calc").btnSolverNet.Caption = "Solving Fe2 - Net Feed"
    SolverOk SetCell:="$M$69", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$69"
    SolverAdd CellRef:="$M$69", Relation:=3, FormulaText:="0"
    SolverOk SetCell:="$M$69", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$69"
    Solversolve UserFinish:=True
    SolverDelete CellRef:="$M$69", Relation:=3, FormulaText:="0"
End If

    '   Fe3
If Range("$L$70").Value <= 0 Then
    Range("$N$70").Value = 0
Else
    Worksheets("Calc").btnSolverNet.Caption = "Solving Fe3 - Net Feed"
    SolverOk SetCell:="$M$70", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$70"
    SolverAdd CellRef:="$M$70", Relation:=3, FormulaText:="0"
    SolverOk SetCell:="$M$70", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$70"
    Solversolve UserFinish:=True
    SolverDelete CellRef:="$M$70", Relation:=3, FormulaText:="0"
End If

DoEvents
```

```
    '   Now Anions
        '   Cl
        If Range("$C$72").Value <= 0 Then
            Range("$C$74").Value = 0
        Else
            Worksheets("Calc").btnSolverNet.Caption = "Solving Cl - Net Feed"
            SolverOk SetCell:="$C$73", MaxMinVal:=2, ValueOf:="0", ByChange:="$C$74"
            SolverAdd CellRef:="$C$73", Relation:=3, FormulaText:="0"
            SolverOk SetCell:="$C$73", MaxMinVal:=2, ValueOf:="0", ByChange:="$C$74"
            Solversolve UserFinish:=True
            SolverDelete CellRef:="$C$73", Relation:=3, FormulaText:="0"
        End If

        '   SO4
        If Range("$D$72").Value <= 0 Then
            Range("$D$74").Value = 0
        Else
            Worksheets("Calc").btnSolverNet.Caption = "Solving SO4 - Net Feed"
            SolverOk SetCell:="$D$73", MaxMinVal:=2, ValueOf:="0", ByChange:="$D$74"
            SolverAdd CellRef:="$D$73", Relation:=3, FormulaText:="0"
            SolverOk SetCell:="$D$73", MaxMinVal:=2, ValueOf:="0", ByChange:="$D$74"
            Solversolve UserFinish:=True
            SolverDelete CellRef:="$D$73", Relation:=3, FormulaText:="0"
        End If

        '   HCO3
        If Range("$F$72").Value <= 0 Then
            Range("$F$74").Value = 0
        Else
            Worksheets("Calc").btnSolverNet.Caption = "Solving HCO3 - Net Feed"
            SolverOk SetCell:="$F$73", MaxMinVal:=2, ValueOf:="0", ByChange:="$F$74"
            SolverAdd CellRef:="$F$73", Relation:=3, FormulaText:="0"
            SolverOk SetCell:="$F$73", MaxMinVal:=2, ValueOf:="0", ByChange:="$F$74"
            Solversolve UserFinish:=True
            SolverDelete CellRef:="$F$73", Relation:=3, FormulaText:="0"
        End If

        '   HPO4
        If Range("$H$72").Value <= 0 Then
            Range("$H$74").Value = 0
        Else
            Worksheets("Calc").btnSolverNet.Caption = "Solving HPO4 - Net Feed"
            SolverOk SetCell:="$H$73", MaxMinVal:=2, ValueOf:="0", ByChange:="$H$74"
            SolverAdd CellRef:="$H$73", Relation:=3, FormulaText:="0"
            SolverOk SetCell:="$H$73", MaxMinVal:=2, ValueOf:="0", ByChange:="$H$74"
            Solversolve UserFinish:=True
            SolverDelete CellRef:="$H$73", Relation:=3, FormulaText:="0"
        End If

        DoEvents


        '   H2PO4
        If Range("$I$72").Value <= 0 Then
            Range("$I$74").Value = 0
        Else
            Worksheets("Calc").btnSolverNet.Caption = "Solving H2PO4 - Net Feed"
            SolverOk SetCell:="$I$73", MaxMinVal:=2, ValueOf:="0", ByChange:="$I$74"
            SolverAdd CellRef:="$I$73", Relation:=3, FormulaText:="0"
            SolverOk SetCell:="$I$73", MaxMinVal:=2, ValueOf:="0", ByChange:="$I$74"
            Solversolve UserFinish:=True
            SolverDelete CellRef:="$I$73", Relation:=3, FormulaText:="0"
        End If
        Worksheets("Calc").btnSolverNet.Caption = "Net Feed - Complete Pass " & Format(n, "#") &
" of " & Format(nLoops, "#")

    Next n
End If
If nSections <> 0 Then GoTo OuttaHere

SectionConcentrate:
```

```
'   CONCENTRATE
Worksheets("Calc").btnRunSolver.Width = nbtnWidth * 0.9
Worksheets("Calc").btnRunSolver.Caption = "Solving Conc."
For n = 1 To nLoops  ' loops if necessary
    DoEvents
    If boolAbortScaleCalcs = True Then
        boolAbortScaleCalcs = False
        If MsgBox("Abort Scaling Calcs?", vbOKCancel) = vbOK Then
            GoTo OuttaHere
        End If
    End If
    Worksheets("Calc").btnSolverConc.Caption = "Solving Na - Concentrate"
    SolverOk SetCell:="$M$94", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$94"
    SolverOptions MaxTime:=100, Iterations:=10000, Precision:=0.00000001, _
        AssumeLinear:=True, StepThru:=False, Estimates:=1, Derivatives:=1, SearchOption _
        :=1, IntTolerance:=0.001, Scaling:=True, Convergence:=0.001, AssumeNonNeg:=True

    '   Need to make sure old criteria are deleted
        SolverDelete CellRef:="$H$43", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M$34", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M$35", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M$36", Relation:=3, FormulaText:="0"
        '   $N$38 keeps appearing so get rid of it just in case.
        SolverDelete CellRef:="$N$38", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$H$73", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M$64", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M$65", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M$66", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$N$68", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$H$103", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M$94", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M$95", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$M$96", Relation:=3, FormulaText:="0"
        SolverDelete CellRef:="$N$98", Relation:=3, FormulaText:="0"

    DoEvents

    If Range("$L$94").Value <= 0 Then
        Range("$N$94").Value = 0
    Else
        SolverAdd CellRef:="$M$94", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$M$94", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$94"
        '   This UserFinish:=True causes the last section to be bypassed
        '   This is an undocumented feature of the Excel Solver
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$M$94", Relation:=3, FormulaText:="0"
    End If

    '   K
    If Range("$L$95").Value <= 0 Then
        Range("$N$95").Value = 0
    Else
        Worksheets("Calc").btnSolverConc.Caption = "Solving K - Concentrate"
        SolverOk SetCell:="$M$95", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$95"
        SolverAdd CellRef:="$M$95", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$M$95", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$95"
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$M$95", Relation:=3, FormulaText:="0"
    End If

        '   Ca
    If Range("$L$96").Value <= 0 Then
        Range("$N$96").Value = 0
    Else
        Worksheets("Calc").btnSolverConc.Caption = "Solving Ca - Concentrate"
        SolverOk SetCell:="$M$96", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$96"
        SolverAdd CellRef:="$M$96", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$M$96", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$96"
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$M$96", Relation:=3, FormulaText:="0"
```

```
    End If

        '   Mg
     If Range("$L$97").Value <= 0 Then
        Range("$N$97").Value = 0
    Else
        Worksheets("Calc").btnSolverConc.Caption = "Solving Mg - Concentrate"
        SolverOk SetCell:="$M$97", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$97"
        SolverAdd CellRef:="$M$97", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$M$97", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$97"
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$M$97", Relation:=3, FormulaText:="0"
    End If

    DoEvents

        '   Al
    If Range("$L$98").Value <= 0 Then
        Range("$N$98").Value = 0
    Else
        Worksheets("Calc").btnSolverConc.Caption = "Solving Al - Concentrate"
        SolverOk SetCell:="$M$98", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$98"
        'SolverDelete CellRef:="$M$97", Relation:=3, FormulaText:="0"
        SolverAdd CellRef:="$M$98", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$M$98", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$98"
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$M$98", Relation:=3, FormulaText:="0"
    End If

        '   Fe2
    If Range("$L$99").Value <= 0 Then
        Range("$N$99").Value = 0
    Else
        Worksheets("Calc").btnSolverConc.Caption = "Solving Fe2 - Concentrate"
        SolverOk SetCell:="$M$99", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$99"
        SolverAdd CellRef:="$M$99", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$M$99", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$99"
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$M$99", Relation:=3, FormulaText:="0"
    End If

        '   Fe3
    If Range("$L$100").Value <= 0 Then
        Range("$N$100").Value = 0
    Else
        Worksheets("Calc").btnSolverConc.Caption = "Solving Fe3 - Concentrate"
        SolverOk SetCell:="$M$100", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$100"
        SolverAdd CellRef:="$M$100", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$M$100", MaxMinVal:=2, ValueOf:="0", ByChange:="$N$100"
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$M$100", Relation:=3, FormulaText:="0"
    End If

    DoEvents

'   Now Anions
    '   Cl
    If Range("$C$102").Value <= 0 Then
        Range("$C$104").Value = 0
    Else
        Worksheets("Calc").btnSolverConc.Caption = "Solving Cl - Concentrate"
        SolverOk SetCell:="$C$103", MaxMinVal:=2, ValueOf:="0", ByChange:="$C$104"
        SolverAdd CellRef:="$C$103", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$C$103", MaxMinVal:=2, ValueOf:="0", ByChange:="$C$104"
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$C$103", Relation:=3, FormulaText:="0"
    End If

    '   SO4
    If Range("$D$102").Value <= 0 Then
        Range("$D$104").Value = 0
```

```
    Else
        Worksheets("Calc").btnSolverConc.Caption = "Solving SO4 - Concentrate"
        SolverOk SetCell:="$D$103", MaxMinVal:=2, ValueOf:="0", ByChange:="$D$104"
        SolverAdd CellRef:="$D$103", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$D$103", MaxMinVal:=2, ValueOf:="0", ByChange:="$D$104"
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$D$103", Relation:=3, FormulaText:="0"
    End If

    DoEvents

    '   HCO3
    If Range("$F$102").Value <= 0 Then
        Range("$F$104").Value = 0
    Else
        Worksheets("Calc").btnSolverConc.Caption = "Solving HCO3 - Concentrate"
        SolverOk SetCell:="$F$103", MaxMinVal:=2, ValueOf:="0", ByChange:="$F$104"
        SolverAdd CellRef:="$F$103", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$F$103", MaxMinVal:=2, ValueOf:="0", ByChange:="$F$104"
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$F$103", Relation:=3, FormulaText:="0"
    End If

    '   HPO4
    If Range("$H$102").Value <= 0 Then
        Range("$H$104").Value = 0
    Else
        Worksheets("Calc").btnSolverConc.Caption = "Solving HPO4 - Concentrate"
        SolverOk SetCell:="$H$103", MaxMinVal:=2, ValueOf:="0", ByChange:="$H$104"
        SolverAdd CellRef:="$H$103", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$H$103", MaxMinVal:=2, ValueOf:="0", ByChange:="$H$104"
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$H$103", Relation:=3, FormulaText:="0"
    End If


    '   H2PO4
    If Range("$I$102").Value <= 0 Then
        Range("$I$104").Value = 0
    Else
        Worksheets("Calc").btnSolverConc.Caption = "Solving H2PO4 - Concentrate"
        SolverOk SetCell:="$I$103", MaxMinVal:=2, ValueOf:="0", ByChange:="$I$104"
        SolverAdd CellRef:="$I$103", Relation:=3, FormulaText:="0"
        SolverOk SetCell:="$I$103", MaxMinVal:=2, ValueOf:="0", ByChange:="$I$104"
        Solversolve UserFinish:=True
        SolverDelete CellRef:="$I$103", Relation:=3, FormulaText:="0"
    End If
    Worksheets("Calc").btnSolverConc.Caption = "Concentrate - Complete Pass " & Format(n, "#") &
" of " & Format(nLoops, "#")
Next n

'   we're done
OuttaHere:
    Worksheets("Calc").btnRunSolver.Caption = "Start Solver - ALL"
    Worksheets("Calc").btnRunSolver.BackColor = &HE0E0E0
    Worksheets("Calc").btnRunSolver.Width = nbtnWidth
    Worksheets("Calc").btnSolverTreated.Caption = "Start Solver - Treated Feed"
    Worksheets("Calc").btnSolverNet.Caption = "Start Solver - Net Feed"
    Worksheets("Calc").btnSolverConc.Caption = "Start Solver - Concentrate"
End Sub
```

## ModMembranes

This module contains the code used by the list boxes on the System Design sheet and the Add/Edit membrane functions.

```
Attribute VB_Name = "ModMembranes"
Option Explicit
Global tempStop As Boolean
```

```
Public ListLabels(10) As String
Public MembraneList() As String

'   The main variable for the membrane Element type
'   The naming is confusing, since the membrane and element are combined
Public Type Membrane
    Description As String
    Diameter As Single
    Length As Single
    Type As Integer
    NaCl_rejection As Double
    MgSO4_rejection As Double
    TestPressure As Double
    TestTDS As Double
    TestFlow As Double
    TestTemperature As Double
    Area As Double

    '   The values below as Calculated Values from the spec sheet data
    Avalue As Double
    BValue As Double
    BIons(LAST_ION) As Double

    '   These are set according to the type of membrane
    BTempCorr As Double
    ATempCorr As Double
End Type
Public MembAvail() As Membrane
Public MyMembrane As Membrane
```

*The "tempstop" variable is used at many places to avoid looping through a control or function. Activating or selecting a cell or button calls the "click" event. This variable allows the click event to be ignored or used as called for.*

```
'   Sub to load the available types - uses MembraneData sheet information
Public Sub LoadTypeList()
    tempStop = True
    Dim n As Integer, NT As Integer
    Worksheets("Membrane Data").Activate
    Worksheets("Membrane Data").Range("ListLabels").Select
    NT = Worksheets("Membrane Data").Range("NumberTypes").Value
    For n = 1 To NT
        ActiveCell.Offset(1, 0).Select
        ListLabels(n) = ActiveCell.Value
    Next
    Worksheets("System Design").lstMembraneType.Clear
    For n = 1 To NT
        Worksheets("System Design").lstMembraneType.AddItem Trim(ListLabels(n))
    Next
    Worksheets("System Design").lstMembraneType.AddItem "Create New Element Type"
    Worksheets("System Design").Activate
    If Worksheets("System Design").Range("FileOpened").Value = "True" Then
        Worksheets("System Design").lstMembraneType.ListIndex = 0
        Worksheets("System Design").Range("TypeIndex").Value = 0
    Else
        Worksheets("System Design").lstMembraneType.ListIndex = Worksheets("System
Design").Range("TypeIndex").Value
    End If
    tempStop = False
End Sub


Public Sub GetMembraneList(mt As Integer)
    tempStop = True
    Dim dummy As Double
    Dim n As Integer, m As Integer
    Dim BeginRecord As Long, EndRecord As Long, TypeCount As Integer
'    If boolDebug2 Then MsgBox "loading list"
    TypeCount = Worksheets("Membrane Data").Range("TypeCount").Offset(mt, 0).Value
    BeginRecord = Worksheets("Membrane Data").Range("BeginRow").Offset(mt, 0).Value
```

```
    EndRecord = BeginRecord + (TypeCount - 1)

    ReDim MembraneList(4, TypeCount)
    ReDim MembAvail(TypeCount) As Membrane

    '   now rebuild the element type list box with the right elements
    Worksheets("System Design").lstSelectMembrane.Clear

    If TypeCount = 0 Then
        m = MsgBox("There are no membranes of the selected type in the" + Chr(13) + _
        "membrane database.", vbOKOnly, "List Empty")
        Worksheets("System Design").lstSelectMembrane.ListIndex = -1
        tempStop = False
        Exit Sub
    End If

'   get a brief description into the record and save it into the membrane variable MembAvail(n)
'   This will later use the ListIndex of the lstSelectMembrane as the array pointer to get the
data
    For n = 1 To TypeCount
        With MembAvail(n)
        MembraneList(1, n) = Trim(Range("Description").Offset(BeginRecord + 2 + n - 1, 0).Value)
'Trim(ActiveCell.Value)
        MembraneList(2, n) = Trim(Range("Diameter").Offset(BeginRecord + 2 + n - 1, 0).Value)
'Trim(ActiveCell.Value)
        MembraneList(3, n) = Trim(Range("Length").Offset(BeginRecord + 2 + n - 1, 0).Value)
'Trim(ActiveCell.Value)
        MembraneList(4, n) = Trim(Range("Elem_Area").Offset(BeginRecord + 2 + n - 1, 0).Value)

        .Description = MembraneList(1, n)
        .Diameter = MembraneList(2, n)
        .Length = MembraneList(3, n)
        .Type = Range("Type").Offset(BeginRecord + 2 + n - 1, 0).Value
        .NaCl_rejection = Range("NaCl_Rejection").Offset(BeginRecord + 2 + n - 1, 0).Value
        .MgSO4_rejection = Range("MgSO4_Rejection").Offset(BeginRecord + 2 + n - 1, 0).Value
        .TestPressure = Range("Test_Pressure").Offset(BeginRecord + 2 + n - 1, 0).Value
        .TestTDS = Range("Test_Tds").Offset(BeginRecord + 2 + n - 1, 0).Value
        .TestFlow = Range("Test_Flow_Rate").Offset(BeginRecord + 2 + n - 1, 0).Value
        .TestTemperature = Range("Test_Temperature").Offset(BeginRecord + 2 + n - 1, 0).Value
        .Area = Range("Elem_Area").Offset(BeginRecord + 2 + n - 1, 0).Value

        '   The values below as Calculated Values from the spec sheet data and are not needed now
as they
        '   are the same for all elements of this type mem,brea
        'Avalue As Double
        'BValue As Double
        'BIons(MAX_IONS) As Double

        '   These are set according to the type of membrane but can
        'BTempCorr As Double
        'ATempCorr As Double

    End With
    Next n

    Worksheets("System Design").Activate
    For n = 1 To TypeCount
        Worksheets("System Design").lstSelectMembrane.AddItem Trim(MembraneList(1, n)) & ", " &
Trim(MembraneList(2, n)) & " in. Dia., " & _
        Trim(MembraneList(3, n)) & " in. Len., " & Trim(MembraneList(4, n)) & " ft^2"
    Next
    If Worksheets("System Design").Range("FileOpened").Value = "True" Then
        Worksheets("System Design").lstSelectMembrane.ListIndex = 0
    Else
        Worksheets("System Design").lstSelectMembrane.ListIndex = Worksheets("System
Design").Range("MembraneIndex").Value
    End If
    n = Worksheets("System Design").lstSelectMembrane.ListIndex + 1

    '   now put the selected element in the 1st row of Membrane Data so the calculation engine
can grab it.
```

```
    With MembAvail(n)
        Range("Description").Offset(-1, 0).Value = .Description
        Range("Diameter").Offset(-1, 0).Value = .Diameter
        Range("Length").Offset(-1, 0).Value = .Length
        Range("Type").Offset(-1, 0).Value = .Type
        Range("NaCl_Rejection").Offset(-1, 0).Value = .NaCl_rejection
        Range("MgSO4_Rejection").Offset(-1, 0).Value = .MgSO4_rejection
        Range("Test_Pressure").Offset(-1, 0).Value = .TestPressure
        Range("Test_Tds").Offset(-1, 0).Value = .TestTDS
        Range("Test_Flow_Rate").Offset(-1, 0).Value = .TestFlow
        Range("Test_Temperature").Offset(-1, 0).Value = .TestTDS
        Range("Elem_Area").Offset(-1, 0).Value = .Area
    End With
    tempStop = False
End Sub
```

*When the user selects the membrane – eg a 4" diameter, TFC membrane element. The specifics of that element are copied to the first row of the Membrane Data sheet. The calculation engine then looks in this first row to get the data it needs to complete the system design.*

```
Public Sub GetMembraneData(MembraneType As Integer, MembraneIndex As Integer)
Dim n As Integer
Dim nOffset As Integer
Dim dummy As Double
    n = Worksheets("System Design").lstSelectMembrane.ListIndex + 1
        '   now put the selected element in the 1st row of Membrane Data so the calculation
engine can grab it.
    On Error Resume Next
    With MembAvail(n)
        '   Err.Raise 9  ' test to be sure we have initialized the variables
        If Err.Number = 9 Then
            nOffset = Worksheets("System Design").lstMembraneType.ListIndex + 1
            GetMembraneList nOffset
            On Error GoTo 0
        End If
        Range("Description").Offset(-1, 0).Value = .Description
        Range("Diameter").Offset(-1, 0).Value = .Diameter
        Range("Length").Offset(-1, 0).Value = .Length
        Range("Type").Offset(-1, 0).Value = .Type
        Range("NaCl_Rejection").Offset(-1, 0).Value = .NaCl_rejection
        Range("MgSO4_Rejection").Offset(-1, 0).Value = .MgSO4_rejection
        Range("Test_Pressure").Offset(-1, 0).Value = .TestPressure
        Range("Test_Tds").Offset(-1, 0).Value = .TestTDS
        Range("Test_Flow_Rate").Offset(-1, 0).Value = .TestFlow
        Range("Test_Temperature").Offset(-1, 0).Value = .TestTDS
        Range("Elem_Area").Offset(-1, 0).Value = .Area
    End With
    tempStop = False
End Sub
```

## System Design Sheet Code

The code associated and stored on this sheet is used for several important calculations. In addition, this section contains an explanation of how the bank and flow graphics work.

```
Option Explicit
```

*Constants used to enter the flow graphic calculation*

```
Const ENTER_FEED = 1
Const ENTER_INTRECOVERY = 2
Const ENTER_SYSRECOVERY = 3
Const ENTER_PRODUCT = 4
Const ENTER_CONC = 5
Const ENTER_RECYCLE = 6
Const ENTER_UNITS = 7   ' used for changing units - called from Flow_Select_Drop_Click
Public NewType As Boolean
```

```
Private Sub btnAddMembrane_Click()
    Worksheets("Membrane Data").Activate
    frmMembraneData.Show
End Sub

Public Sub btnCalculate_Click()
    MainCalc
End Sub

Private Sub CommandButton1_Click()
    lblRecalc.Visible = False
    msgRecalc.Visible = False
End Sub

Private Sub btnRestoreDefault_Click()
        Worksheets("Membrane Data").Activate
        Worksheets("Membrane Data").Range("Type" & Trim(Str(Worksheets("Membrane Data").Range _
        ("NumberTypes").Value + 1))).Offset(1, 0).Select
End Sub

Private Sub btnStreams_Click()
    Worksheets("Streams").Activate
    Worksheets("Streams").Range("B10").Select
End Sub

Private Sub btnSummary_Click()
    Worksheets("Summary").Activate
    Worksheets("Summary").Range("B10").Select
End Sub
```

*The list box for membrane types – ie nanofiltration, cellulose acetate, etc.*

```
Private Sub lstMembraneType_Change()
    If tempStop Then Exit Sub
    Application.ScreenUpdating = False
    If lstMembraneType.ListIndex + 1 = lstMembraneType.ListCount Then
        NewType = True
        tempStop = True
        lstMembraneType.ListIndex = 0
        tempStop = False
        Worksheets("System Design").Range("TypeIndex").Value = 0
        Worksheets("Membrane Data").Activate
        Application.ScreenUpdating = True
        Worksheets("Membrane Data").Range("Type" & Trim(Str(Worksheets("Membrane Data").Range _
        ("NumberTypes").Value + 1))).Offset(1, 0).Select

        Exit Sub
    Else
        NewType = False
        '   need to remember ListIndex starts at 0 so need to add 1 if first number is 1
        GetMembraneList lstMembraneType.ListIndex + 1
        Worksheets("System Design").Range("TypeIndex").Value = lstMembraneType.ListIndex
        setChanged
    End If
    Application.ScreenUpdating = True
    Worksheets("System Design").Range("J10").Select

End Sub

Private Sub lstSelectMembrane_Change()
    If Not NewType Then setChanged
    GetMembraneData lstMembraneType.ListIndex + 1, lstSelectMembrane.ListIndex + 1
    If Worksheets("System Design").Range("FileOpened").Value = "True" Then Worksheets("System
Design").Range("MembraneIndex").Value = lstSelectMembrane.ListIndex
End Sub


Private Sub spNumberBanks_Change()
    Dim n As Integer
    n = Range("NumberBanks").Value
    setChanged
```

```
End Sub

Private Sub spNumberBanks_SpinDown()
    Dim n As Integer
    n = Range("NumberBanks").Value
    SendToBack n
End Sub

Private Sub spNumberBanks_SpinUp()
    Dim n As Integer
    n = Range("NumberBanks").Value
    BringToFront n
End Sub

Private Sub spNumberElementsPerVessel_Change()
    setChanged
End Sub
```

*Change the number of banks using the spinners. The rectangles (ActiveSheet.Shapes) are always in existence. They either hide behind or move to the front. Since Excel assigns an index to each shape as it is created, we had no control over the index. Thus the first bank shapes start at 196, Bank 2 at 216, and so on. The .Zorder command is the command for front or back.*

```
Private Sub spBank1_SpinDown()
Dim n As Integer
Dim base1 As Integer
Dim strName As String
    setChanged
    n = Range("Bank1").Value
    If n < 1 Then n = 1
    base1 = 196
    If n > 19 Then Exit Sub
    strName = Trim(Str(base1 + n))
    ActiveSheet.Shapes(strName).Select
    Selection.ShapeRange.ZOrder msoSendToBack
    Range("Bank1").Select
End Sub

Private Sub spBank1_SpinUp()
    Dim n As Integer
    Dim base1 As Integer
    Dim strName As String
    setChanged
    n = Range("Bank1").Value
    base1 = 195
    If n > 20 Then Exit Sub
    strName = Trim(Str(base1 + n))
    ActiveSheet.Shapes(strName).Select
    Selection.ShapeRange.ZOrder msoBringToFront
    Range("Bank1").Select
End Sub

Private Sub spBank2_Change()
setChanged
End Sub

Private Sub spBank2_SpinDown()
    Dim n As Integer
    Dim base2 As Integer
    Dim strName As String
    setChanged
    n = Range("Bank2").Value
    If n < 1 Then n = 1
    base2 = 216
    If n > 19 Then Exit Sub
    strName = Trim(Str(base2 + n))
    ActiveSheet.Shapes(strName).Select
    Selection.ShapeRange.ZOrder msoSendToBack
    Range("Bank2").Select
```

```
End Sub

Private Sub spBank2_SpinUp()
    Dim n As Integer
    Dim base2 As Integer
    Dim strName As String
    setChanged
    n = Range("Bank2").Value
    base2 = 215
    If n > 20 Then Exit Sub
    strName = Trim(Str(base2 + n))
    ActiveSheet.Shapes(strName).Select
    Selection.ShapeRange.ZOrder msoBringToFront
    Range("Bank2").Select
End Sub

Private Sub SendToBack(n As Integer)
    Dim m As Integer, base As Integer
    Dim strName As String
    If n < 4 Then
        base = 256
        For m = base To base + 19
            strName = "Rectangle " + Trim(Str(m))
            'On Error Resume Next
            ActiveSheet.Shapes(strName).Select
            Selection.ShapeRange.ZOrder msoSendToBack
        Next
        spBank4.Enabled = False
        Range("Bank4").Value = 0
        Range("NumberBanks").Select
    End If
    If n < 3 Then
        base = 236
        For m = base To base + 19
            strName = "Rectangle " + Trim(Str(m))
            ActiveSheet.Shapes(strName).Select
            Selection.ShapeRange.ZOrder msoSendToBack
        Next
        spBank3.Enabled = False
        Range("Bank3").Value = 0
        Range("NumberBanks").Select
    End If
    If n < 2 Then
        base = 216
        For m = base To base + 19
            strName = "Rectangle " + Trim(Str(m))
            ActiveSheet.Shapes(strName).Select
            Selection.ShapeRange.ZOrder msoSendToBack
        Next
        spBank2.Enabled = False
        Range("Bank2").Value = 0
        Range("NumberBanks").Select
    End If
    If n < 1 Then
        base = 196
        For m = base To base + 19
            strName = "Rectangle " + Trim(Str(m))
            ActiveSheet.Shapes(strName).Select
            Selection.ShapeRange.ZOrder msoSendToBack
        Next
        spBank1.Enabled = False
        Range("Bank1").Value = 0
        Range("NumberBanks").Select
    End If
End Sub

Private Sub BringToFront(n As Integer)
    Dim strName As String
    Select Case n
    Case 1
        strName = "Rectangle " + Trim(Str(196))
```

```
        ActiveSheet.Shapes(strName).Select
        Selection.ShapeRange.ZOrder msoBringToFront
        Range("Bank1").Value = 1
        spBank1.Enabled = True
        Range("NumberBanks").Select
    Case 2
        strName = "Rectangle " + Trim(Str(216))
        ActiveSheet.Shapes(strName).Select
        Selection.ShapeRange.ZOrder msoBringToFront
        Range("Bank2").Value = 1
        spBank2.Enabled = True
        Range("NumberBanks").Select
    Case 3
        strName = "Rectangle " + Trim(Str(236))
        ActiveSheet.Shapes(strName).Select
        Selection.ShapeRange.ZOrder msoBringToFront
        Range("Bank3").Value = 1
        spBank3.Enabled = True
        Range("NumberBanks").Select
    Case 4
        strName = "Rectangle " + Trim(Str(256))
        ActiveSheet.Shapes(strName).Select
        Selection.ShapeRange.ZOrder msoBringToFront
        Range("Bank4").Value = 1
        spBank4.Enabled = True
        Range("NumberBanks").Select
    End Select
End Sub

Private Sub spBank3_Change()
    setChanged
End Sub

Private Sub spBank3_SpinDown()
    Dim n As Integer
    Dim base3 As Integer
    Dim strName As String
    setChanged
    n = Range("Bank3").Value
    If n < 1 Then n = 1
    base3 = 236
    If n > 19 Then Exit Sub
    strName = Trim(Str(base3 + n))
    ActiveSheet.Shapes(strName).Select
    Selection.ShapeRange.ZOrder msoSendToBack
    Range("Bank3").Select
End Sub

Private Sub spBank3_SpinUp()
    Dim n As Integer
    Dim base3 As Integer
    Dim strName As String
    setChanged
    n = Range("Bank3").Value
    base3 = 235
    If n > 20 Then Exit Sub
    strName = Trim(Str(base3 + n))
    ActiveSheet.Shapes(strName).Select
    Selection.ShapeRange.ZOrder msoBringToFront
    Range("Bank3").Select
End Sub

Private Sub spBank4_Change()
    setChanged
End Sub

Private Sub spBank4_SpinDown()
    Dim n As Integer
    Dim base4 As Integer
    Dim strName As String
    setChanged
```

```
    n = Range("Bank4").Value
    If n < 1 Then n = 1
    base4 = 256
    If n > 19 Then Exit Sub
    strName = Trim(Str(base4 + n))
    ActiveSheet.Shapes(strName).Select
    Selection.ShapeRange.ZOrder msoSendToBack
    Range("Bank4").Select
End Sub

Private Sub spBank4_SpinUp()
    Dim n As Integer
    Dim base4 As Integer
    Dim strName As String
    setChanged
    n = Range("Bank4").Value
    base4 = 255
    If n > 20 Then Exit Sub
    strName = Trim(Str(base4 + n))
    ActiveSheet.Shapes(strName).Select
    Selection.ShapeRange.ZOrder msoBringToFront
    Range("Bank4").Select
End Sub
```

*The flow rate calls begin here. We first determine which spinner or flow rate was changed then recalculate accordingly.*

```
Private Sub spConcentrateFlow_Change()
    If tempStop Then Exit Sub
    tempStop = True
    Range("ProductFlow").Value = Range("FeedFlow").Value - Range("ConcentrateFlow").Value
    Range("SystemRecovery").Value = (Range("ProductFlow").Value / Range("FeedFlow").Value) * 100
    Range("InternalRecovery").Value = CInt((Range("ProductFlow").Value /
(Range("RecycleFlow").Value + Range("FeedFlow").Value)) * 100)
    tempStop = False
    setChanged
End Sub

Private Sub spFeedFlow_Change()
If tempStop Then Exit Sub
    UpdateFlows ENTER_FEED
End Sub


Private Sub spInternalRecovery_Change()
If tempStop Then Exit Sub
    UpdateFlows ENTER_INTRECOVERY
End Sub

Private Sub spProductFlow_Change()
If tempStop Then Exit Sub
    UpdateFlows ENTER_PRODUCT
End Sub

Private Sub spRecycleFlow_Change()
If tempStop Then Exit Sub
    UpdateFlows ENTER_RECYCLE
End Sub

Private Sub spSystemRecovery_Change()
If tempStop Then Exit Sub
    UpdateFlows ENTER_SYSRECOVERY
End Sub


Private Sub Worksheet_Activate()
    Dim n As Integer

    If Worksheets("System Design").Range("FileOpened").Value = "False" And Not tempStop Then
        LoadTypeList
```

```
        tempStop = True
        lstMembraneType.ListIndex = Range("TypeIndex").Value
        GetMembraneList Range("TypeIndex").Value + 1
        n = CInt(Range("MembraneIndex").Value)
        Worksheets("System Design").lstSelectMembrane.ListIndex = n
        tempStop = False
        Worksheets("System Design").Range("FileOpened").Value = "True"
    ElseIf Not tempStop And lstMembraneType.ListCount <> Worksheets("Membrane
Data").Range("NumberTypes").Value + 1 Then
        LoadTypeList
        GetMembraneList 1
    End If
End Sub

Private Sub Worksheet_Change(ByVal Target As Excel.Range)
Dim nCount As Integer

nCount = Range("nCount").Value

    If nCount = 1 Then Exit Sub
    If lblRecalc.Visible Then
        Exit Sub
    Else
        msgRecalc.Visible = True
        lblRecalc.Visible = True
        'MsgBox ("Data has changed - You will need to redo the calculations")
        Range("Ncount").Value = 1
    End If
End Sub

Private Sub setChanged()
    Range("nCount").Value = 0
    Worksheets("Summary").Range("SystemCalculated").Value = "Needs Recalc"
    Worksheets("Streams").Range("SystemCalculated1").Value = "Needs Recalc"
End Sub
```

*Flow rates are calculated here depending on entry value. nEntering is one of the constants defined at the start of the module.*

```
Sub UpdateFlows(nEntering As Integer)
Dim fFeedTDS As Double
Dim fnetFeedTDS As Double
Dim fConcTDS As Double

Dim fFeed As Double
Dim fProduct As Double
Dim fConc As Double
Dim fRecycle As Double
Dim fSysRecov As Double
Dim fIntRecov As Double
Dim fNetFeed As Double
fFeed = Range("Feedflow").Value
fProduct = Range("productflow").Value
fConc = Range("ConcentrateFlow").Value
fRecycle = Range("Recycleflow").Value
fSysRecov = Range("SystemRecovery").Value / 100
fIntRecov = Range("InternalRecovery").Value / 100
fNetFeed = fFeed + fRecycle

Select Case nEntering
    Case ENTER_UNITS
    '   only changed units and now have a new feed value to work from
    '   recoveries unchanged, recalc flows from new feed
        fProduct = fFeed * fSysRecov
        fConc = fFeed - fProduct
        If Abs(fIntRecov - fSysRecov) <= 0.001 Or fIntRecov = fSysRecov Then
            fRecycle = 0#
        Else
            fRecycle = (fProduct - fIntRecov * fFeed) / fIntRecov
        End If
        fNetFeed = fFeed + fRecycle
```

```
    Case ENTER_FEED
    '   recovery is unchanged, flows to match recovery
        fProduct = fFeed * fSysRecov
        fConc = fFeed - fProduct
        fNetFeed = fFeed + fRecycle
        fIntRecov = fProduct / fNetFeed
    Case ENTER_PRODUCT
    '   product flow has changed, Hold recovery constant, change the feed as needed, frecycle
unchanged
        fFeed = fProduct / fSysRecov
        fNetFeed = fFeed + fRecycle
        fIntRecov = fProduct / fNetFeed
        fConc = fFeed - fProduct
    Case ENTER_CONC
    '   changed the concentrate,  modify the recovery as needed, other flows unchanged
    '   but first check for max recovery - 95%
        If fConc / fFeed < 0.05 Then fConc = 0.05 * fFeed
    '   now catch the case where concentrate is too high
        If fConc / fFeed > 0.95 Then fConc = 0.95 * fFeed
        fSysRecov = 1 - fConc / fFeed
    Case ENTER_RECYCLE
    '   recycle flow is changed
        fNetFeed = fFeed + fRecycle
        fIntRecov = fProduct / fNetFeed

    Case ENTER_SYSRECOVERY
    '   hold the permeate flow constant
        fFeed = fProduct / fSysRecov
        fConc = fFeed - fProduct
        fNetFeed = fFeed + fRecycle
        fIntRecov = fProduct / fNetFeed
        If Abs(fIntRecov - fSysRecov) <= 0.0001 Or fIntRecov = fSysRecov Then
            fIntRecov = fSysRecov
            fRecycle = 0#
            fNetFeed = fFeed


        End If

    Case ENTER_INTRECOVERY
        If Abs(fIntRecov - fSysRecov) <= 0.001 Or fIntRecov = fSysRecov Then
            fRecycle = 0#
            fIntRecov = fSysRecov
            '   internal recovery cannot be greater than the system recovery
            ElseIf fSysRecov < fIntRecov Then
                fIntRecov = fSysRecov
                fRecycle = 0#
                fNetFeed = fFeed
            Else
                fRecycle = (fProduct - fIntRecov * fFeed) / fIntRecov
                fNetFeed = fFeed + fRecycle
        End If
End Select
'   now fill in the data and edit boxes
tempStop = True
    Range("Feedflow").Value = fFeed
'        txtFeed.Text = SetFormat(fFeed)
    Range("productflow").Value = fProduct
        'txtProduct.Text = SetFormat(fProduct)
    Range("ConcentrateFlow").Value = fConc
        'txtConc.Text = SetFormat(fConc)
    Range("Recycleflow").Value = fRecycle
        'txtRecycle.Text = SetFormat(fRecycle)
    Range("SystemRecovery").Value = fSysRecov * 100
        'txtSysRecovery.Text = SetFormat(fSysRecov * 100)
    Range("InternalRecovery").Value = fIntRecov * 100
        'txtIntRecovery.Text = SetFormat(fIntRecov * 100)
    lblNetFeed.Caption = "Net = " & SetFormat(fFeed + fRecycle)

    fFeedTDS = Worksheets("Analysis").Range("TDS_Treated").Value
```

```
    fConcTDS = (fFeedTDS - 0.01 * fFeedTDS * fSysRecov) / (1 - fSysRecov)

    If fRecycle > 0 Then
        fnetFeedTDS = (fFeedTDS * fFeed + fConcTDS * fRecycle) / (fFeed + fRecycle)
    Else
        fnetFeedTDS = fFeedTDS
    End If
    Range("ProductEstimated").Value = 0.01 * (fnetFeedTDS + fConcTDS) / 2
    Range("Net_Feed_Estimated").Value = fnetFeedTDS
setChanged
tempStop = False
End Sub

Private Sub Worksheet_SelectionChange(ByVal Target As Excel.Range)
'   need this to update if only a small change to recovery is made
Dim nRow As Integer, nCol As Integer
nRow = Target.Row
nCol = Target.Column
If nRow = Range("SystemRecovery").Row And nCol = Range("SystemRecovery").Column Then
        UpdateFlows ENTER_SYSRECOVERY
    ElseIf nRow = Range("InternalRecovery").Row And nCol = Range("InternalRecovery").Column Then
        UpdateFlows ENTER_INTRECOVERY
End If

End Sub
```